جامعة جدة
University of Jeddah

CCAI422 Recommender System Course Project

Phase 2: Project Code

Supervisor Dr.Safa Alsefri, Dr.Latifa Al-jiffry
by Bedoor Ayad, Raneem Alomari, Deema Al-saygh

CCAI422 RECOMMENDER SYSTEM COURSE PROJECT

PHASE 2: PROJECT CODE

CODE ERRORS:
We examine and rectify various code errors, including syntax problems, logical issues, and runtime exceptions. We ensure that the code executes smoothly without encountering any errors.

Data Preprocessing: we have implemented these Preprocessing steps

- **Binarizing the ratings**: There are some database of ratings ranging from 0.5 to 5. in these scenarios Binarizing the ratings can be useful especially when dealing with recommendation systems so we have add method to checks if the 'Rating' column contains float values then binariz it else If the original ratings are not float, it prints a message indicating that there's no need for binarization.

- **Handeling imbalanced data using Undersampling techniques** : we found that the distribution is not perfectly balanced, but it's common to have some class imbalance in real-world datasets. In Our case, Ratings 4 and 3 have more instances compared to others. Knowing that handling class imbalance is crucial in machine learning, especially when certain classes have significantly fewer instances than others. We considered Undersampling techniques : Decrease the number of instances in the majority class by randomly removing examples. That result in 721356 instances for each class.

- **Filter triples (data, user, item)**: this step we didnt add it is included in the orginal code to preprocess and filter the input data based on user and item counts, potentially improving the quality of the data for further analysis or modeling. It allows you to focus on a subset of users and items that meet specified interaction thresholds. Considering items that were rated by at least 5 users and users who rated at least 5 movies.

DOCUMENTATION:
We've written a comment at each function to improve the readability of the code by explaining complex logic or algorithms in more understandable language, and to help other developers (including your future self) understand the code's functionality.

CCAI422 RECOMMENDER SYSTEM COURSE PROJECT
PHASE 2: PROJECT CODE

Hyperparameters Tuning

The Multi DAE & VAE architecture

Input Layer 1:
Weight shape: (219, 600)
Bias shape: (600,)
Layer 2:
Weight shape: (600, 200)
Bias shape: (200,)
Layer 3:
Weight shape: (200, 600)
Bias shape: (600,)
Layer 4:
Weight shape: (600, 219)
Bias shape: (219,)

Training Multi-DAE Denoising Autoencoder (DAE) & Multi-Modal Variational Autoencoder (VAE)

The hyperparameters for both DAE & VAE are:

p_dims = decoder dimension
q_dims = encoder dimension,
lam = regularization parameter = 0.01
lr = learning rate = 1e-3
training batch_size = 500
validation batch_size_vad = 2000
training n_epochs = 200
total_anneal_steps = 200000, Total gradient updates
anneal_cap = 0.2, maximum annealing value

1. Total Anneal Steps (total_anneal_steps):
   - If your optimization process tends to converge quickly, you might not need a large number of annealing steps.
   - On the other hand, if convergence is slow or you have a large and complex model, you may want to increase this value.
2. Annealing Parameter Cap (anneal_cap):
   - The cap should be set based on the expected range for the annealing parameter. If it's too small, the annealing process might not have a significant effect.
   - If it's too large, the annealing parameter may grow unchecked and potentially lead to instability or divergence in the optimization process.

dimensions for the neural network layers (p_dims)

- 200 units/neurons and represents the latent factors
- Next 600 units/neurons
- Finally layer represents the number of items in the dataset

dimensions for the neural network layers (q_dims)

- First layer represents the number of items in the dataset
- Next 600 units/neurons
- 200 units/neurons and represents the latent factors

CCAI422 RECOMMENDER SYSTEM COURSE PROJECT

PHASE 2: PROJECT CODE

SELECTION OF EVALUATION METRIC:

we have choosen these evaluation metrix for our recommender system
1. Calculating spread metric is calculated by counting the total number of ratings contained in the ratings matrix it range of sparsity matrix values is from 0% to 100%.

2. Discounted Cumulative Gain (DCG) is a metric commonly used to evaluate the ranking quality of a model. NDCG@k specifically refers to the NDCG calculated at a particular position, typically at position k.

- NDCG close to 1: This suggests that the system is doing an excellent job in ranking relevant items at the top positions. The higher the mean NDCG, the better the ranking quality.
- NDCG close to 0: This indicates poor performance, where relevant items are not well-ranked, and the system may need improvement.

3. Recall at k batch : has a range of [0, 1]. It measures the proportion of relevant items that were successfully retrieved in the top-k recommendations.

OVERFITTING AND UNDERFITTING:

During the implementation, we encountered the issue of underfitting. Underfitting occurs when the model fails to capture the underlying patterns and relationships in the data, resulting in poor performance and inaccurate predictions. To address this issue, we took several steps to improve the model's performance and mitigate underfitting.

1. **Undersampling techniques**: We noticed that our dataset had imbalanced class distributions, with some ratings having significantly fewer instances than others. This class imbalance can lead to biased predictions and affect the model's ability to learn. To mitigate this issue, we employed undersampling techniques. We randomly removed examples from the majority class to balance the class distribution and provide a more representative training set. This helped prevent the model from being biased towards the majority class and improved its overall performance.
2. **Regularization Techniques**: We introduced regularization techniques, such as L1 and L2 regularization, to improve the model's generalization. Regularization adds a penalty term to the loss function, discouraging the model from assigning excessive importance to certain features or weights. This helps reduce the impact of noisy or irrelevant information and improves the model's ability to generalize.
3. **Adjusting the Number of Epochs**: The number of epochs used during model training was adjusted. The number of epochs determines how many times the model iterates over the entire dataset during training. Changing the number of epochs allows control over the model's learning process.

CCAI422 RECOMMENDER SYSTEM COURSE PROJECT

PHASE 2: PROJECT CODE

IDENTIFIED CHALLENGES AND PROPOSED IMPROVEMENTS:

The paper "Variational Autoencoders for Collaborative Filtering" by Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara introduces variational autoencoders (VAEs) for collaborative filtering.

Shortcomings of the existing methodology:

1. Cold-start problem: Collaborative filtering struggles with insufficient data on new users or items. VAE-based approaches also face this challenge and may not accurately recommend for new users or items.
2. Lack of uncertainty estimation: Existing VAE-based models do not provide uncertainty measures in their predictions. Uncertainty estimation is important in recommendation systems to help users understand recommendation reliability.
3. Difficulty in modeling long-tail items: Collaborative filtering struggles with recommending infrequently rated or low-interaction items. VAE-based approaches may not effectively capture preferences and characteristics of such items.

To improve the methodology, the authors propose the following enhancements:

1. Hybrid recommender system: Combine VAE-based collaborative filtering with other techniques like content-based filtering or knowledge-based systems to address the cold-start problem and provide accurate recommendations for new and existing users/items.
2. Bayesian extensions: Incorporate Bayesian techniques into the VAE framework to estimate uncertainty in recommendations, giving users more reliable and interpretable results.
3. Incorporating side information: Include additional data like item attributes or user demographics in the VAE-based model to better capture the unique characteristics of long-tail items, even with limited interaction data.

TEAM CONTRIBUTIONS:

Name	ID	Work Tasks
Bedoor Ayad	2005961	Code Errors,Overfitting and Underfitting Identified Challenges and Proposed Improvements
Raneem Alomari	2006352	Code Errors, Data Preprocessing Hyperparameters Tuning, Evaluation Metric
Deema Al-sayegh	2006085	Data Preprocessing Documentation