1 PSEUDOCODE

Algorithm 1: CDM

Input: Target user u, training dataset $\mathcal{I}_{training}$, testing dataset $\mathcal{I}_{testing}$, the sampling parameter k, the output list size K and the backbone model f;

Output: the Top-K list of $\mathcal{I}_{testing}$;

// start training;
Initialize the student model stu;
Choose the initial item $i_1 = \underset{i \in I_{training}}{\operatorname{argmax}} f(u, i)$;
Select the remaining $K-1$ items by Eq.(2);
Label the top-K items as positive samples and the rest as negative samples
Compute $y_{tea} = \mathbb{I}\left(i^t \in \mathcal{R}\right)$
for *stop condition is not reached* **do**
    Randomly sample a batch of data from $\mathcal{I}_{training}$;
    for *each candidate item in the batch* **do**
        Set the item as the target item i_t;
        Employ Gumbel-Top-k trick to sample $\mathcal{I}_t^{pos} = \left\{i_1^{pos}, i_2^{pos}, \ldots, i_k^{pos}\right\}$ and $\mathcal{I}_t^{neg} = \left\{i_1^{neg}, i_2^{neg}, \ldots, i_k^{neg}\right\}$ according to Eq.(5~10);
        Compute the attention score w between the target item and its sampled candidate items following Eq.(11);
        Compute the positive context embedding $\mathbf{C}^{pos} = \text{softmax}\left(\widetilde{\mathbf{w}}^{pos}\right) \cdot \mathbf{V}^{pos}$ and the negative context embedding $\mathbf{C}^{neg} = \text{softmax}\left(-\widetilde{\mathbf{w}}^{neg}\right) \cdot \mathbf{V}^{neg}$;
        Compute the final context embedding $\mathbf{C} = \text{FFN}\left(\text{concat}\left(\mathbf{u} \odot \mathbf{C}^{pos}, \mathbf{u} \odot \mathbf{C}^{neg}\right)\right)$;
        Compute the output of the student model y_{stu} by Eq.(17);
        Update the student model by minimizing the loss $\mathcal{L}$ in Eq.(20);
    end
end
// start inference;
for *each candidate item in* $\mathcal{I}_{testing}$ **do**
    Compute the output of the student model ***stu***, getting $y_{stu}(u, i, I)$;
    Compute the output of the backbone model $\boldsymbol{f}$, getting $f(u, i)$;
    Compute $Finalscore = f(u, i) + \gamma * y_{stu}(u, i, I)$;
end
Heap-sort the $\mathcal{I}_{testing}$ by the final scores and get the Top-K list;
return the Top-K list of $\mathcal{I}_{testing}$;