School of Computing

CS4242 Social Media Computing

AY23/24 Semester 2

Project Report

Chen Hsiao Ting
A0222182R

Table of Contents

1. Introduction

1.1. Overview

Recommendation systems are pivotal in shaping user experience on digital platforms, especially in social media and micro-video environments. They are tasked with the complex challenge of sifting through extensive databases to deliver personalized content, thereby enhancing user engagement and platform loyalty.

1.2. Problem Statement

The dynamic and vast nature of micro-video content presents unique challenges for recommendation systems, demanding a balance between accuracy in user preferences and the introduction of diverse content. Achieving this balance is crucial for user retention and satisfaction in an increasingly competitive digital landscape.

1.3. Objective

This project aims to refine recommendation strategies by enhancing the existing collaborative filtering (CF) based matrix factorization algorithm with additional visual and category features. The goal is to not only enhance the accuracy of content recommendations but also to promote the discovery of diverse content, thereby enriching the user's experience and engagement with the platform.

Additionally, it seeks to evaluate the effects of model selection criteria, learning rates, and dropout rates on the overall performance of recommendation systems in terms of accuracy and diversity.

2. Algorithm Exploration & Implementation

2.1. Visual & Category Features Data Preprocessing

In this project, visual and category feature data preprocessing involves loading the respective feature sets from pre-saved numpy files. The visual features are stored as dictionaries where keys are item IDs and values are 512-dimensional feature vectors. Similarly, category features, which classify items into predefined categories based on content characteristics, are also loaded as dictionaries with item IDs as keys and anonymized category integers as values. Refer to Appendix A for the actual implementation to load the visual and category features.

2.2. Content-Based Matrix Factorization (CBMF) Model

The Content-Based Matrix Factorization (CBMF) model extends traditional collaborative filtering by incorporating visual and category features into the recommendation process. It is designed to improve recommendation accuracy and diversity by utilizing additional content information. Refer to Appendix B for actual implementation of the CBMF model.

2.3. Accuracy Metrics

There are a total of two different accuracy metrics, namely recall and Normalized Discounted Cumulative (NDCG), implemented in this project. The following sections discuss each of the metric in details. The implementation of both accuracy metrics are already provided in the original code.

2.3.1. Recall

Recall is a crucial accuracy metric in recommendation systems, focusing on the **fraction of relevant items correctly identified in the top recommendations.** It is fundamental for understanding how well a model can identify items that users are genuinely interested in, thereby enhancing user satisfaction.

2.3.2. Normalized Discounted Cumulative (NDCG)

Normalized Discounted Cumulative (NDCG) complements recall by evaluating the quality of the **ranked recommendation list**, emphasizing the placement of relevant items. A higher NDCG indicates that the model not only identifies relevant items but also ranks them higher, mirroring the ideal user experience more closely.

2.4. Diversity Metrics

There are a total of four different diversity metrics, namely Intra-List Diversity (ILD), Intra-List Diversity with Visual and Category features (ILD-VC), item coverage and category coverage, implemented in this project. The following sections discuss each of the metric in details.

2.4.1. Intra-List Diversity (ILD)

Intra-List Diversity (ILD) measures diversity within the **top-10 recommended items by counting unique pairs of items that differ in category.** The normalized score indicates how varied the recommended items are, with a score of 1 representing maximum diversity. Refer to Appendix C for actual implementation of the ILD metric.

2.4.2. Intra-List Diversity with Visual and Category features (ILD-VC)

Intra-List Diversity with Visual and Category features (ILD-VC) **extends ILD by incorporating both visual and categorical differences** between items in a list. It uses a weighted average of these dissimilarities to provide a broader assessment of diversity, reflecting variations in both visual content and categories. Refer to Appendix D for actual implementation of the ILD-VC metric.

2.4.2. Item Coverage

Item coverage is a broad metric that reflects the system's ability to recommend a wide array of items, minimizing the over-concentration on popular items. High item coverage indicates a system's **effectiveness in uncovering and recommending "long-tail" items**, which might be highly relevant but less popular. Refer to Appendix E for actual implementation of the item coverage metric.

2.4.3. Category Coverage

Category coverage extends the concept of diversity to the categorical level, **ensuring that the recommended items span across various categories**. This metric is significant for content discovery, allowing users to explore content from categories they might not have encountered otherwise, thereby enriching their experience. Refer to Appendix F for actual implementation of the category coverage metric.

2.5. F1 Score

In this project, the F1 score is a **harmonic mean that combines NDCG and ILD to evaluate the model's effectiveness**, balancing the trade-off between accurately recommending relevant items (NDCG) and ensuring a diverse range of categories in the recommendations (ILD).

2.5.1. Definition

The formulae for F1 measurement (NDCG-ILD) is defined as:

$$F_1 = \frac{2 \times NDCG \times ILD}{NDCG + ILD}$$

where ILD (Diversity) is defined as:

$$ILD = \frac{2}{K(K-1)} \Sigma\Sigma \mathbb{I}(category_i \neq category_j)$$

Refer to Appendix G for actual implementation of the F1 score metric.

2.5.2. Justification

In this project, we choose NDCG and ILD over other metrics like recall, item coverage, category coverage and ILD-VC for the F1 score for the sake of simplicity. NDCG is a direct indicator of a model's accuracy in predicting user preferences, focusing on the ranked relevance of recommendations. ILD, on the other hand, represents the diversity aspect by measuring the spread of different content categories within the recommendations.

2.5.3. Accuracy-Diversity Trade-Off

The accuracy-diversity trade-off is a fundamental challenge in recommendation systems, where increasing one often leads to a decrease in the other. High accuracy typically means that the system recommends items very similar to the user's past preferences, which can reduce diversity. Conversely, aiming for high diversity might introduce less relevant items, reducing accuracy.

The F1 score, by incorporating both NDCG and ILD, offers a balanced approach to evaluate models on their ability to maintain this balance. It encourages the development of recommendation algorithms that do not compromise relevance for the sake of diversity or vice versa, striving for a recommendation list that is both accurate and varied.

This balance is crucial for enhancing user satisfaction and engagement, as it caters to their specific interests while simultaneously broadening their content horizons.

2.6. Model Training

Several improvements such as early stopping and tensorboard writer were added to the code base to better optimise the model training process and log the data. The following sections will discuss the improvements in details.

2.6.1. Early Stopping

Early stopping is employed as a pragmatic approach to mitigate overfitting and enhance computational efficiency. By monitoring the recall score on the validation set across training epochs, the training process is terminated if there is no improvement in the recall score for a predefined number of consecutive epochs.

This strategy ensures that the model is saved at the point where it achieves the best balance between accuracy and diversity, preventing the model from learning noise or irrelevant patterns from the training data, which could degrade its performance on unseen data. Refer to Appendix H for actual implementation of the early stopping.

2.6.2. Tensorboard Writer

TensorBoard Writer is utilized in this project as a powerful visualization tool to monitor the training and evaluation metrics in real time. By logging key performance indicators such as recall, NDCG, ILD, ILD-VC, item coverage, category coverage, and F1 score at each epoch, TensorBoard provides an interactive interface to observe trends, compare model variants, and diagnose training behaviors.

