Project 3: Knowledge Graph-based Recommender System

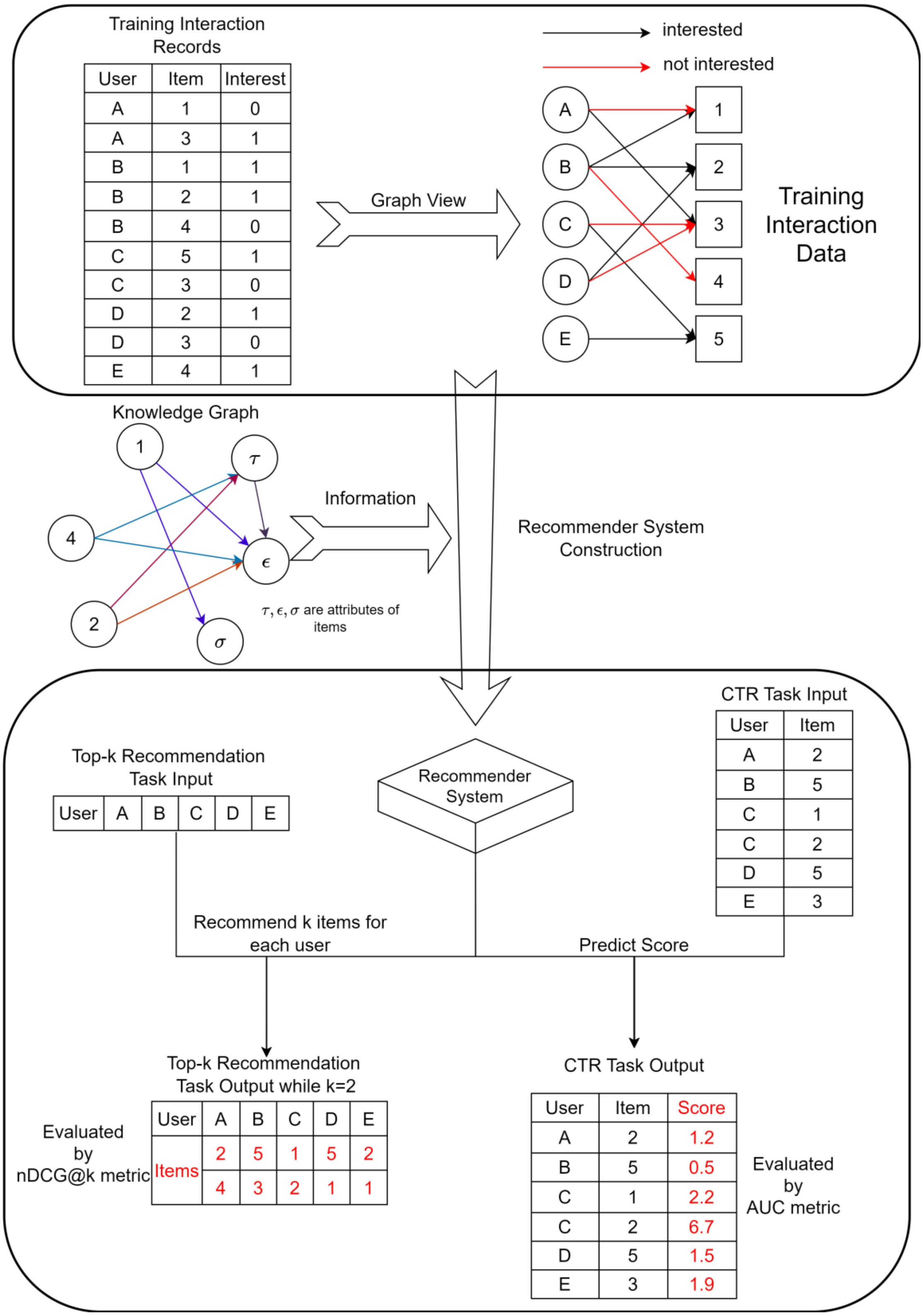
1. Overview

A recommender system is a type of information filtering system that seeks to predict and show the preferences of a user, offering tailored suggestions for items such as books, movies, or music based on their browsing and selection history. These systems utilize algorithms and data analysis to provide personalized recommendations, enhancing user experience and engagement.

