National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Shafik Rahman Salim Badhusha
Student ID: x22123661

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Shafik Rahman Salim Badhusha
Student ID:	x22123661
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	592
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shafik Rahman Salim Badhusha
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shafik Rahman Salim Badhusha
x22123661

1 Introduction

This configuration manual provides a comprehensive guide for replicating the experimental setup and findings of the recommendation systems research project. The study investigates the efficacy of different sequential recommendation models, such as GRU4Rec and BERT4Rec, specifically in the context of fashion domain data. This manual provides a detailed description of the software, packages, and modules used, along with their exact versions. The purpose is to create a consistent environment for replicating the experiments. Furthermore, a systematic series of instructions is included to assist users in configuring the environment, conducting experiments, and analysing the outcomes. Users can reproduce the experimental setup and contribute to the understanding and advancement of sequential recommendation algorithms in real-world scenarios by following this handbook.

2 Development Environment

The development environment utilised here uses Google Cloud Platform (GCP) services to ensure effective data management and powerful computational resources. H&M data is securely stored in cloud-based storage to ensure scalable accessibility. Both hardware and software specifications and tools used are mentioned in detail below.

2.1 Hardware Specification

A compute-optimized virtual machine (VM) was set up on GCP with TensorFlow 2.13, a prominent deep learning framework. The virtual machine (VM) was equipped with 8 virtual central processing units (vCPUs) and 32 gigabytes (GB) of random access memory (RAM), guaranteeing significant computing capacity for model training and analysis. The specifications are shown in the below figure.

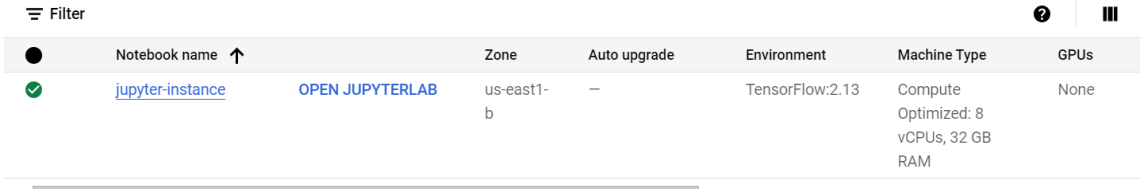

Figure 1: Google Cloud Instance

2.2 Software requirements

The tools and programming language used are mentioned below.

- **Programming Language**: Python version 3.10.13
- **Integrated Development Environment (IDE)**: Jupyter Lab

2.3 Storage Requirement :

Data storage was crucial in this research, with Google Cloud Storage acting as the main store for datasets. All three articles, customer information, and transaction records are stored here. Cloud storage facilitated the ability to access data in a scalable and secure manner, hence enhancing the efficiency of data administration and retrieval. In addition, Google Cloud Storage stores result data and log files, serving as a centralised repository for important outputs and facilitating systematic analysis. The data can be downloaded from the H&M competition hosted in Kaggle.[1]