This facilitates a deeper understanding of how different parameters and model architectures affect the recommendation system's performance, guiding the selection of optimal configurations and highlighting areas for further improvement. Refer to Appendix I for actual implementation of the tensorboard writer.

3. Experimental Results & Analysis

3.1. Selection of Best Model (F1 vs Recall)

The original recommendation system provided takes the best model from the best recall at 10. However, given that we now have additional diversity metrics on top of the original accuracy metrics, we want to find out which selection of the best model would be more suitable in the context of this project.

3.1.1. Experimental Results

In this experiment, we keep the content weight at 0.1, learning rate at 0.001 and dropout rate at 0. The only changing variable here is the selection of the best model by highest F1 score, recall or NDCG. Refer to Appendix J (Figure 1) for experimental results.

3.1.2. Observations

From Appendix J, we can observe that the selection of model based on highest F1 score and highest NDCG have highly similar results in both accuracy and diversity. Whereas selection of model based on highest recall gives a higher accuracy and diversity for both MF and CBMF compared to the other two models.

In addition, observe that the F1 score in CBMF is above that in MF for all 3 models. Looking closer, this difference mostly comes from the accuracy metric. As there is a significant increase of accuracy, and a slight decrease of diversity from MF to CBMF.

3.1.3. Analysis

From the experimental results, we can conclude that **CBMF outperforms MF in F1 score, which is mainly contributed by the accuracy metrics.** And selection of model based on recall score gives a higher accuracy and similar levels of diversity results compared to the other two selections of model. As a result, in the following experiments, I will take the best model from the best recall score.

3.2. Varying Content Weight

In a CBMF model, content weight is a parameter that determines the relative influence of visual and category features on the overall recommendation score, where a value of 0 represents exclusive reliance on user-item interactions and a value of 1 signifies exclusive dependence on content features. In this section, we want to explore how content weight affects F1 score for MF and CBMF.

3.2.1.1. Experimental Results

Taking best model from the best recall score, with varying content weight from 0.0 to 1.0, 0.001 learning rate and 0 dropout rate, the following Figure 2 shows the summary results for CBMF model.

Weight	F1	Recall	NDCG	ILD	ILD-VC	Item Cov	Cat Cov
0.0	0.0332	0.0246	0.0169	0.9949	0.9619	0.0508	0.1957
0.1	**0.0514**	**0.0468**	**0.0264**	**0.9801**	**0.9537**	**0.5818**	**0.8940**
0.2	0.0270	0.0296	0.0137	0.9777	0.9529	0.5520	0.8750
0.3	0.0353	0.0394	0.018	0.9741	0.9480	0.3626	0.7500
0.4	0.0463	0.0493	0.0237	0.9775	0.9599	0.3495	0.7799
0.5	0.0438	0.0468	0.0224	0.9610	0.9384	0.6571	0.9049
0.6	0.0424	0.0443	0.0217	0.9646	0.9524	0.4988	0.8995
0.7	0.0388	0.0394	0.0198	0.9190	0.9032	0.7611	0.9049
0.8	0.0395	0.0419	0.0202	0.8707	0.8710	0.7599	0.9239
0.9	0.0268	0.032	0.0136	0.8627	0.8733	0.6464	0.8967
1.0	0.0206	0.0246	0.0104	0.8673	0.8845	0.5878	0.8614

Figure 2: Summary table showing Impact of varying content weight on CBMF model.

3.2.1.2. Observations

Analyzing the CBMF model with varying content weights from 0.0 to 1.0 indicates that a moderate content weight, specifically around 0.1, yields the best performance across metrics like recall (0.0468), NDCG (0.0264), and F1 score (0.0514). As the content weight increases

beyond this point, accuracy and diversity generally tend to decrease, although the item coverage and category coverage sometimes increase slightly.

3.2.1.3. Analysis

Based on the experimental results, we can conclude that **a balanced approach, with 0.1 content weight, to integrating content-based features with collaborative filtering elements is most effective for enhancing recommendation performance.** This balance allows the model to leverage the strengths of both content-related information and historical user-item interactions, which not only improves the accuracy of recommendations but also maintains a good level of diversity.

This performance trend likely stems from enhancing recommendation relevance without overshadowing the collaborative aspect. Excessive content weighting can lead to overfitting on content features, diminishing the model's overall effectiveness. A balanced approach ensures the model leverages both content and collaborative data, promoting accurate and diverse recommendations.

As a result, in our subsequent experiments, we will be fixing content weight at 0.1.

3.3. Accuracy Metrics (Recall, NDCG)

3.3.1. MF vs CBMF on Accuracy

The accuracy metrics defined in this project include recall and NDCG. In this section, we want to explore how MF and CBMF perform differently in recall and NDCG.

3.3.1.1. Experimental Results

Taking best model from the best recall score, with 0.1 content weight, 0.001 learning rate and 0 dropout rate, Appendix J (Figure 3 and 4) shows the recall and NDCG graphs for MF and CBMF.

3.3.1.2. Observations

Observe from Appendix J that the CBMF curves are largely above the MF curves in both recall and NDCG. The peak points of recall and NDCG in CBMF are also higher than that in MF.

3.3.1.3. Analysis

Based on the experimental results, we can conclude that **CBMF outperforms MF in accuracy, since CBMF has better recall and NDCG than MF in general.**

CBMF incorporates additional information, such as visual features and category features into the recommendation process, which can provide more context for predictions compared to MF, which primarily relies on user-item interaction data. This additional context can help CBMF make more accurate predictions, especially in cases where the interaction data is sparse.

3.3.2. Impact of Learning Rate on Accuracy

In machine learning, the learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function. In this section, we want to explore the impact of learning rate on the convergence speed, and the peak values of recall and NDCG.

3.3.2.1. Experimental Results

We take the best model from the best recall score, keep the content weight at 0.1 and dropout rate at 0. The only changing variable here is the learning rate, which is set to 0.0005, 0.001, and 0.01. Refer to Appendix J (Figure 5) shows the summary of the experimental results.

3.3.2.2. Observations

Observe that CBMF model with a 0.001 learning rate achieved the highest recall and NDCG values at the earliest epoch (4), suggesting it converges faster and more effectively at this rate. In contrast, the lowest (0.0005) and highest (0.01) learning rates resulted in either slower convergence or lower peak values, demonstrating less efficiency.

3.3.2.3. Analysis

Based on the observations, we can conclude that a **moderate learning rate of 0.001 consistently results in quicker convergence and superior performance in terms of recall and NDCG for both CBMF and MF models.** This suggests that it strikes an optimal balance between rapid learning and the ability to fine-tune to nuanced data patterns, avoiding the extremes of underfitting and overshooting which occur at lower and higher learning rates, respectively.

The reason a 0.001 learning rate performs optimally is likely due to its balance between exploration and exploitation during the training process. Lower learning rates (0.0005) might be too conservative, causing the model to adjust too slowly, thus taking longer to converge and failing to capture more complex patterns within the available training epochs. Conversely, a higher rate (0.01) can cause the model to adjust too rapidly, potentially leading to instability in learning and missing the optimal convergence points by overshooting them.