Knowledge Graph (KG)-based recommender system technology uses knowledge graphs as auxiliary information to improve the accuracy and explain-ability of the result of recommendations. Knowledge graphs, which are heterogeneous graphs with nodes representing entities and edges representing relationships, help to illustrate the relationships between items and their attributes, and integrate user and user-side information, capturing the relationships between users and items, as well as user preferences, more accurately.

In this project, you need to design an algorithm to construct a KG-based Recommender System, which is used to predict whether a user is interested in an item that they have not encountered before, based on all previously known likes and dislikes of the user towards items, as well as a KG. The score you get will be given according to the performance of your algorithm in our test.

2. Preliminary


interested
not interested
Graph View
A
B
C
D
E
1
2
3
4
5
Training Interaction Data
Knowledge Graph
1
τ
4
ϵ
2
σ
Information
τ, ϵ, σ are attributes of items
Recommender System Construction
Recommender System
Recommend k items for each user
Predict Score
Evaluated by nDCG@k metric
Evaluated by AUC metric

Training Interaction Records

User	Item	Interest
A	1	0
A	3	1
B	1	1
B	2	1
B	4	0
C	5	1
C	3	0
D	2	1
D	3	0
E	4	1

Top-k Recommendation Task Input

User	A	B	C	D	E

CTR Task Input

User	Item
A	2
B	5
C	1
C	2
D	5
E	3

Top-k Recommendation Task Output while k=2

User	A	B	C	D	E
Items	2	5	1	5	2
	4	3	2	1	1

CTR Task Output

User	Item	Score
A	2	1.2
B	5	0.5
C	1	2.2
C	2	6.7
D	5	1.5
E	3	1.9

Figure 1 Illustration of KG-based Recommender System construction and evaluation

2.1 Recommender System

Recommender system (RS) is a predictor used to predict the level of interest in the item from the user. While we build the recommender system, the known data is the interaction records between the users and the items. The interaction records consist of a lot of 3-dimensional vectors, the first dimension is the index of the user, the second dimension is the index of the item, and the third dimension is usually a 0/1 value that represents whether the user is interested in the item or not. The interaction records only include a part of the combinations of all the users and all the items, and the other combinations are unknown. By analyzing known interaction records, the recommender system tries to predict whether the user is interested in the item, while the combination of user and item is unknown prior.

To predict whether the user is interested in the item or not, it can be seen as a binary classification problem, while the input is the user and item, and the output is the prediction result. Practically, the output of the recommender system is not a binary value, but a score representing the level of interest in the item from the user. The higher score means the recommender system thinks the user has more interest in the item.

There are two tasks usually used to evaluate the performance of the recommender system, called the Click-Through-Rate (CTR) prediction task and the Top-k recommendation task. In this project, you are given a training dataset Y_{train} and need to design a recommender system. The training dataset Y_{train} is a $m \times 3$ matrix, and m is the number of the interaction records, while each interaction record is a 3-dimension vector, the first dimension is the index of the user, the second dimension is the index of the item, and the third dimension is a 0/1 value that represents whether the user is interested in the item or not. The performance of your recommender system on the CTR prediction task and the Top-k recommendation task will be evaluated on a testing dataset that is unseen to you. The detailed task content of CTR prediction and Top-k recommendation will be introduced below.

In the CTR prediction task, you will be given an $n \times 2$ matrix, and n is the number of the samples that need to be predicted, it is also called as the test dataset, marked as Y_{test}. In each row of the matrix, there are 2 elements, the first one is the index of the user and the second one is the index of the item. For each row, you need to give a float score indicating the interest level from the user to the item, a higher score means the user has a higher interest in the item. In this task, you should return an array with length n, the ith element in the array corresponding to the prediction result of the ith row in the given $n \times 2$ matrix. In this project, the performance of the CTR prediction task will be evaluated by the AUC (Area Under Curve) metric. The calculation of AUC is as follows. Suppose the samples of the test dataset T_{test} can be split to positive set S and negative set S' according to the ground truth, while all the samples in S are positive samples and all the samples in S' are negative samples. For a sample s, the score you predicted will be marked as $\mathrm{score}(s)$, then we can define the calculation process of AUC is shown below:

$$\mathrm{AUC} = \frac{\sum_{s\in S}\sum_{s'\in S'}\mathrm{I}(s,s')}{|S| \times |S'|},$$

the function I is:

$$\mathrm{I}(s,s') = \begin{cases} 0, & \mathrm{score}(s) < \mathrm{score}(s') \\ 0.5, & \mathrm{score}(s) = \mathrm{score}(s') \\ 1, & \mathrm{score}(s) > \mathrm{score}(s') \end{cases}$$

