Лабораторная работа №6:

"Разработка системы предсказания поведения на основании графовых моделей"

Цель: обучение работе с графовым типом данных и графовыми нейронными сетями.

Задача: подготовить графовый датасет из базы данных о покупках и построить модель предсказания совершения покупки.

Графовые нейронные сети

Графовые нейронные сети - тип нейронной сети, которая напрямую работает со структурой графа. Типичным применениями GNN являются:

- Классификация узлов;
- Предсказание связей;
- Графовая классификация;
- Распознавание движений;
- Рекомендательные системы.

В данной лабораторной работе будет происходить работа над **графовыми сверточными сетями**. Отличаются они от сверточных нейронных сетей нефиксированной структурой, функция свертки не является .

Подробнее можно прочитать тут: https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

Тут можно почитать современные подходы к использованию графовых сверточных сетей https://paperswithcode.com/method/gcn

Датасет

В качестве базы данных предлагаем использовать датасет о покупках пользователей в одном магазине товаров RecSys Challenge 2015 (https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015).

Скачать датасет можно отсюда: https://drive.google.com/drive/folders/1gtAeXPTj-c0RwVOKreMrZ3bfSmCwl2y-?usp=sharing (lite-версия является облегченной версией исходного датасета, рекомендуем использовать её)

Также рекомендуем загружать данные в виде архива и распаковывать через пакет zipfile или/и скачивать датасет в собственный Google Drive и примонтировать его в колаб.

Установка библиотек, выгрузка исходных датасетов

```
# Slow method of installing pytorch geometric
# !pip install torch_geometric
# !pip install torch_sparse
# !pip install torch_scatter

# Install pytorch geometric
!pip install torch-sparse -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-cluster -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-geometric -f https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
!pip install torch-scatter==2.0.8 -f https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html
```

```
Installing collected packages: torch-sparse
Successfully installed torch-sparse-0.6.13
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Collecting torch-cluster
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu113/torch_cluster-1.6.0-cp37-cp37m
     |████████████████████████████████| 2.5 MB 2.7 MB/s
Installing collected packages: torch-cluster
Successfully installed torch-cluster-1.6.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Collecting torch-spline-conv
  Downloading https://data.pyg.org/whl/torch-1.11.0%2Bcu113/torch_spline_conv-1.2.1-cp37-c
     |████████████████████████████████| 750 kB 2.8 MB/s
Installing collected packages: torch-spline-conv
Successfully installed torch-spline-conv-1.2.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Collecting torch-geometric
  Downloading torch_geometric-2.0.4.tar.gz (407 kB)
     |████████████████████████████████| 407 kB 4.8 MB/s
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from torc
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from torc
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from tc
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from t
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from py
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/p
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packa
Building wheels for collected packages: torch-geometric
  Building wheel for torch-geometric (setup.py) ... done
  Created wheel for torch-geometric: filename=torch_geometric-2.0.4-py3-none-any.whl size=
  Stored in directory: /root/.cache/pip/wheels/18/a6/a4/ca18c3051fcead866fe7b85700ee2240d8
Successfully built torch-geometric
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.0.4
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public
Looking in links: https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html
Collecting torch-scatter==2.0.8
  Downloading torch_scatter-2.0.8.tar.gz (21 kB)
Building wheels for collected packages: torch-scatter
  Building wheel for torch-scatter (setup.py) ... done
  Created wheel for torch-scatter: filename=torch_scatter-2.0.8-cp37-cp37m-linux_x86_64.wh
  Stored in directory: /root/.cache/pip/wheels/96/e4/4e/2bcc6de6a801960aedbca43f7106d268f7
Successfully built torch-scatter
Installing collected packages: torch-scatter
Successfully installed torch-scatter-2.0.8
```