As a result, in our subsequent experiments, we will be fixing learning rate at 0.001.

3.3.3. Impact of Dropout Rate on Accuracy

Dropout rate is the probability of randomly setting a neuron's output to zero during training, which helps prevent overfitting by encouraging the model to learn more robust features.

With effective dropout, the gap between validation and test performance should be smaller, indicating that the model generalizes better and is less overfitted to the validation set. In addition, with a suitable dropout rate, the accuracy metrics should plateau or continue to improve slowly, showing more robust learning.

In this section, we will observe the effect of dropout rate, especially in preventing overfitting and its impact on recall and NDCG.

3.3.3.1. Experimental Results

As mentioned in the previous section, we will keep the content weight at 0.1 and learning rate at 0.001, since it performs better in terms of both converging speed and accuracy. In this section, we will experiment with 0 and 0.2 dropout rates. Refer to Appendix J (Figure 6) for the summary of the experimental results.

3.3.3.2. Observations

From the data, the absence of dropout (0 rate) results in higher peak values for recall and NDCG, particularly in the CBMF model. At a 0 dropout rate, CBMF achieves a recall of 0.0542 and an NDCG of 0.0256, both at epoch 4, significantly outperforming the MF model. However, with a 0.2 dropout rate, while the MF model shows a slight improvement in NDCG, both models exhibit reduced recall and NDCG compared to the 0 dropout case.

3.3.3.3. Analysis

The higher performance metrics in the absence of dropout suggest that, for this dataset and under these specific experimental conditions, **dropout may be inhibiting the models' ability to optimally fit to the data**, possibly by excessively penalizing their capacity to learn intricate patterns in the data.

This observed behavior might stem from the nature of dropout as a regularization technique designed to prevent overfitting by randomly deactivating a subset of neurons during training. While effective in complex networks prone to overfitting, in contexts where the data does not overly compel models to memorize rather than generalize, such as possibly with this dataset, dropout could unnecessarily limit the learning process. Consequently, this might lead to underfitting, where the models fail to capture essential data patterns, reflected in the lower recall and NDCG scores when dropout is applied.

As such, in the subsequent experiment, we will adopt 0 dropout rate.

3.4. Diversity Metrics (ILD, ILD-VC, Item Coverage, Category Coverage)

3.4.1. MF vs CBMF on Diversity

There are a few diversity metrics defined in this project, namely Intra-List Diversity (ILD), Intra-List Diversity with Visual and Category features (ILD-VC), item coverage, and category coverage, which are influenced by how diverse the recommendations are within the list and across the entire item space.

In this section, we will explore how MF and CBMF perform differently in ILD, ILD-VC, item coverage, and category coverage.

3.4.1.1. Experimental Results

Taking best model from the best recall score, with 0.1 content weight, 0.001 learning rate and 0 dropout rate, the figures (Figures 7, 8, 9 and 10) in Appendix J are the ILD, ILD-VC, item coverage, and category coverage graphs for MF and CBMF.

3.4.1.2. Observations

In ILD and ILD-VC, both CBMF and MF models demonstrate high scores, consistently above 0.95, throughout the epochs. However, MF shows a slightly better result compared to CBMF, though the difference converges to close to 0.01 as the epoch increases. Whereas in item coverage and category coverage, CBMF consistently achieves higher coverage than MF across all epochs.

3.4.1.3. Analysis

The observed data indicates that the **diversity levels of CBMF model remainly closely similar to that of MF model**, suggesting that incorporating visual and category features into the recommendation algorithm maintains the range of content recommended to users at a high level. This is crucial for platforms aiming to enhance content discovery and maintain user engagement by exposing them to a diverse array of media.

3.5. F1 Score

3.5.1. MF vs CBMF on F1 Score

In this section, we want to explore how MF and CBMF perform differently in F1 score.

3.5.1.1. Experimental Results

Taking best model from best Recall, with 0.1 content weight, 0.001 learning rate and 0 dropout rate, Appendix J (Figure 11) shows the F1 scores for MF and CBMF.

3.5.1.2. Observations

Observe from Figure 11 that the CBMF curves are largely above the MF curve in F1 score. The peak F1 score in CBMF is also higher than that in MF.

3.5.1.3. Analysis

Based on the experimental results, we can conclude that CBMF outperforms MF in F1 score, since CBMF has better F1 score than MF in general. Combining our knowledge from the previous sections, we can conclude that **CBMF outperforms MF in terms of accuracy, while maintaining similar high levels of diversity.**

CBMF may outperform FM in accuracy while maintaining similar levels of diversity due to their ability to leverage additional item-specific features such as visual and category features. This enrichment allows CBMF to discern more nuanced preferences and patterns within user-item interactions, leading to more accurate predictions of user interests.

By effectively incorporating these features, CBMF can enhance the precision of recommendations without necessarily sacrificing diversity, as the model can still recommend a wide range of items by capturing a broader spectrum of user preferences.

4. Limitations

While the approach of selecting the best model based on the highest Recall, using a content weight of 0.1, learning rate of 0.001 and dropout rate of 0 has shown promising results, it is not without its limitations.

Firstly, relying predominantly on F1 scores derived from NDCG and ILD may overlook other crucial aspects of recommendation systems like user satisfaction, serendipity, or novelty. Future research could aim to develop a more comprehensive evaluation framework that encompasses a broader range of metrics to capture the multifaceted nature of user experience.

In addition, the chosen content weight, learning rate and dropout rate were fixed throughout the training process. Adaptive approaches that adjust these parameters dynamically based on the model's performance during training could potentially lead to better convergence and performance.

5. Conclusion

In conclusion, this project successfully demonstrated the potential of augmenting collaborative filtering recommendation systems with visual and category features to enhance micro-video recommendations. By implementing and comparing the CBMF with MF, it was observed that **CBMF not only improved the accuracy metrics, such as recall and NDCG, but also managed to maintain comparable levels of diversity**, evidenced by similar ILD, ILD-VC, item coverage, and category coverage metrics across both models.

Through evaluating models based on an F1 score that harmonizes NDCG and ILD, this project has underscored the importance of balancing accuracy with diversity in recommendations. The selection of optimal parameters, such as a content weight of 0.1 and learning rate of 0.001, further contributed to the effectiveness of the recommendation models.

However, the study's approach, while effective, has its constraints, notably in the evaluation metrics and parameter settings. The exclusive use of F1 scores focusing on NDCG and ILD might not fully capture the multifaceted nature of user satisfaction in recommendation systems. Moreover, the fixed content weight, learning and dropout rates throughout training overlook the potential benefits of adaptive adjustments. Addressing these limitations through more comprehensive metrics and dynamic parameter tuning represents a promising direction for future research to enhance recommendation system performance.

Appendix A

To load the visual and category features, the following code is added to the `load_all()` in `data-utils.py`.

```
# Load visual and category features
visual_features = np.load(visual_feature_path, allow_pickle=True).item()
category_features = np.load(category_feature_path,
allow_pickle=True).item()
```

This code loads two sets of data, visual features and category features, from saved files. It uses NumPy's load function to read the data, allowing for the loading of Python objects (allow_pickle=**True**).