In the Top-k recommendation task, you will be given a user list U_{test}, which consists of l user indices. For each user in U_{test}, you need to recommend k items for him, they should be the k items that the user is most interested in according to the recommender system you design. The data you return should be a $l \times k$ matrix, the ith row of the matrix you return consists of the k items that the ith user is most interested in according to your recommender system. The items should be ordered by the interest level from the ith user to the item, the interest level is also predicted by your recommender system. In this project, there is $k = 5$ in the Top-k recommendation task and the task performance will be evaluated by the nDCG@k metric, where nDCG is the abbreviation of normalized Discounted Cumulative Gain. Suppose for the ith user in U_{test}, according to the ground truth, there is a positive item set S_i, which contains all the items that the user u is interested in. Assume the return matrix is M, we can define the calculation process of nDCG@k as shown below and k is set to 5 in this project:

$$\mathrm{nDCG@}k(f, Y_{test}) = \frac{1}{l}\sum_{i=1}^{l}\frac{\mathrm{DCG}_i@k(S_i, M_i)}{\mathrm{iDCG}_i@k(S_i)},$$

where $\mathrm{iDCG}_i@k$ is the theoretical maximum value of $\mathrm{DCG}_i@k$ when the ground truth for the ith user is S_i. And $\mathrm{DCG}_i@k$ is defined as:

$$\mathrm{DCG}_i@k(S_i, M_i) = \sum_{j=1}^{k} \frac{\mathrm{I}(S_i, M_{ij})}{\log_2(j+1)},$$

$$\mathrm{iDCG}_i@k(S_i) = \sum_{j=1}^{\min(k,|S_i|)} \frac{1}{\log_2(j+1)},$$

and $\mathrm{I}(S_i, M_{ij})$ is:

$$\mathrm{I}(S_i, M_{ij}) = \begin{cases} 1, & M_{ij} \text{ is in } S_i \\ 0, & M_{ij} \text{ is not in } S_i \end{cases}.$$

2.2 Knowledge Graph

Knowledge Graph (KG) is an abstract representation of knowledge in the real world. It records the facts by describing the relationship between **entities**. In the knowledge graph, the entity is a concrete thing or concept, it is always a subject or an object in a sentence that describes a fact. Knowledge graph is always represented as a heterogeneous directed graph, which means the vertices and edges in the graph have different categories, and the edges are directed.

If we represent the knowledge graph as a heterogeneous directed graph, the entities in the knowledge graph correspond to the vertices in the graph, and the relations between entities in the knowledge graph correspond to the edges in the graph, which means the relations in the knowledge graph are also directed. For example, there is a knowledge graph that describes the information about SUSTech. There are some entities CSE, $\mathrm{Xin\ Yao}$, $\mathrm{Collage\ of\ Engineering}$. There are also some relations between these entities, such as $(\mathrm{CSE}, \mathit{Department\ Head}, \mathrm{Xin\ Yao})$, $(\mathrm{College\ of\ Engineering}, \mathit{Subordinate}, \mathrm{CSE})$. This expression of the relation is called as relation triple. There are three elements in the triples, head entity, relation type, and tail entity. Head entity is the subject of this relation, relation type indicates the category of this relation, and tail entity is the object of this relation. In the relation $(\mathrm{CSE}, \mathit{Department\ Head}, \mathrm{Xin\ Yao})$, its head entity is CSE, relation type is $\mathit{Department\ Head}$, and the tail entity is $\mathrm{Xin\ Yao}$. From the heterogeneous directed graph representation and relation triples representation of knowledge graph, we can find that a relation triple corresponds to an edge of the graph. The head entity is the start vertex of the edge, the tail entity is the target vertex of the edge, and the relation type is the edge category.

The knowledge graph stores so many facts that it can supply some prior information to the recommender system construction process to help improve the performance of the recommender system.