RANDOM_SEED: 42

BASE_DIR: " /content/ "

```
import numpy as np
import pandas as pd
import pickle
import csv
import os

from sklearn.preprocessing import LabelEncoder

import torch

# PyG - PyTorch Geometric
from torch_geometric.data import Data, DataLoader, InMemoryDataset

from tqdm import tqdm


RANDOM_SEED = 42 #@param { type: "integer" }
BASE_DIR = '/content/' #@param { type: "string" }
np.random.seed(RANDOM_SEED)


# Check if CUDA is available for colab
torch.cuda.is_available
```

```
  <function torch.cuda.is_available>
```

```
# Unpack files from zip-file
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
!gdown --folder https://drive.google.com/drive/folders/1uf01kKc9lklZmj1cxzuEDdmz6NCHZAwS?usp=shar

# import zipfile
# with zipfile.ZipFile(BASE_DIR + 'yoochoose-data-lite.zip', 'r') as zip_ref:
#     zip_ref.extractall(BASE_DIR)
```

```
  Mounted at /content/drive
  Retrieving folder list
  Retrieving folder 1yQCCGMYKTct7i2Mc2928jZnTOAAmcHKW yoochoose-data-lite
  Processing file 1-01nNKISsKnJOAeX51W5PXSWQOg7WWPn dataset-README.txt
```

```
Processing file 1uQzJgdkpOrdxSK4BMyUxL95f_LIwmJvK yoochoose-buys-lite.dat
Processing file 1ufKQVH8HiEeUyZo3DFEJCAHcX1Ld3_DS yoochoose-data-lite.zip
Processing file 17kGMxOoIeepXecHFrrC83RaA0ZnFpfsS yoochoose-data.zip
Retrieving folder list completed
Building directory structure
Building directory structure completed
Downloading...
From: https://drive.google.com/uc?id=1-01nNKISsKnJOAeX51W5PXSWQOg7WWPn
To: /content/lab6/yoochoose-data-lite/dataset-README.txt
100% 3.97k/3.97k [00:00<00:00, 13.6MB/s]
Downloading...
From: https://drive.google.com/uc?id=1uQzJgdkpOrdxSK4BMyUxL95f_LIwmJvK
To: /content/lab6/yoochoose-data-lite/yoochoose-buys-lite.dat
100% 10.9M/10.9M [00:00<00:00, 206MB/s]
Downloading...
From: https://drive.google.com/uc?id=1ufKQVH8HiEeUyZo3DFEJCAHcX1Ld3_DS
To: /content/lab6/yoochoose-data-lite.zip
100% 49.8M/49.8M [00:00<00:00, 246MB/s]
Downloading...
From: https://drive.google.com/uc?id=17kGMxOoIeepXecHFrrC83RaA0ZnFpfsS
To: /content/lab6/yoochoose-data.zip
100% 378M/378M [00:01<00:00, 305MB/s]
Download completed
```

Анализ исходных данных

```
# Read dataset of items in store
import os
os.chdir('/content/drive/MyDrive/MMO/MMO/lab6/yoochoose-data-lite')
df = pd.read_csv('yoochoose-clicks-lite.dat')
# df.columns = ['session_id', 'timestamp', 'item_id', 'category']
df.head()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: C
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	session_id	timestamp	item_id	category
0	9	2014-04-06T11:26:24.127Z	214576500	0
1	9	2014-04-06T11:28:54.654Z	214576500	0
2	9	2014-04-06T11:29:13.479Z	214576500	0
3	19	2014-04-01T20:52:12.357Z	214561790	0
4	19	2014-04-01T20:52:13.758Z	214561790	0

```
# Read dataset of purchases
buy_df = pd.read_csv('/content/drive/MyDrive/MMO/MMO/lab6/yoochoose-data-lite/yoochoose-buys-lite
# buy_df.columns = ['session_id', 'timestamp', 'item_id', 'price', 'quantity']
buy_df.head()
```

	session_id	timestamp	item_id	price	quantity
0	420374	2014-04-06T18:44:58.314Z	214537888	12462	1
1	420374	2014-04-06T18:44:58.325Z	214537850	10471	1
2	489758	2014-04-06T09:59:52.422Z	214826955	1360	2
3	489758	2014-04-06T09:59:52.476Z	214826715	732	2