Each line reads a file and converts it to a dictionary (item()), where each key is an identifier for a specific item, and the value is either a set of visual features or a category label that describes the item's type.

Appendix B

Refer to the following code for actual implementation of the CBMF model.

```
class ContentBasedMF(nn.Module):
    def __init__(self, num_users, num_items, visual_feature_size=512,
category_size=368, embedding_size=8, dropout=0, mean=0):
        super(ContentBasedMF, self).__init__()

        # User embeddings and biases
        self.user_emb = nn.Embedding(num_users, embedding_size)
        self.user_bias = nn.Embedding(num_users, 1)

        # Item embeddings and biases
        self.item_emb = nn.Embedding(num_items, embedding_size)
        self.item_bias = nn.Embedding(num_items, 1)

        # Non-linear transformation for visual features
        self.visual_emb = nn.Sequential(
            nn.Linear(visual_feature_size, embedding_size * 2),
            nn.ReLU(),
            nn.Linear(embedding_size * 2, embedding_size),
        )
        self.visual_bias = nn.Parameter(torch.zeros(1))
```

```
        self.category_emb = nn.Embedding(category_size, embedding_size)
        self.category_bias = nn.Embedding(category_size, 1)

        # Initialization of weights and biases
        self.user_emb.weight.data.uniform_(0, 0.005)
        self.user_bias.weight.data.uniform_(-0.01, 0.01)
        self.item_emb.weight.data.uniform_(0, 0.005)
        self.item_bias.weight.data.uniform_(-0.01, 0.01)

        for layer in self.visual_emb:
            if isinstance(layer, nn.Linear):
                nn.init.uniform_(layer.weight, 0, 0.005)
        self.visual_bias.data.uniform_(-0.01, 0.01)

        self.category_emb.weight.data.uniform_(0, 0.005)
        self.category_bias.weight.data.uniform_(-0.01, 0.01)

        # Mean rating for initialization and dropout for regularization
        self.mean = nn.Parameter(torch.FloatTensor([mean]), False)
        self.dropout = nn.Dropout(dropout)

    def forward(self, u_id, i_id, weight, visual_features,
category_features):
        # Get user and item embeddings
        U = self.user_emb(u_id)
        I = self.item_emb(i_id)

        # Get biases
        b_u = self.user_bias(u_id).squeeze()      # User bias
        b_i = self.item_bias(i_id).squeeze()      # Item bias
        b_c = self.category_bias(category_features).squeeze()  #
Category bias

        # Get content feature embeddings
        V = self.visual_emb(visual_features)        # Visual feature
embeddings
        C = self.category_emb(category_features)    # Category feature
embeddings

        I = (1 - weight) * I + weight * (V + C)
        prediction = ((U * I).sum(1) + b_u + b_i + weight *
(self.visual_bias + b_c) + self.mean)

        return self.dropout(prediction)
```

The CBMF model enhances traditional matrix factorization by incorporating visual and category features of items. It uses non-linear transformations for visual data to capture complex patterns and linear embeddings for category data to maintain categorical distinctions.

The model combines these enriched item representations with user embeddings, applying a content weight to balance the influence of visual and categorical data, aiming to improve prediction accuracy in recommendations.

Appendix C

Refer to the following code for actual implementation of the ILD metric.

```
def intra_list_diversity(recommends, category_features):
    K = 10  # Top K items to consider for diversity calculation
    total_diversity = 0

    for rec_list in recommends:
        diversity_count = 0
        # Considering all unique pairs in the top K recommendations
        for i in range(K):
            for j in range(i + 1, K):
                # Increment diversity count if the categories are
different
                if category_features[rec_list[i]] !=
category_features[rec_list[j]]:
                    diversity_count += 1

        # Compute diversity for the current list
        list_diversity = (2 * diversity_count) / (K * (K - 1))
        total_diversity += list_diversity

    # Average diversity across all recommendation lists
    avg_diversity = total_diversity / len(recommends) if recommends else
0
    return avg_diversity
```

This function calculates the diversity within recommended lists of items based on their categories. It evaluates diversity by counting the number of distinct pairs of items with different categories within the top 10 recommended items for each list. The diversity for each list is then normalized by the total possible number of distinct pairs. The function finally returns the average diversity across all provided recommendation lists, providing a measure of how varied the recommended items are in terms of their categories.

Appendix D

Refer to the following code for actual implementation of the ILD-VC metric.

```
def intra_list_diversity_with_visual_category(recommends,
visual_features, category_features, use_content_features,
visual_weight=0.5):
      if not use_content_features:
        # For MF model, calculate diversity based on categories alone
            total_category_diversity = 0
            for rec_list in recommends:
                  categories_in_list = set(category_features[item] for
item in rec_list)
            # Diversity is the proportion of unique categories in the
recommendation list
                  list_diversity = len(categories_in_list) /
len(rec_list)
                  total_category_diversity += list_diversity

            return total_category_diversity / len(recommends) if
len(recommends) > 0 else 0

      # ContentBasedMF Model
      total_diversity = 0
      num_lists = len(recommends)

      for rec_list in recommends:
            list_diversity = 0
            num_pairs = 0

            for i in range(len(rec_list)):
                  for j in range(i + 1, len(rec_list)):
                        # Visual dissimilarity
                        vis_dissimilarity =
visual_dissimilarity(visual_features[rec_list[i]],
visual_features[rec_list[j]])

                        # Category dissimilarity
                        cat_dissimilarity =
category_dissimilarity(category_features[rec_list[i]],
category_features[rec_list[j]])

                        # Combined dissimilarity (weighted average)
                        combined_dissimilarity = (visual_weight *
vis_dissimilarity) + ((1 - visual_weight) * cat_dissimilarity)
```

```
                        list_diversity += combined_dissimilarity
                        num_pairs += 1

            if num_pairs > 0:
                  total_diversity += list_diversity / num_pairs

      return total_diversity / num_lists if num_lists > 0 else 0
```

This metric measures diversity within recommended lists by considering both visual and category features of items.

For the MF model, it calculates diversity based solely on the unique categories present in each recommendation list. For the CBMF model, it evaluates diversity by calculating a weighted average of visual and category dissimilarities between each pair of recommended items.

This diversity metric integrates visual differences alongside category differences, allowing a nuanced assessment of diversity that accounts for both content type and visual appearance, with the balance controlled by a `visual_weight` parameter.

Appendix E

Refer to the following code for actual implementation of the Item Coverage metric.

```
def item_coverage(recommends, num_total_items):
      # Get all unique items from all recommended item lists
    unique_items = set(item for rec_list in recommends for item in
rec_list)
    item_coverage = len(unique_items) / num_total_items
    return item_coverage
```

This function calculates the coverage of unique items recommended across all user recommendation lists compared to the total number of items available. It aggregates all unique items recommended and divides by the total number of items in the dataset, providing a metric that shows how broadly the recommendation system explores the item space.

This helps evaluate the system's ability to recommend a diverse set of items, rather than repeatedly suggesting the same few items.