3. Problem Formulation

Given an interaction record set Y_{train}, and a knowledge graph $\mathcal{G} = (V, E)$. For each interaction record $y_{uw} \in Y_{train}, u \in U, w \in W$, where U is the user set and W is the item set, it is a 0/1 value that represents whether the user is interested in the item or not, 1 means interested in and 0 means not. $\mathcal{G} = (V, E)$ is a knowledge graph about the items, V is the entity set and E is the relation set, which means that the entities in V are items and something that is related to the items, and the relations in E describe and the relationship between the items or their attributes. Based on the given Y_{train} and $\mathcal{G}$, you are asked to design a recommender system with a score function $f(u, w)$, which is used to predict the interest level from user u to item w, the higher score means the higher interest level. There are two tasks in this project you need to do:

(1) Maximum the AUC metric of your score function $f(u, w)$ on a test dataset Y_{test}, i.e.,

$$\max_{f} \mathrm{AUC}(f, Y_{test})$$

(2) Maximum the nDCG@k metric of your score function $f(u, w)$ on a test dataset Y_{test}, while $k = 5$, i.e.,

$$\max_{f} \mathrm{nDCG@5}(f, Y_{test})$$

4. Submission and Evaluation

4.1 Programming Language

Python 3.10

4.2 Library Support

torch 1.13 (only CPU)

numpy	1.26.2
sklearn	1.3.0
scipy	1.10.1
networkx	3.2.1

4.3 Evaluation Environment

During the execution of your algorithm, you will be restricted from using a maximum of 8 CPU cores and 8GB of memory. The initialization of a KGRS instance must be completed within 60 seconds. Additionally, your algorithm's training process should conclude in 600 seconds. For task-specific evaluations, the CTR task should be completed in 30 seconds, and the Top-k Recommendation task evaluation should be finished within 60 seconds.

If your initialization stage exceeds the time limit, you will lose all scores. Exceeding the time limit during the training stage will not result in score loss, but the training will be suspended, and the evaluation phase will go ahead. However, if you exceed the time limit during any evaluation stage, you will lose the scores for that specific evaluation task.

4.4 Project Submission

The code files should be submitted on the OJ platform, and the URL of the OJ platform is: https://spaces.sustech.cloud/ . You need to submit a ZIP file that contains at least one file: the main code file \`kgrs.py\` and it should be placed in the root directory of your submission ZIP file. You can add other necessary files in your folder and the total size of your ZIP file cannot exceed 1MB. The main code file \`kgrs.py\` should comply with the requirements below.

In the \`kgrs.py\` file, you need to implement a Python Class named \`KGRS\` and the class must have the following four member functions:

```
from typing import Dict, Tuple, List
import numpy as np

class KGRS:
    def __init__(self, train_pos: np.array, train_neg: np.array, kg_lines: List[str]):
        """
        Initialize the Algorithm
        :param train_pos: The Positive Samples in the Training Set, is a numpy matrix
```

```
                         with shape (n,3), while `n` is the number of positive
samples,
                        and in each sample, the first number represent the user, the
                        second represent the item, and the last indicate interest or
                        not. e.g. [[1,2,1], [2,5,1],[1,3,1]] indicate that user 1 has
                        interest in item 2 and item 3, user 2 has interest in item 5.
        :param train_neg: The Negative Samples in the Training Set, is a numpy matrix
                          with shape (n,3), while `n` is the number of positive
                         samples, and in each sample, the first number represent the
                         user, the second represent the item, and the last indicate
                         interest or not. e.g. [[1,4,0], [2,2,0],[1,5,0]] indicate
                         that user 1 has no interest in item 4 and item 5, user 2 has
                         no interest in item 2.
         :param kg_lines: The Knowledge Graph Lines, is a list of strings. Each
                           element in the list is a string representing one relation
                           in the Knowledge Graph. The string can be split into 3
                           parts by '\t', the first part is head entity, the second
                           part is relation type, and the third part is tail entity.
                           E.g. ["749\tfilm.film.writer\t2347"] represent a Knowledge
                           Graph only has one relation, in that relation, head entity
                           is 749, tail entity is 2347, and the relation type is
                           "film.film.writer".
        """
        raise NotImplementedError

    def training(self):
        """
        Train the Recommender System
        :return: None
        """
        raise NotImplementedError

    def eval_ctr(self, test_data: np.array) -> np.array:
        """
        Evaluate the CTR Task result
        :param test_data: The test data that you need to predict. The data is a numpy
                         matrix with shape (n, 2), while `n` is the number of the test
                         samples, and in each sample, the first dimension is the user
                         and the second is the item. e.g. [[2, 4], [2, 6], [4, 1]]
                         means you need to predict the interest level of: from user 2
                         to item 4, from user 2 to item 6, and from user 4 to item 1.
        :return: The prediction result, is an n dimension numpy array, and the i-th
                 dimension means the predicted interest level of the i-th sample,
                 while the higher score means user has higher interest in the item.
                 e.g. while test_data=[[2, 4], [2, 6], [4, 1]], the return value [1.2,
                 3.3, 0.7] means that the interest level from user 2 to item 6 is
                 highest in these samples, and interest level from user 2 to item 4 is
                 second highest, interest level from user 4 to item 1 is lowest.
        """
        raise NotImplementedError

    def eval_topk(self, users: List[int], k: int = 5) -> List[List[int]]:
        """
        Evaluate the Top-K Recommendation Task result
        :param users: The list of the id of the users that need to be recommended
                      items. e.g. [2, 4, 8] means you need to recommend k items for
                      the user 2, 4, 8 respectively, and the term of the user and
                      recommended item cannot have appeared in the train_pos data.
        :param k: The number of the items recommended to each user. In this project,
                 k=5.
        :return: The items recommended to the users respectively, and the order of the
                 items should be sorted by the interest level of the user to the item.
                 e.g. while user=[2, 4, 8] and k=5, the return value is [[2, 5, 7, 4,
                 6],[3, 5, 2, 1, 21],[12, 43, 7, 3, 2]] means you will recommend item
                 2, 5, 7, 4, 6 to user 2, recommend item 3, 5, 2, 1, 21 to user 4, and
                 recommend item 12, 43, 7, 3, 2 to user 8, and the interest level from
                 user to the item in the recommend list are degressive.
        """
        raise NotImplementedError
```