```
# Filter out item session with length < 2
df['valid_session'] = df.session_id.map(df.groupby('session_id')['item_id'].size() > 2)
df = df.loc[df.valid_session].drop('valid_session',axis=1)
df.nunique()
```

```
    session_id    1000000
    timestamp     5557758
    item_id         37644
    category          275
    dtype: int64
```

```
# Randomly sample a couple of them
NUM_SESSIONS = 50000 #@param { type: "integer" }
sampled_session_id = np.random.choice(df.session_id.unique(), NUM_SESSIONS, replace=False)
df = df.loc[df.session_id.isin(sampled_session_id)]
df.nunique()
```

NUM_SESSIONS: 50000

```
    session_id     50000
    timestamp     278442
    item_id        18461
    category         110
    dtype: int64
```

```
# Average length of session
df.groupby('session_id')['item_id'].size().mean()
```

```
    5.56902
```

```
# Encode item and category id in item dataset so that ids will be in range (0,len(d
item_encoder = LabelEncoder()
category_encoder = LabelEncoder()
df['item_id'] = item_encoder.fit_transform(df.item_id)
df['category']= category_encoder.fit_transform(df.category.apply(str))
df.head()
```

	session_id	timestamp	item_id	category
0	9	2014-04-06T11:26:24.127Z	3496	0
1	9	2014-04-06T11:28:54.654Z	3496	0
2	9	2014-04-06T11:29:13.479Z	3496	0
102	171	2014-04-03T17:45:25.575Z	10049	0
103	171	2014-04-03T17:45:33.177Z	10137	0

```
# Encode item and category id in purchase dataset
buy_df = buy_df.loc[buy_df.session_id.isin(df.session_id)]
buy_df['item_id'] = item_encoder.transform(buy_df.item_id)
buy_df.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  This is separate from the ipykernel package so we can avoid doing imports until
```

	session_id	timestamp	item_id	price	quantity
46	489491	2014-04-06T12:41:34.047Z	12633	1046	4
47	489491	2014-04-06T12:41:34.091Z	12634	627	2
61	70353	2014-04-06T10:55:06.086Z	14345	41783	1
62	489671	2014-04-03T15:48:37.392Z	12489	4188	1
63	489671	2014-04-03T15:59:35.495Z	12489	4188	1

```
# Get item dictionary with grouping by session
buy_item_dict = dict(buy_df.groupby('session_id')['item_id'].apply(list))
buy_item_dict
```

```
2300966: [14062, 1121],
2301182: [12974, 10465, 1819],
2301826: [13164, 13167, 13286, 13166],
2302889: [2125],
2303773: [13166, 13164, 13290, 13288, 15921],
2305544: [13164, 13168],
2306951: [13164, 13164, 13286, 13168, 13166, 13167, 13285],
2307524: [13164, 13286, 13167],
2310501: [13168, 8393, 13168],
2311333: [13285, 13165, 13168],
2312773: [13164, 13286, 9699],
2316132: [7877],
2317759: [14951, 7439, 7439, 14951],
2318429: [18460],
2318877: [11221, 12355],
2319826: [13279, 13285, 13164],
2320679: [13108],
2322056: [12869, 13510],
2322739: [3718, 8278, 10962],
2323802: [18460],
2324881: [13279, 10606, 10607],
2325937: [18460],
2326769: [12826, 13832],
2330123: [13603, 8531],
2330138: [13286, 13164],
2331598: [13165, 13285, 13066],
2332823: [12791, 11662, 12362, 11839, 13712],
2339363: [4025],
2343879: [13164, 13285, 13166],
2343936: [9615],
2343972: [13289, 13167, 13290, 13166, 13289, 13166, 13167, 13290],
2344106: [13165, 12871, 10626, 13168],
2344962: [18460],
2346906: [18460],
2349727: [13613, 13637, 263],
2349987: [12768, 13638, 8573, 13637],
2352556: [12205],
2356753: [3790, 13065],
2358446: [13712, 12868, 13293, 12655],
2362019: [15297, 15009],
2362443: [13712, 12362, 14111],
2369971: [13165, 13164, 13166, 13168],
2373271: [2106],
2374763: [6824],
2379251: [13167],
2381423: [11564],
2389753: [13285, 13164, 13166, 13168],
2390563: [372],
2390959: [832],
2391268: [18460, 18460, 18460, 18460, 18460, 18460],
2391993: [13164, 13285],
2397941: [15683],
2399516: [18460, 18460],
2400744: [6813],
2401798: [12630, 13499, 12630, 13499],
2402286: [12815],
2411157: [8253],
...}
```

Сборка выборки для обучения