Appendix F

Refer to the following code for actual implementation of the Category Coverage metric.

```
def category_coverage(recommends, category_features):
    # Track unique categories in the recommended items
    unique_categories = set()

    for rec_list in recommends:
        for item_id in rec_list:
            # Add the category of each recommended item to the set of
unique categories
            unique_categories.add(category_features[item_id])

    # Calculate the category coverage
    category_coverage = len(unique_categories) / 368

    return category_coverage
```

This function assesses the diversity of item categories in recommendations by determining the proportion of unique categories found in all recommended lists relative to a total of 368 possible categories. It iterates over each recommended item list, collecting unique categories from each item based on their `category_features`, thereby evaluating how well the recommendation system covers the spectrum of available item categories.

This metric is crucial for understanding whether the system can recommend a wide variety of content types to users.

Appendix G

Refer to the following code for actual implementation of the F1 score metric.

```
def calculate_f1(ndcg, ild):
    return 2 * (ndcg * ild) / (ndcg + ild) if (ndcg + ild) > 0 else 0
```

This function computes the F1 score as the harmonic mean of two recommendation system metrics: NDCG and LD. It effectively balances the accuracy of recommendations (NDCG) with their diversity (ILD), providing a single measure that accounts for both aspects.

This calculation ensures that high scores are only achieved when both NDCG and ILD are favorable, encouraging recommendation systems to maintain a balance between relevance and variety.

Appendix H

Refer to the following code for actual implementation of the early stopping.

```
# Early Stopping Check based on F1 Score on valid set
if valid_recall > best_recall:
      best_recall = valid_recall
      no_improvement_count = 0
      best_epoch = epoch
      print('---'*18)
      print(f"Highest Recall: {best_recall} at epoch: {best_epoch}")
      print('---'*18)
      best_results = (valid_recall, valid_ndcg)
      best_valid_ild = valid_ild[0]
      best_valid_ild_vc = valid_ild_vc[0]
      best_valid_item_coverage = valid_item_coverage[0]
      best_valid_category_coverage = valid_category_coverage[0]
      best_valid_f1 = valid_f1
      best_test_results = (test_recall, test_ndcg)
      best_test_ild = test_ild[0]
      best_test_ild_vc = test_ild_vc[0]
      best_test_item_coverage = test_item_coverage[0]
      best_test_category_coverage = test_category_coverage[0]
      best_test_f1 = test_f1

      # save model
      if not os.path.exists(args.model_path):
            os.mkdir(args.model_path)
      torch.save(model,
'{}BestRecall_NDCG_{}_{}_{}lr_{}dr.pth'.format(args.model_path,
args.weight, args.model, args.lr, args.dropout))

else:
      no_improvement_count += 1
      if no_improvement_count >= patience:
            print(f"No improvement in Recall for {patience} consecutive
epochs. Stopping early.")
            break
```

The early stopping mechanism implemented in this snippet is used to halt training when there is no improvement in the F1 score on the validation set over a specified number of epochs, defined by patience.

When a new best F1 score is found, the model updates its best performance metrics, saves the model state, and resets a counter tracking epochs without improvement. If the F1 score does not improve for patience consecutive epochs, training stops prematurely.

This approach helps prevent overfitting by stopping training when the model no longer gains significant learning from additional epochs.

Appendix I

Refer to the following code for actual implementation of the TensorBoard Writer.

```
# Logging metrics to TensorBoard
writer.add_scalars('Recall', {'Valid': valid_recall[0], 'Test':
test_recall[0]}, epoch)
writer.add_scalars('NDCG', {'Valid': valid_ndcg[0], 'Test':
test_ndcg[0]}, epoch)
writer.add_scalars('ILD', {'Valid': valid_ild[0], 'Test': test_ild[0]},
epoch)
writer.add_scalars('ILD_Visual_Category', {'Valid': valid_ild_vc[0],
'Test': test_ild_vc[0]}, epoch)
writer.add_scalars('Item_Coverage', {'Valid': valid_item_coverage[0],
'Test': test_item_coverage[0]}, epoch)
writer.add_scalars('Category_Coverage', {'Valid':
valid_category_coverage[0], 'Test': test_category_coverage[0]}, epoch)
writer.add_scalars('F1_Score', {'Valid': valid_f1, 'Test': test_f1},
epoch)
```

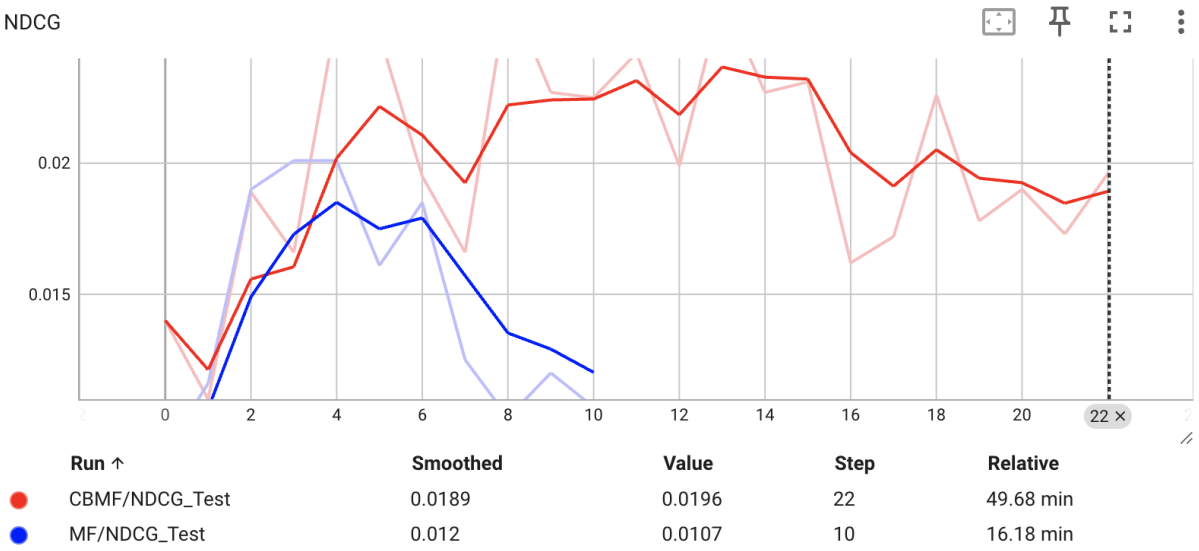
This code snippet uses the TensorBoard logging utility to track and visualize key performance metrics over training epochs for both validation and test datasets. Metrics such as Recall, NDCG, ILD, ILD-VC, Item Coverage, Category Coverage, and F1 Score are logged, allowing for a detailed comparison of model performance across different stages of training.

This facilitates a more intuitive understanding of how the model's predictions evolve and improve with further training, and how effectively the model generalizes from training to testing conditions.

