CS4242: Assessment of Recommendation Project

- **Algorithm exploration**: the implementation and presentation of the extended content-based algorithms.
- **Performance Evaluation**: To focus on the Accuracy & Diversity of the results.
- **For project report**, you are expected to explore different variants of algorithms with analysis of results and insights
    - **For algorithm**, you can design and analyse different variants of algorithms to arrive at the best version
    - For **Performance**:
        - You are encouraged to explore different metrics for Accuracy and Diversity.
        - You may consider, e.g., Recall or NDCG for Accuracy, and Coverage for Diversity.
        - You might utilize F1 score (which gives equal weight to accuracy and diversity) to analyse the performance
        - The metrics employed should be reasonable and self-justifiable for analysing the results.

The Metric to be Used for Project Assessment

- **Performance**: The accuracy & diversity of the results.
- For **online evaluation:**
  - **Accuracy:** NDCG@10 = NDCG of the Top-10 recommended items (metric implementation is given in your code)
  - **Diversity:** Intra-List-Diversity@10 (K=10 in below equation)

$$ILD = \frac{2}{K(K-1)} \sum_{i=1}^{K} \sum_{j \neq i}^{K} \mathbb{I}\,(category_i \neq category\_j)$$

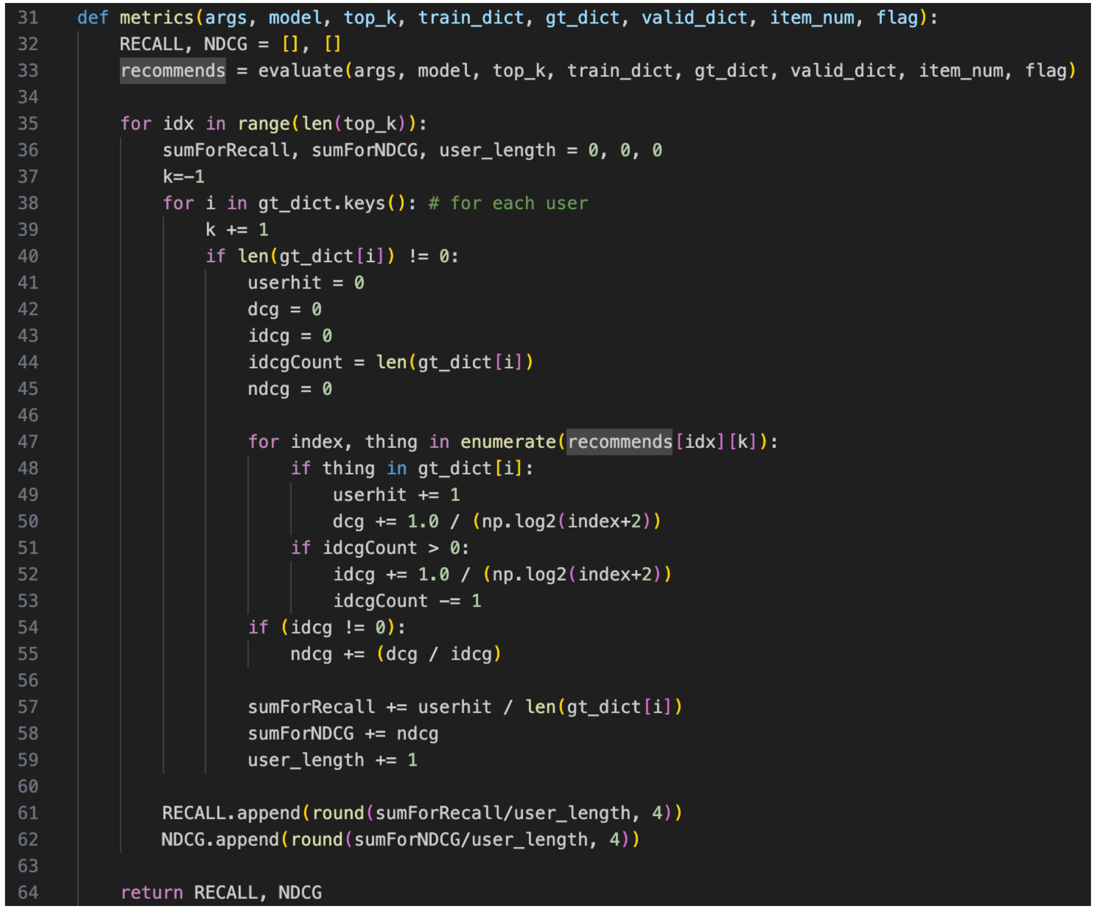
    - $\mathbb{I}\,(category_i \neq category\_j)$ is the indicator function whose value is set to 1 if the category of item i and item j is different, otherwise 0.
  - **F1 measure** (NDCG-ILD)

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

  - **We will calculate the F1 score for each user, and then use the averaged F1 score across all users to evaluate your model.**

Online Evaluation

- **A held-out testing set will be given**
  - 100 users, in the same format as the testing_dict.npy
- **Evaluation metrics for online evaluation**
  - F1 (NDCG@10, ILD@10)
  - The evaluation script, including the NDCG, ILD, F1 metrics will be given before the online evaluation.
    - A new function of **metrics** -- make sure that it works on your code.



```
31  def metrics(args, model, top_k, train_dict, gt_dict, valid_dict, item_num, flag):
32      RECALL, NDCG = [], []
33      recommends = evaluate(args, model, top_k, train_dict, gt_dict, valid_dict, item_num, flag)
34  
35      for idx in range(len(top_k)):
36          sumForRecall, sumForNDCG, user_length = 0, 0, 0
37          k=-1
38          for i in gt_dict.keys(): # for each user
39              k += 1
40              if len(gt_dict[i]) != 0:
41                  userhit = 0
42                  dcg = 0
43                  idcg = 0
44                  idcgCount = len(gt_dict[i])
45                  ndcg = 0
46  
47                  for index, thing in enumerate(recommends[idx][k]):
48                      if thing in gt_dict[i]:
49                          userhit += 1
50                          dcg += 1.0 / (np.log2(index+2))
51                      if idcgCount > 0:
52                          idcg += 1.0 / (np.log2(index+2))
53                          idcgCount -= 1
54                  if (idcg != 0):
55                      ndcg += (dcg / idcg)
56  
57                  sumForRecall += userhit / len(gt_dict[i])
58                  sumForNDCG += ndcg
59                  user_length += 1
60  
61          RECALL.append(round(sumForRecall/user_length, 4))
62          NDCG.append(round(sumForNDCG/user_length, 4))
63  
64      return RECALL, NDCG
```

- Make sure that your model will output the “recommendations” by calling the evaluate function (line 33)
  - a recommendation lists
  - E.g., [[1,9,128,43,98,666,7,8,987,10], [10,9,8,7,6,5,4,3,2,1], …, [0,2,1,5,6,7,888,4,3,9]]

user_1’s
recommendation list

user_2’s
recommendation list

user_100’s
recommendation list