```
# Transform df into tensor data
def transform_dataset(df, buy_item_dict):
    data_list = []

    # Group by session
    grouped = df.groupby('session_id')
    for session_id, group in tqdm(grouped):
        le = LabelEncoder()
        sess_item_id = le.fit_transform(group.item_id)
        group = group.reset_index(drop=True)
        group['sess_item_id'] = sess_item_id

        #get input features
        node_features = group.loc[group.session_id==session_id,
                                                        ['sess_item_id','item_id','
        node_features = torch.LongTensor(node_features).unsqueeze(1)
        target_nodes = group.sess_item_id.values[1:]
        source_nodes = group.sess_item_id.values[:-1]

        edge_index = torch.tensor([source_nodes,
                                                target_nodes], dtype=torch.long)
        x = node_features

        #get result
        if session_id in buy_item_dict:
            positive_indices = le.transform(buy_item_dict[session_id])
            label = np.zeros(len(node_features))
            label[positive_indices] = 1
        else:
            label = [0] * len(node_features)

        y = torch.FloatTensor(label)

        data = Data(x=x, edge_index=edge_index, y=y)

        data_list.append(data)

    return data_list

# Pytorch class for creating datasets
class YooChooseDataset(InMemoryDataset):
    def __init__(self, root, transform=None, pre_transform=None):
        super(YooChooseDataset, self).__init__(root, transform, pre_transform)
        self.data, self.slices = torch.load(self.processed_paths[0])

    @property
    def raw_file_names(self):
        return []

    @property
    def processed_file_names(self):
        return [BASE_DIR+'yoochoose_click_binary_100000_sess.dataset']

    def download(self):
        pass

    def process(self):
        data_list = transform_dataset(df, buy_item_dict)

        data, slices = self.collate(data_list)
        torch.save((data, slices), self.processed_paths[0])


# Prepare dataset
dataset = YooChooseDataset('./')
```

```
Processing...
  0%|          | 0/50000 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/ipykernel_lau
100%|██████████| 50000/50000 [02:59<00:00, 277.80it/s]
Done!
```

Разделение выборки

```
# train_test_split
dataset = dataset.shuffle()
```

```
one_tenth_length = int(len(dataset) * 0.1)
train_dataset = dataset[:one_tenth_length * 8]
val_dataset = dataset[one_tenth_length*8:one_tenth_length * 9]
test_dataset = dataset[one_tenth_length*9:]
len(train_dataset), len(val_dataset), len(test_dataset)
```

```
(40000, 5000, 5000)
```

```
# Load dataset into PyG loaders
batch_size= 512
train_loader = DataLoader(train_dataset, batch_size=batch_size)
val_loader = DataLoader(val_dataset, batch_size=batch_size)
test_loader = DataLoader(test_dataset, batch_size=batch_size)
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.
  warnings.warn(out)
```

```
# Load dataset into PyG loaders
num_items = df.item_id.max() +1
num_categories = df.category.max()+1
num_items , num_categories
```