Appendix J

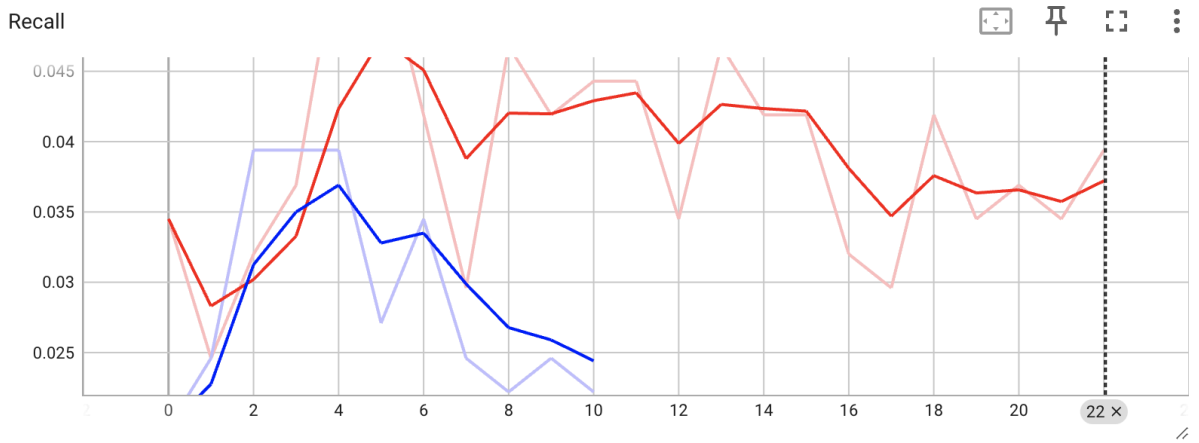
Best Model	MF	CBMF
F1	Best Epoch with highest F1 score is 0. Recall: 0.0197 **NDCG: 0.009** **ILD: 0.9964** ILD-VC: 0.9840 Item Coverage: 0.0466 Category Coverage: 0.1902 **F1: 0.0178**	Best Epoch with highest F1 score is 4. Recall: 0.0542 **NDCG: 0.0256** **ILD: 0.9690** ILD-VC: 0.9332 Item Coverage: 0.4319 Category Coverage: 0.8043 **F1: 0.0499**
Recall	Best Epoch with highest recall is 1. Recall: 0.0246 **NDCG: 0.0116** **ILD: 0.9955** ILD-VC: 0.9798 Item Coverage: 0.1338 Category Coverage: 0.3940 **F1: 0.0229**	Best Epoch with highest recall is 13. Recall: 0.0468 **NDCG: 0.0264** **ILD: 0.9801** ILD-VC: 0.9537 Item Coverage: 0.5818 Category Coverage: 0.8940 **F1: 0.0514**
NDCG	Best Epoch with highest NDCG is 0. Recall: 0.0197 **NDCG: 0.009** **ILD: 0.9964** ILD-VC: 0.9840 Item Coverage: 0.0466 Category Coverage: 0.1902 **F1: 0.0178**	Best Epoch with highest NDCG is 4. Recall: 0.0542 **NDCG: 0.0256** **ILD: 0.9690** ILD-VC: 0.9332 Item Coverage: 0.4319 Category Coverage: 0.8043 **F1: 0.0499**

Figure 1: Results of selection of best model by highest F1 score, recall or NDCG.



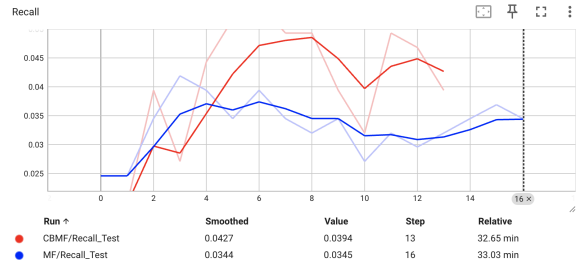
Run ↑	Smoothed	Value	Step	Relative
CBMF/NDCG_Test	0.0189	0.0196	22	49.68 min
MF/NDCG_Test	0.012	0.0107	10	16.18 min

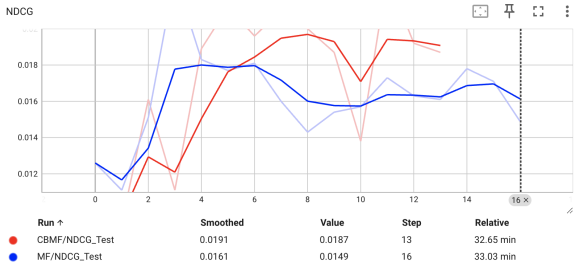
Figure 3: Recall for MF and CBMF.

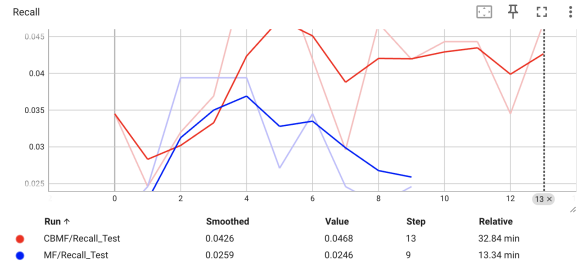


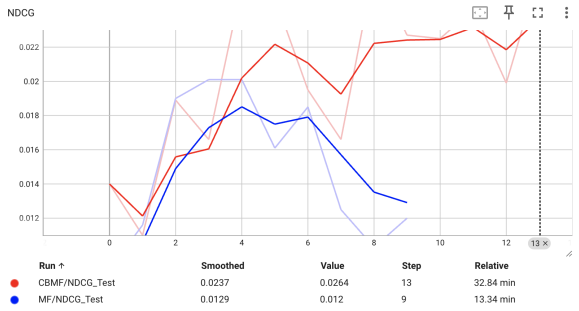
Run ↑	Smoothed	Value	Step	Relative
CBMF/Recall_Test	0.0372	0.0394	22	49.68 min
MF/Recall_Test	0.0244	0.0222	10	16.18 min

Figure 4: NDCG for MF and CBMF.

<table>
<tr><th colspan="2">0.0005 Learning Rate</th></tr>
<tr>
<td>Recall
 

CBMF: best recall 0.0443 at epoch 4
MF: best recall 0.0345 at epoch 7</td>
<td>NDCG
 

CBMF: best NDCG 0.0189 at epoch 4
MF: best NDCG 0.016 at epoch 7</td>
</tr>
<tr><th colspan="2">0.001 Learning Rate</th></tr>
<tr>
<td>Recall
 

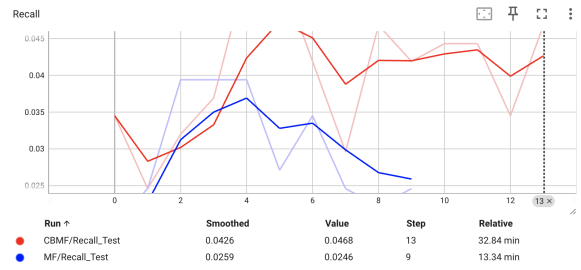
CBMF: best recall 0.0542 at epoch 4
MF: best recall 0.0197 at epoch 0</td>
<td>NDCG
 