4.5 Dataset

The Recommender System dataset we use comes from MoviesLens-1M, and the interaction records have been transformed to 0-1 type. The training data will be released to you on the Blackboard platform and the test data won't be released. In the training dataset, there are 26,638 positive samples and 24,037 negative samples. The numbers in the test dataset will be 10~15 times those in the training dataset.

The training data contains two files, ` train_pos.npy` and `train_neg.npy`, which contain the positive samples and negative samples of the training data. Each `npy` file is a $n \times 3$ dimension matrix, while n is the number of the samples. The three values in each row represent the user id, item id and interest or not, while the third value equal to 0 means the user has no interest in the item, and 1 means the user has interest.

There are 6729 nodes and 20195 relations in the knowledge graph we provided, while there are 7 kinds of relations in the KG. The 7 relations are: film.film.star, film.film.genre, film.film.writer, film.film.director, film.film.rating, film.film.language, and film.film.country. Furthermore, the KG is about the items in the interaction data, and the item indices in the interaction data are the same as the indices of the corresponding entities in the KG. The users in the interaction data don't appear in the KG. E.g. if there are 5 items with indices 1, 3, 5, 8, 9 in the interaction data, the entities in the KG with the indices 1, 3, 5, 8, 9 are corresponding to these items, and the other entities in the KG are the attributes of the items, and the relationships between attributes and items are shown in the KG's relations.

The KG data is a file named \`kg.txt\` with 20195 rows, and each row can be split into 3 parts by the "\t" character. The row in the KG data represents a relation, and the 3 parts of the row represent the head entity, relation type, tail entity of the relation respectively.

5. Grading Rules

The scoring rules for this project comprise two components: the project report and code evaluation. The total value of this project is 15 points.

5.1 Project Reports

A report about KGRS (in pdf format) must be submitted, in which you should describe the core idea of your algorithm design, entail each component of your algorithm, illustrate the algorithm structure, and give the pseudo-code.

Please note that your code will only be assessed and scored if you have submitted the project report.

You will get **0** score in this project if you haven't submitted the report!

5.2 Code Evaluation

The evaluation criteria for this project are straightforward, encompassing two key metrics:

AUC and nDCG@5. Each metric is classified into three distinct grade levels. The specific levels for these metrics and their corresponding grades are detailed in the table provided below.

	Score = 5	Score = 7	Score = 7.5
AUC	≥ 0.6	≥ 0.65	≥ 0.7
nDCG@5	≥ 0.05	≥ 0.075	≥ 0.12

E.g. if you get a AUC = 0.64 and nDCG@5 = 0.11 on the test set, the total score of your project 3 will be 5 + 7 = 12.

The total score of the project 3 is 7.5 + 7.5 = 15.

You will get **0** score in this project if you haven’t submitted the report!

6. Reference

[1] Q. Guo et al., “A Survey on Knowledge Graph-Based Recommender Systems,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2020, doi: 10/ghxwqg.