```
(18461, 109)
```

Настройка модели для обучения

```
embed_dim = 128
from torch_geometric.nn import GraphConv, TopKPooling, GatedGraphConv, SAGEConv, SGConv
from torch_geometric.nn import global_mean_pool as gap, global_max_pool as gmp
import torch.nn.functional as F

class Net(torch.nn.Module):
        def __init__(self):
                super(Net, self).__init__()
                # Model Structure
                self.conv1 = GraphConv(embed_dim * 2, 128)
                self.pool1 = TopKPooling(128, ratio=0.9)
                self.conv2 = GraphConv(128, 128)
                self.pool2 = TopKPooling(128, ratio=0.9)
                self.conv3 = GraphConv(128, 128)
                self.pool3 = TopKPooling(128, ratio=0.9)
                self.item_embedding = torch.nn.Embedding(num_embeddings=num_items, embedding_dim
                self.category_embedding = torch.nn.Embedding(num_embeddings=num_categories, embe
                self.lin1 = torch.nn.Linear(256, 256)
                self.lin2 = torch.nn.Linear(256, 128)
                self.bn1 = torch.nn.BatchNorm1d(128)
                self.bn2 = torch.nn.BatchNorm1d(64)
                self.act1 = torch.nn.ReLU()
                self.act2 = torch.nn.ReLU()

        # Forward step of a model
```

```
    def forward(self, data):
        x, edge_index, batch = data.x, data.edge_index, data.batch

        item_id = x[:,:,0]
        category = x[:,:,1]


        emb_item = self.item_embedding(item_id).squeeze(1)
        emb_category = self.category_embedding(category).squeeze(1)

        x = torch.cat([emb_item, emb_category], dim=1)
        # print(x.shape)
        x = F.relu(self.conv1(x, edge_index))
        # print(x.shape)
        r = self.pool1(x, edge_index, None, batch)
        # print(r)
        x, edge_index, _, batch, _, _ = self.pool1(x, edge_index, None, batch)
        x1 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = F.relu(self.conv2(x, edge_index))

        x, edge_index, _, batch, _, _ = self.pool2(x, edge_index, None, batch)
        x2 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = F.relu(self.conv3(x, edge_index))

        x, edge_index, _, batch, _, _ = self.pool3(x, edge_index, None, batch)
        x3 = torch.cat([gmp(x, batch), gap(x, batch)], dim=1)

        x = x1 + x2 + x3

        x = self.lin1(x)
        x = self.act1(x)
        x = self.lin2(x)
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.act2(x)

        outputs = []
        for i in range(x.size(0)):
            output = torch.matmul(emb_item[data.batch == i], x[i,:])

            outputs.append(output)

        x = torch.cat(outputs, dim=0)
        x = torch.sigmoid(x)

        return x
```

Обучение нейронной сверточной сети

```
# Enable CUDA computing
device = torch.device('cuda')
```

```
model = Net().to(device)
# Choose optimizer and criterion for learning
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
crit = torch.nn.BCELoss()


# Train function
def train():
        model.train()

        loss_all = 0
        for data in train_loader:
                data = data.to(device)
                optimizer.zero_grad()
                output = model(data)

                label = data.y.to(device)
                loss = crit(output, label)
                loss.backward()
                loss_all += data.num_graphs * loss.item()
                optimizer.step()
        return loss_all / len(train_dataset)


# Evaluate result of a model
from sklearn.metrics import roc_auc_score
def evaluate(loader):
        model.eval()

        predictions = []
        labels = []

        with torch.no_grad():
                for data in loader:

                        data = data.to(device)
                        pred = model(data).detach().cpu().numpy()

                        label = data.y.detach().cpu().numpy()
                        predictions.append(pred)
                        labels.append(label)

        predictions = np.hstack(predictions)
        labels = np.hstack(labels)

        return roc_auc_score(labels, predictions)


# