Towards Long-term Fairness in Recommendation

Phase 2 Improvements in Project

Devika Singh, Rinshi Kumari, Varsha
210101036, 210101109, 210108040

Indian Institute of Guwahati

Contents

Abstract I

We proposed and implemented the following improvements on the code from paper - "Towards Long-term Fairness in Recommendation"

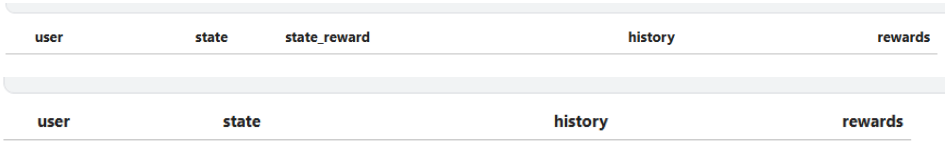
1. Application Analysis
2. Individual Fairness
3. Enhance Model Interpretability

Application Analysis - Overview I

- Analyzing how well the framework from the paper adapts to different recommendation systems is crucial, given its focus on movie recommendations.
- In e-commerce, where product diversity and seller visibility are key concerns, the method's dynamic fairness approach would need to address the challenge of balancing fairness across rapidly changing inventory and seller performance.
- Evaluating the method in these domains would help determine its effectiveness in maintaining fairness in diverse, dynamic environments.

Input Files - Movie Recommendation

- Input size - 100k
- Number of users - 943
- Number of items - 1682
- Vector embedding size - 100
- Input files:
  - pmf_item_embed.npy - shape(1682,100)
  - pmf_user_embed.npy - shape(943,100)
  - item_cost_indicator_28.npy - shape(1682)
  - train_data.csv
  - test_data.csv



- Training data columns

- Testing data columns

Input Files - E commerce I

- Input size - 24999
- Number of users - 19774
- Number of items - 15
- Vector embedding size - 10 (need to be less than 15)
- In data product category is given (eg. car). We consider each category as an item.
- Columns in original e-commerce dataset(2) and their renaming
  - customer_id (np.int32) - userId
  - product_category (np.int32) - itemId [minus 501 from each in data , corrected during output]
  - time_on_site [Minutes] (np.float64) - time_on_site
  - clicks_in_site (np.float64) - clicks_in_site
  - date (datetime, day is first in format) - timestamp

Input Files - E commerce II

- Rating column
  - rating column - added as clicks_in_site / time_on_site where both are non-null, otherwise it is set to null.
  - We are using **implicit rating** here.
  - Since the clicks on a category of items should increase if user spends more time on the website, we normalize the clicks by the time on site.
- Generated Input files:
  - pmf_item_embed.npy - shape(1682,100)
  - pmf_user_embed.npy - shape(943,100)
  - item_cost_indicator_28.npy - shape(1682)
  - train_data.csv -
  - test_data.csv
- Click here to visit the Kaggle for Data Preprocessing
- Click here to visit the Kaggle for Running of code

Individual Fairness - Overview I

- To further generalize the framework, we incorporate individual fairness along- side the group fairness focused on in the paper.
- Individual fairness ensures that users or items with similar characteristics are treated similarly, regardless of their protected attributes, addressing a critical gap in current models.
- By extending the dynamic fairness learning approach already used to individual fairness, this refinement will allow the system to mitigate bias at a more personalized level, adapting to user-specific needs over time.
- This ensures a fairer recommendation process for both long-term and individual dimensions.

User Constraint

```python
# Calculate the total interaction frequency for each user (sum of ratings per user)
user_interaction = dg.data.groupby('userId')['rating'].sum()

# Define the threshold for the bottom 20% to identify low-interaction users
sensitivity_threshold = np.percentile(user_interaction, 20)

# Create user_indicator where 1 represents sensitive (low interaction) users and 0 otherwise
user_indicator = (user_interaction <= sensitivity_threshold).astype(int)

# Map the user_indicator back to the main DataFrame

print("User indicator based on low interaction frequency has been added to the data.")

np.save('/kaggle/working/item_user_indicator_28_e_commerce.npy', user_indicator)
#print(cost_indicator_array.shape)
print(user_indicator)
```

Enhance Model Interpretability - Overview I

- Introducing techniques to enhance model interpretability, such as LIME (Local Interpretable Model-agnostic Explanations), would be highly applicable in the context of long-term fairness in recommendations.
- LIME can help identify how specific features influence the recommendations, which is crucial for uncovering biases in the model.
- By examining feature importance, we can see how attributes like item popularity, user demographics, or other characteristics contribute to the recommendation process.
- This would allow addressing of any biases, making the model not only fairer but also more transparent.
- Applying such interpretability techniques would further refine fairness interventions and ensure they align with the model's goals.

Updating the code to work with newer versions of libraries - Overview I

- We have updated teh code to work with newer numpy and tensorflow libraries.
- We have cut down on the originally large codebase in the paper, to extract the parts that are being used. This is important because there are reduntant versions of the classes in the paper's github codebase which are not used during the training.
- Applying such techniques has increased the readbility of the code for FCPO.

Conclusion I

- The model works well for e-commerce dataset. Comparable results to other models on kaggle for this dataset(2)(good rating ones).
- Using user constraints enhances the model accuracy in prediction.
- Enhancing model interpretability using LIME has been studied by the team members.
- Instead of Enhancing model interpretability we originally proposed we have instead put our efforts in updating the codebase to newer libraries, and cleaning it for easier readability and removing redundancy.

References I

[1] M. Wen, O. Bastani, and U. Topcu. "Fairness with Dynamics." *ArXiv* abs/1901.08568 (2019).

[2] Original E-commerce dataset. Accessed from Kaggle.