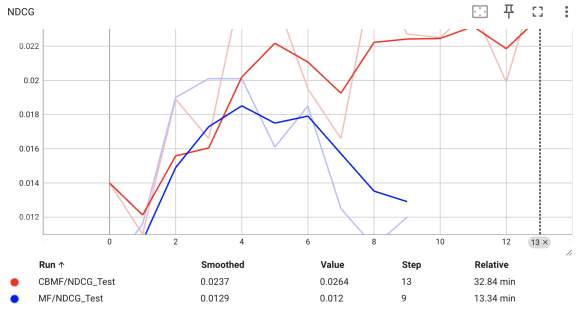
CBMF: best NDCG 0.0256 at epoch 4
MF: best NDCG 0.009 at epoch 0</td>
</tr>
<tr><th colspan="2">0.01 Learning Rate</th></tr>
<tr>
<td>Recall
 

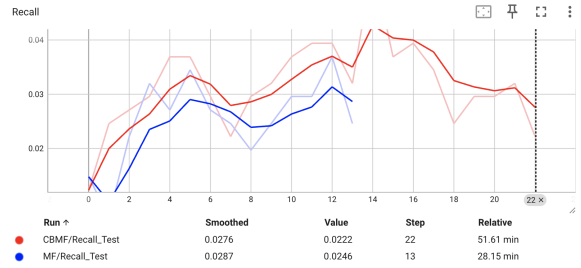
CBMF: best recall 0.032 at epoch 18
MF: best recall 0.0172 at epoch 19</td>
<td>NDCG
 

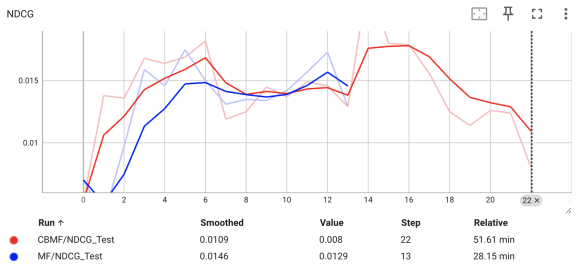
CBMF: best NDCG 0.0142 at epoch 18
MF: best NDCG 0.0086 at epoch 19</td>
</tr>
</table>

Figure 5: Impact of learning rates on convergence speed and accuracy.

<table>
<tr><th colspan="2">0 Dropout Rate</th></tr>
<tr>
<td>Recall
 

CBMF: best recall 0.0542 at epoch 4
MF: best recall 0.0197 at epoch 0</td>
<td>NDCG
 

CBMF: best NDCG 0.0256 at epoch 4
MF: best NDCG 0.009 at epoch 0</td>
</tr>
<tr><th colspan="2">0.2 Dropout Rate</th></tr>
<tr>
<td>Recall
 

CBMF: best recall 0.032 at epoch 13
MF: best recall 0.0271 at epoch 4</td>
<td>NDCG
 

CBMF: best NDCG 0.0129 at epoch 13
MF: best NDCG 0.0146 at epoch 4</td>
</tr>
</table>

Figure 6: Impact of dropout rates on accuracy.

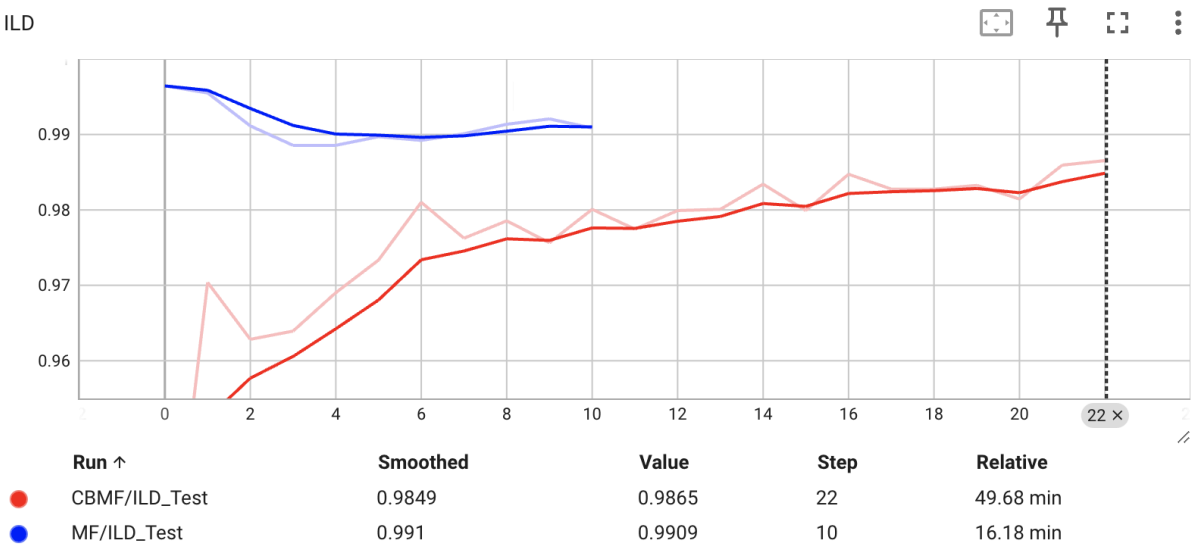


Figure 7: Intra-List Diversity (ILD) diagram

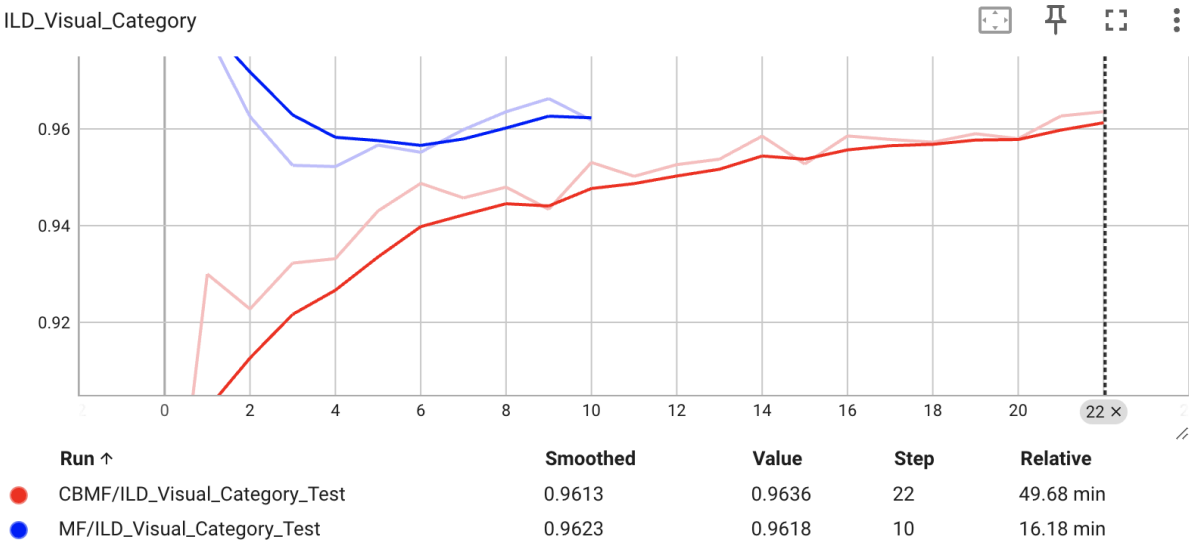


Figure 8: Intra-List Diversity with Visual and Category features (ILD-VC) diagram

