A Case Study in Recommender Systems for Click Prediction

Kevin Antony, Utkarsh Lal, Bonnie Liu, Nisha McNealis, Siddharth Nandy

1 Abstract

This project investigates the efficacy of various models for click-through rate (CTR) prediction tasks in online advertising. We aimed to predict user click behavior on items provided by Taboola.com, leveraging techniques such as SVD, neural collaborative filtering, and other architectures used for recommendation tasks. The dataset consisted of labeled instances indicating whether a user clicked on a specific item. Our findings indicate that among the models explored, the Transformer model exhibited the highest accuracy with 74.1%, followed closely by DeepFM with 73.5%. Neural CF also demonstrated promising performance with an accuracy of 73.4%. These results underscore the efficacy of deep learning-based approaches for CTR prediction tasks. Furthermore, our study sheds light on the challenges associated with recommender systems in real-world settings, especially as they relate to computational constraints of large models and the sparse nature of data used for recommendation tasks.

2 Introduction

Digital advertising is a multi-billion-dollar industry, with global spend expected to continually rise. This project revolves around optimizing recommendation systems for CTRs, a critical metric in digital advertising and content distribution. Simple CTR prediction models often fall short in capturing complex nonlinear relationships and interactions between users and items. By leveraging advanced recommendation algorithms, our project is able to predict user interactions with online content with marginally better accuracy than baseline models. This predictive capability can significantly enhance personalized content delivery across various industries, from e-commerce to online media, improving user engagement and satisfaction. Furthermore, the application of this project can lead to more efficient ad placement, driving revenue growth and better resource allocation for businesses in the digital space.

3 Literature Review

Click-through rate (CTR) prediction is a well-studied task in machine learning. In this section, we provide an overview of several notable prior works.

3.1 Deep Interest Network for Click-Through Rate Prediction

In 2019, Zhou et al. proposed a Deep Interest Network (DIN) for CTR prediction [1]. The key contribution of this model is an adaptive representation vector that varies depending on the ad. Traditional CTR models use a fixed-length representation vector, which is unable to capture diverse user behaviors. In addition to the representation vector, the authors also introduce two techniques to further improve the model's predictions. First, they use mini-batch aware regularization, meaning that they only calculate the L2-norm over the parameters of sparse features appearing in each mini-batch. This makes their regularization process much more computationally efficient while still reducing overfitting. Second, they create a data adaptive activation function by using a modified form of PReLU that adjusts based on the distribution of the input data. They note that both of these techniques

can be applied to other models and tasks. The authors find that their model is able to outperform a wide range of competitors, especially on a rich dataset.

3.2 Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction

Liu et al. (2019) [2] discusses the feasibility of using convolutional neural networks (CNNs) to augment the feature spaces of CTR problems, a persistent problem due to the sparse nature of useful interactions. Previous solutions involve artificially creating features, but feature engineering is typically impractical due to being expensive and requiring domain knowledge. The proposed Feature Generation by Convolutional Neural Network (FGCNN) model has two components: feature generation, through which a CNN generates local patters and recombines them for new features, and deep classifier, which learns from the augmented feature space. In feature generation, the authors introduce a recombination layer after the convolutional and pooling layers pairs in order to capture global feature interactions for the deep classifier. The results from the paper show that this experimental paradigm yielded superior performance across three different datasets when compared to nine other models. Furthermore, the authors showed that combining the feature generation component with some of the baseline models yielded superior performance, highlighting its flexibility in architecture selection for CTR problems.

3.3 Modeling Feature Interactions via Graph Neural Networks for CTR Prediction

Li et al. (2019) explores the use of graph neural networks (GNNs) to better learn high-order feature interactions [3]. They propose a new model Feature interaction Graph Neural Networks (Fi-GNNs) that are based on GNNs to to model sophisticated feature interactions in a more flexible and explicit way than compared to other Deep Neural Networks (DNNs). The Fi-GNN model is able to differ from previous CTR models that don't utilize GNNs through the use of a flexible edge-wise interaction function which is more effective at modeling feature interaction. It also improves on prior GNN models which typically use binary edge weights that fail to reflect the importance of the relations while the Fi-GNN models edge-wise interactions through attention edge weights and edge-wise transformation functions which better highlights relations. Additionally it introduces extra residual connections in addition to just Gated Recurrent Unit (GRU) that conventional GGNN's use to allow for low order feature reuse. The authors found that across 2 datasets that their model outperformed others for CTR predictions showcasing the usefulness of leveraging graph representation for the task.

3.4 LibRec: A Leading Java Library for Recommender Systems

This paper by Guo et al. introduces LibRec, an easy-to-use recommender system focused on end-to-end recommendation process [4]. It allows users to train and deploy different kinds of recommendation models, and the authors found that LibRec performs faster than existing recommendation libraries while maintaining the same performance. We use a modified version of LibRec in our methods below.