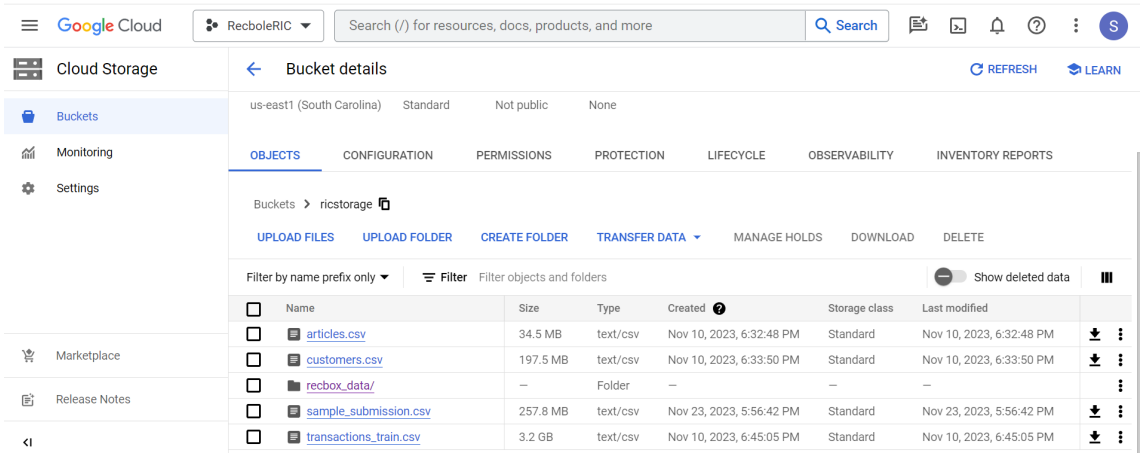

Figure 2: Google Cloud storage

3 Implementation

3.1 Python Libraries required

Figure 3 displays a comprehensive list of the essential Python libraries required for the execution of code. Python libraries can be installed using the pip command.

- numpy
- pandas
- seaborn
- matplotlib
- tqdm

[1]H&M dataset: `https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations`

- kmeans_pytorch
- recbole

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from tqdm.notebook import tqdm
import kmeans_pytorch
import random
import datetime as dt
import torch
from tqdm import tqdm
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
import logging
import logging
from logging import getLogger
from recbole.config import Config
from recbole.data import create_dataset, data_preparation
from recbole.model.sequential_recommender import GRU4Rec
from recbole.model.sequential_recommender import BERT4Rec
from recbole.utils.case_study import full_sort_topk
from recbole.trainer import Trainer
from recbole.utils import init_seed, init_logger
from matplotlib import pyplot as plt
```

Figure 3: Libraries Used

3.2 EDA, Preparation and Modelling

Importing and exploring the fashion datasets, Checking for missing data, and performing cleaning procedures are the first steps in the study. The next step is exploratory data analysis (EDA) and visualisations covering a variety of topics like popular fashion items, customer preferences, and product type. Followed by modelling and evaluation. These step-by-step procedures code

```
articles = pd.read_csv("gs://ricstorage/articles.csv")
customers = pd.read_csv("gs://ricstorage/customers.csv")
transactions = pd.read_csv("gs://ricstorage/transactions_train.csv")
```

Figure 4: Data Loading

```
# Perform the joins
merged_transactions = pd.merge(transactions, customers, on='customer_id', how='inner')
```

```
merged_transactions = pd.merge(merged_transactions, articles, on='article_id', how='inner')
```

Figure 5: Merging Data using Joins

```
merged_transactions['t_dat'] = pd.to_datetime(merged_transactions['t_dat'])
```

```
merged_transactions['year_month'] = merged_transactions['t_dat'].dt.to_period('M')
merged_transactions['year'] = merged_transactions['t_dat'].dt.year
merged_transactions.head()
```

Figure 6: Converting date datatype

```
# Filter the DataFrame to extract data from '2020-08-01' onwards
data_2020 = merged_transactions[merged_transactions['t_dat'] >= '2020-08-26']
#data_2020 = merged_transactions[merged_transactions['t_dat'] >= '2020-08-01']
```

Figure 7: Filtering data using Date

```
def missing_data(data):
    total = data.isnull().sum().sort_values(ascending = False)
    percent = (data.isnull().sum()/data.isnull().count()*100).sort_values(ascending = False)
    return pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
```

```
missing_data(data_2020)
```

Figure 8: Function to find missing values

```
def unique_values(data):
    total = data.count()
    tt = pd.DataFrame(total)
    tt.columns = ['Total']
    uniques = []
    for col in data.columns:
        unique = data[col].nunique()
        uniques.append(unique)
    tt['Uniques'] = uniques
    return tt
```

```
unique_values(data_2020)
```

Figure 9: Function to find unique values

```
temp = data_2020.groupby(["product_group_name"])["product_type_name"].nunique()
df = pd.DataFrame({'Product Group': temp.index,
                   'Product Types': temp.values
                  })
df = df.sort_values(['Product Types'], ascending=False)
plt.figure(figsize = (8,6))
plt.title('Number of Product Types per each Product Group')
s = sns.barplot(x = 'Product Group', y="Product Types", data=df)
s.set_xticklabels(s.get_xticklabels(),rotation=90)
locs, labels = plt.xticks()
plt.show()
```

Figure 10: Code to display chart