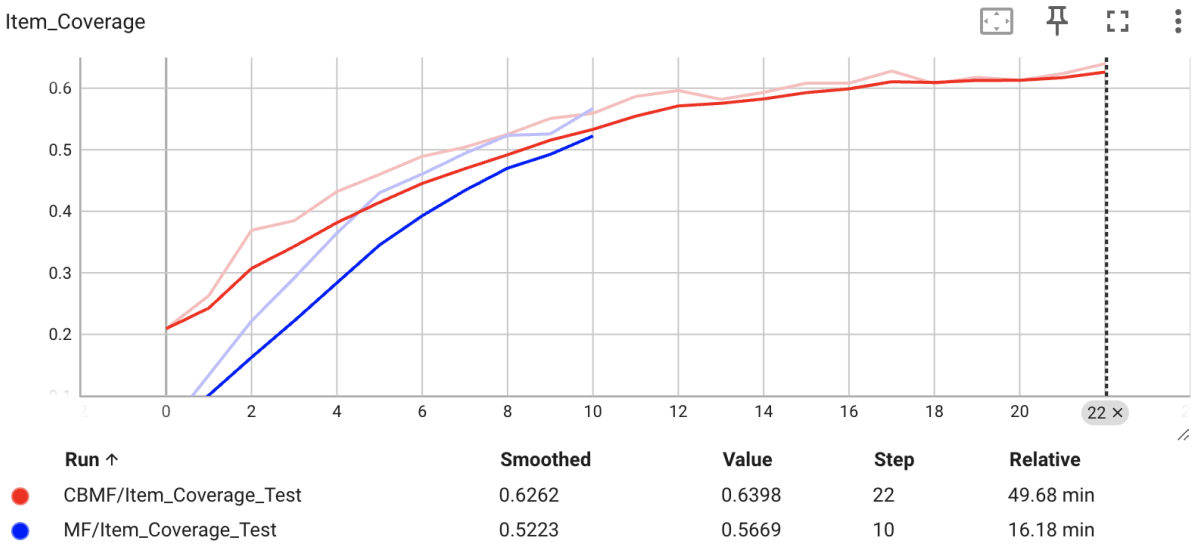


Figure 9: Item coverage diagram

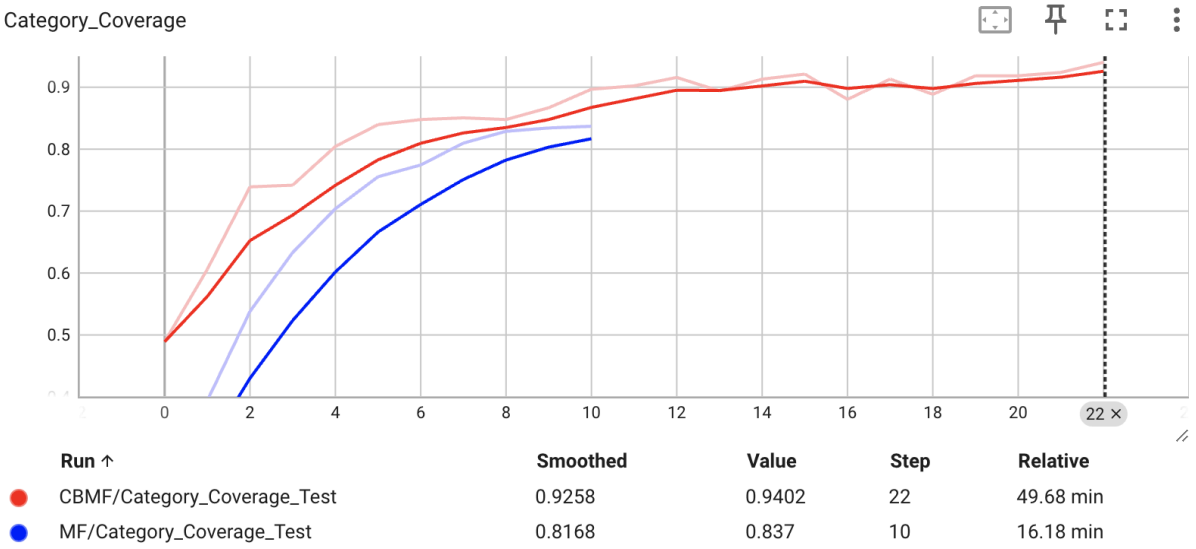
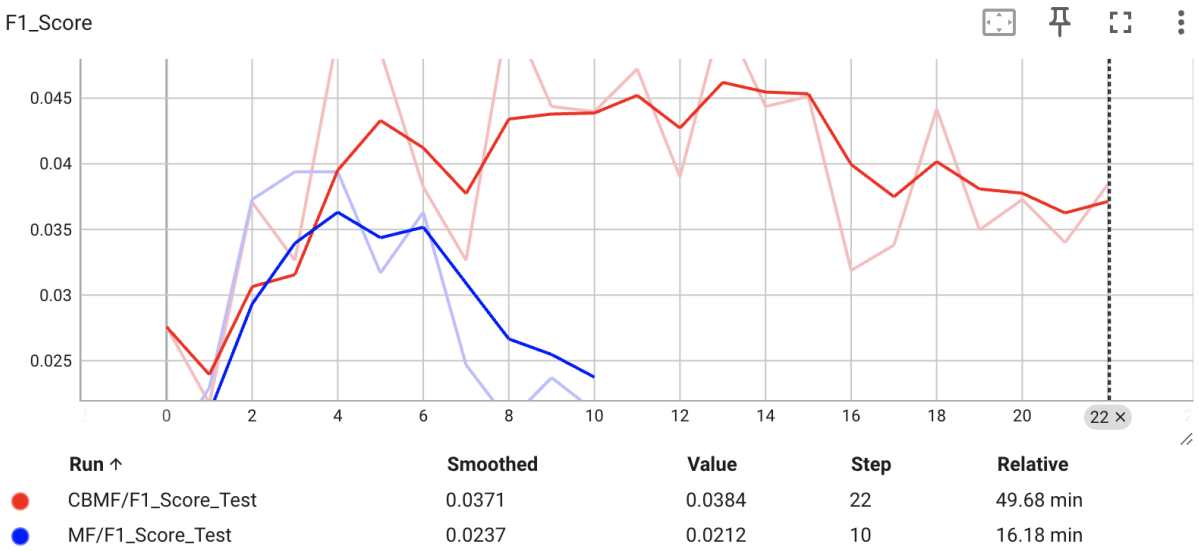


Figure 10: Category coverage diagram



Run ↑	Smoothed	Value	Step	Relative
CBMF/F1_Score_Test	0.0371	0.0384	22	49.68 min
MF/F1_Score_Test	0.0237	0.0212	10	16.18 min

Figure 11: F1 score for MF and CBMF.