4 Methodology

4.1 Dataset

The dataset, provided by Taboola.com, consists of pairs of users and items and is labeled based on whether or not the user clicked on the subject item. There were 30,253,436 usable samples, spanning 16,263,893 users interacting with 95,073 items. In addition to the clicked or not-clicked label, every data instance consisted of three types of features: user features, which stored information regarding the respective user's general interaction history; item features, which stored information about the item's engagement and click history; and context features, which stored metadata about the user-item interaction itself such as time, device platform, etc. Initial testing with our random forest baseline model found that user features and certain item features, specifically those concerning an item's empiric impression and engagement history, had more predictive power than context features like OS family and the day of the week of the user-item interaction.

A notable characteristic of the dataset is its sparsity: namely, of the 16 million users in the entire dataset, over five million users only interacted with one item. As a result, we elected to use a subset of the dataset for our experiments; namely, we only included users who interacted with at least

eight items. Furthermore, to preserve the balance between labels in the overall dataset, we also undersampled the dominant class in the subset (a positive click). The resultant cleaned dataset had 462,030 users and 55,876 items across 4,246,524 user-item pairs. While this data cleaning did not completely solve the sparsity issue, it made our experiments more feasible with our limited compute resources. We believe that focusing on this subset made the experiment more applicable by placing the emphasis on more 'active' users as opposed to those with very few interactions with the items within our dataset.

4.2 Methods

In order to discover the best model that fit our dataset we tested several different model architectures to see what would yield the highest performance. We began with SVD[5] which utilizes matrix factorization techniques to uncover latent factors related to user and item interactions. Additionally, we tested SVD++[6], an extension of the SVD model that incorporates neighborhood methods that are centered on computing the relationships between items or between users and further improves accuracy by taking into account implicit feedback of users to ultimately better learn the user-item matrix. In addition to Collaborative Filtering (CF) models we also wanted to judge the performance of various hybrid recommender systems that employed other techniques to account for data aside from user-item interactions to better predict click through rate. Recently, many models have employed the use of Factorization Machines (FM) with the use of Deep Neural Networks(DNNs) to better learn feature interactions. We wanted to judge the performance of these models and tested Wide&Deep[7] which trains wide linear models and deep neural networks to produce a more robust recommender system and DeepFM[8] which leverages these factorization machines as well as deep neural network to accomplish the goal of feature learning. We also tested a Neural Collaborative Filtering Model (NCF)[9] which combined traditional collaborative filtering methods and also utilizes neural networks to learn feature interactions. Finally, we tested a Transformer model[10] to see how it compared to the various other models. One thing to note is that the models we tested are mainly used for rating tasks and not the binary classification that would tell if an item was clicked by a user that we were attempting to accomplish. In order to achieve this we ranked the items between 0 and 1 and found a threshold to divide the space between clicked and not clicked items. In doing so we were able to adapt the different model architectures to our task and utilize them effectively.

Listing 1: LibRecommender data preparation code

```
from libreco.data import random_split, DatasetPure
from libreco.evaluation import evaluate

# Load your dataset ("clicks.csv" is file containing data)
data = pd.read_csv("clicks.csv")

# LibRecommender module requires "user", "item", and "label"
    columns in input
data.rename(columns={"user_id_hash": "user", "target_id_hash":
    "item", "is_click": "label"}, inplace=True)

# LibRecommender module requires "user" and "item" to be first
    two columns of input
columns = ["user", "item"]
new_col_order = columns + [col for col in data.columns if col
    not in columns]
data = data[new_col_order]

# Split data into training, evaluation, and testing sets
train_data, eval_data, test_data = random_split(data,
    multi_ratios=[0.8, 0.1, 0.1])

# Build training set
train_data, data_info = DatasetPure.build_trainset(train_data)
eval_data = DatasetPure.build_evalset(eval_data)
```

As shown in Listing 1, we had to manipulate our original input data such that it fit the LibRecommender module's input requirement. After training the model, we also had to write our own prediction mechanism using the model since we wanted to binary predictions for is_click, as shown in Listing 2.

Listing 2: LibRecommender model evaluation code

```
users = test_data["user"]
items = test_data["item"]
predictions = []
for user, item in zip(users, items):
    prediction = svd.predict(user, item) # Replace svd with
        model of interest
    predictions.append(prediction)

threshold = .5 # Adjust the threshold as needed

binary_preds = [1 if prediction >= threshold else 0 for
   prediction in predictions]
target = test_data["label"]
correct_predictions = 0
correct_predictions += (binary_preds == target).sum().item()
print(correct_predictions/len(target))
```

5 Results and Discussion

Baseline Model	Accuracy
Logistic Regression	0.5
Decision Tree	0.66
Random Forest	0.73
Perceptron	0.68

Table 1: Baseline Model Accuracy

Since we wanted to compare our deep learning models with traditional machine learning techniques, we began by running some simple baseline models: logistic regression, decision tree, random forest, and perceptron. As seen in Table 1, logistic regression performed the worst out of the baseline models since it is a linear classifier, and for high-dimensional feature spaces like our dataset, logistic regression may struggle to capture the intricate patterns in the data, leading to poorer performance compared to other models. Random forest performed the best out of the baseline models, probably since we were able to makes branches on individual features. Additionally, random forest is simply a collection of decision trees, so as we can see, it outperformed the decision tree model.