```
purchase_dict = {}

for i in tqdm(df.index):
    cust_id = join_df.at[i, 'customer_id']
    art_id = join_df.at[i, 'article_id']
    t_dat = join_df.at[i, 't_dat']

    if cust_id not in purchase_dict:
        purchase_dict[cust_id] = {}

    if art_id not in purchase_dict[cust_id]:
        purchase_dict[cust_id][art_id] = 0

    x = max(1, (last_ts - t_dat).days)

    a, b, c, d = 2.5e4, 1.5e5, 2e-1, 1e3
    y = a / np.sqrt(x) + b * np.exp(-c*x) - d

    value = join_df.at[i, 'quotient'] * max(0, y)
    purchase_dict[cust_id][art_id] += value
```

Figure 11: Top N purchases

```
purchase_df['timestamp'] = purchase_df.t_dat.values.astype(np.int64) // 10 ** 9
purchase_df.head()
```

Figure 12: Converting date to unix_time

```
temp.to_csv('recbole_data/recbox_data/recbox_data.inter', index=False, sep='\t')
```

Figure 13: Saving data to inter format

```
parameter_dict = {
    #'data_path': "gs://ricstorage/recbox_data",
    'data_path': "recbole_data/",
    'USER_ID_FIELD': 'user_id',
    'ITEM_ID_FIELD': 'item_id',
    'TIME_FIELD': 'timestamp',
    'user_inter_num_interval': "[10,inf)",
    'item_inter_num_interval': "[20,inf)",
    'load_col': {'inter': ['user_id', 'item_id', 'timestamp']},
    'train_neg_sample_args' : None,
    'loss_type': 'CE',
    'neg_sampling': None,
    'epochs':5,
    'eval_args': {
        'split': {'RS': [7, 0, 3]},
        'group_by': 'user',
        'order': 'TO',
        'mode': 'full',
        'metric_decimal_place': 4,
        'metrics': ['Recall', 'MRR', 'MAP','Precision','Hit'],
        'topk': 12
    }
}

config = Config(model='GRU4Rec', dataset='recbox_data', config_dict=parameter_dict) #,
```

Figure 14: Model 1 GRU4REC Config

```
parameter_dict = {
    #'data_path': "gs://ricstorage/recbox_data",
    'data_path': "recbole_data/",
    'USER_ID_FIELD': 'user_id',
    'ITEM_ID_FIELD': 'item_id',
    'TIME_FIELD': 'timestamp',
    'user_inter_num_interval': "[10,inf)",
    'item_inter_num_interval': "[20,inf)",
    'load_col': {'inter': ['user_id', 'item_id', 'timestamp']},
    'train_neg_sample_args' : None,
    'loss_type': 'CE',
    'neg_sampling': None,
    'epochs':5,
    'eval_args': {
        'split': {'RS': [7, 0, 3]},
        'group_by': 'user',
        'order': 'TO',
        'mode': 'full',
        'metric_decimal_place': 4,
        'metrics': ['Recall', 'MRR', 'MAP','Precision','Hit'],
        'topk': 12
    }
}

config = Config(model='BERT4Rec', dataset='recbox_data', config_dict=parameter_dict) #,
```

Figure 15: Model 2 BERT4REC Config

```
parameter_dict = {
    'data_path': 'recbole_data/',
    'USER_ID_FIELD': 'user_id',
    'ITEM_ID_FIELD': 'item_id',
    'TIME_FIELD': 'timestamp',
    'user_inter_num_interval': "[10,inf)",
    'item_inter_num_interval': "[10,inf)",
    'load_col': {'inter': ['user_id', 'item_id', 'timestamp'],
                 'item': ['item_id', 'product_code', 'product_type_no', 'product_group_name', 'graphical_appearan
                      'colour_group_code', 'perceived_colour_value_id', 'perceived_colour_master_id',
                      'department_no', 'index_code', 'index_group_no', 'section_no', 'garment_group_no']
            },
    'selected_features': ['product_code', 'product_type_no', 'product_group_name', 'graphical_appearance_no',
                          'colour_group_code', 'perceived_colour_value_id', 'perceived_colour_master_id',
                          'department_no', 'index_code', 'index_group_no', 'section_no', 'garment_group_no'],
    'neg_sampling': None,
    'train_neg_sample_args' : None,
    'epochs': 10,
    'eval_args': {
        'split': {'RS': [7, 0, 3]},
        'group_by': 'user',
        'order': 'TO',
        'mode': 'full'}
}

config = Config(model='GRU4Rec', dataset='recbox_data', config_dict=parameter_dict)
```

Figure 16: Model 3 GRU4REC with additional attribites Config