Model Architecture	Accuracy
SVD	0.719
SVD++	0.726
WideDeep	0.731
Neural CF	0.734
DeepFM	0.735
Transformer	0.741

Table 2: LibRecommender Model Accuracy

We tried multiple models from the LibRecommender module, including SVD, SVD++, WideDeep, Neural CF, DeepFM, and Transformer. For each of these models, we also trained on various hyperparameters. The models listed here are the best-performing ones. As we can see from Table 2, SVD performed the worst, only achieving an accuracy of 0.719 while Transformer achieved an accuracy of 0.741. This could be due to the fact that they excel at capturing long-range dependencies and modeling complex sequential data, and in the context of recommendation systems, user-item interactions can be viewed as sequences of events or behaviors over time.

Even the best LibRecommender models did not outperform random forest by much for our task, which shows that in certain situations, simpler models can achieve comparable results to deep learning. This is an important implication for startups, nonprofits, or other organizations that may not have access to as much compute power since huge datasets require a lot of computational power and GPUs may be expensive.

As mentioned earlier in the paper, we only selected data points from users that had at least 8 clicks in the original Taboola dataset in an attempt to alleviate the data sparsity issue. Even then, collaborative filtering was a weak choice for predicting clicks. Another limitation of our approach is that we used open-source models. Perhaps creating a custom neural collaborative filtering model specifically tailored to our task would have yielded better results than open-source models. Initially, we tried creating a our own model that first learns user and item embeddings using collaborative filtering and then incorporates user, item, and context features in a neural network, but unfortunately, the model took too long to run, even on our subsampled dataset containing data points of users that had at least 8 clicks.

6 Conclusion

In this project, we delved into the intricate realm of CTR prediction, a cornerstone in optimizing the economic engine of digital advertising. Our exploratory journey through various model architectures, from SVD to cutting-edge neural approaches like Neural CF, DeepFM, and Transformers, showcased a spectrum of capabilities in capturing the nuanced patterns of user-item interactions. The data from Taboola, rich yet challenging due to its sparsity, presented an opportunity to innovate beyond conventional methods.

The results were telling: neural models like Transformers nudged ahead, achieving a commendable 74.1% accuracy, underscoring the sophisticated potential of deep learning in deciphering the CTR conundrum. However, the distinction in performance was not stark compared to ensemble techniques like Random Forest, a finding that democratizes the applicability of CTR prediction across organizations with varying computational prowess.

Throughout our project, we encountered significant challenges, particularly in managing the computational demands of processing a massive dataset with 16 million users. The inherent sparsity of the data posed another obstacle, making it difficult to generate accurate recommendations. We tackled these issues head-on by strategically subsetting the data to focus on active users, thereby enhancing the relevance and manageability of our analyses. Furthermore, we overcame computational limitations by optimizing our code, utilizing efficient libraries, and adopting models that balanced accuracy with resource constraints. These pragmatic solutions enabled us to not only achieve but also surpass our benchmarks, demonstrating the feasibility of high-performance recommender systems in resource-varied environments.

Our work stands as a testament to the potential of advanced models in a real-world context, but it also highlights the pragmatic reality of balancing resource investment and outcome. As we gaze ahead, the implications of our research extend to smaller entities and startups, demonstrating that with creativity and strategic model choice, one can remain competitive in the analytics arena. This project not only contributes to the academic and professional discourse on CTR prediction but also paves the way for future explorations in model efficiency and broader accessibility.

References

[1] Zhou, G., Mou, N., Fan, Y., Pi, Q., Bian, W., Zhou, C., Zhu, X., & Gai, K. (2019). Deep interest evolution network for click-through rate prediction. In Proceedings of the AAAI conference on artificial intelligence (AAAI-19). 33(01), 5941–5948. https://doi.org/10.1609/aaai.v33i01.33015941.

[2] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 1119–1129. https://doi.org/10.1145/3308558.3313497

[3] Zekun Li et al. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (Nov. 2019). 10.1145/3357384.3357951. http://dx.doi.org/10.1145/3357384.3357951.

[4] Guibing Guo, Jie Zhang, Zhu Sun and Neil Yorke-Smith, LibRec: A Java Library for Recommender Systems, in Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modelling, Adaptation and Personalization (UMAP), 2015.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30–37, 2009.

[6] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 426–434). https://dl.acm.org/doi/10.1145/1401890.1401944.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems. 7–10.

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In Proceedings of the the International Joint Conference on Artificial Intelligence (IJCAI). https://dl.acm.org/doi/10.5555/3172077.3172127.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web (WWW), 2017. https://dl.acm.org/doi/10.1145/3038912.3052569.

[10] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation. In Fifteenth ACM Conference on Recommender Systems (RecSys '21), 2021. https://doi.org/10.1145/3460231.3474255.