```
parameter_dict = {
    'data_path': 'recbole_data/',
    'USER_ID_FIELD': 'user_id',
    'ITEM_ID_FIELD': 'item_id',
    'TIME_FIELD': 'timestamp',
    'user_inter_num_interval': "[10,inf)",
    'item_inter_num_interval': "[10,inf)",
    'load_col': {'inter': ['user_id', 'item_id', 'timestamp'],
                 'item': ['item_id', 'product_code', 'product_type_no', 'product_group_name', 'graphical_appearan
                      'colour_group_code', 'perceived_colour_value_id', 'perceived_colour_master_id',
                      'department_no', 'index_code', 'index_group_no', 'section_no', 'garment_group_no']
             },
    'selected_features': ['product_code', 'product_type_no', 'product_group_name', 'graphical_appearance_no',
                          'colour_group_code', 'perceived_colour_value_id', 'perceived_colour_master_id',
                          'department_no', 'index_code', 'index_group_no', 'section_no', 'garment_group_no'],
    'neg_sampling': None,
    'train_neg_sample_args' : None,
    'epochs': 10,
    'eval_args': {
        'split': {'RS': [7, 0, 3]},
        'group_by': 'user',
        'order': 'TO',
        'mode': 'full'}
}
config = Config(model='BERT4Rec', dataset='recbox_data', config_dict=parameter_dict)
```

Figure 17: Model 4 BERT4REC with additional attributes Config

```
dataset = create_dataset(config)
logger.info(dataset)
```

Figure 18: Create Dataset using Recbole create_Dataset function

```
# dataset splitting
train_data, valid_data, test_data = data_preparation(config, dataset)
```

Figure 19: Dataset Splitting

```
# model loading and initialization
model = GRU4Rec(config, train_data.dataset).to(config['device'])
logger.info(model)
```

Figure 20: Model Initialization: GRU4Rec

```
# model loading and initialization
model = BERT4Rec(config, train_data.dataset).to(config['device'])
logger.info(model)
```

Figure 21: Model Initialization: Bert4Rec

```
from recbole.utils.case_study import full_sort_topk
external_user_ids = dataset.id2token(
    dataset.uid_field, list(range(dataset.user_num)))[1:]#fist element in array is 'PAD'(default of Recbole) ->re
```

```
topk_items = []
for internal_user_id in list(range(dataset.user_num))[1:]:
    _, topk_iid_list = full_sort_topk([internal_user_id], model, test_data, k=12, device=config['device'])
    last_topk_iid_list = topk_iid_list[-1]
    external_item_list = dataset.id2token(dataset.iid_field, last_topk_iid_list.cpu()).tolist()
    topk_items.append(external_item_list)
print(len(topk_items))
```

```
16870
```

```
external_item_str = [' '.join(x) for x in topk_items]
result = pd.DataFrame(external_user_ids, columns=['customer_id'])
result['prediction'] = external_item_str
result.head()
```

Figure 22: Sorting products based on scores

```
# Filter the DataFrame based on the specified customer_id prefix
result_filtered = result[result['customer_id'].str.startswith(desired_customer_id_prefix)].copy()
result_filtered['prediction'] = result_filtered['prediction'].apply(convert_to_integers)
# Explode the prediction column to have one article_id per row
result_filtered_exploded = result_filtered.explode('prediction')
# Merge on article_id
result_filtered_exploded = pd.merge(result_filtered_exploded, data_2020[['article_id','prod_name']], left_on='pre
unique_rows = result_filtered_exploded[['customer_id', 'prediction', 'prod_name']].drop_duplicates()
unique_rows
```

Figure 23: Displaying product name corresponding to article id

```
map_score = calculate_map(result_list, ground_truth_list)
precision, recall = calculate_precision_recall(result_list, ground_truth_list)
print(f"MAP@12 Score: {map_score}")
print(f"Average Precision: {precision}")
print(f"Average Recall: {recall}")
```

Figure 24: Evaluation Metrics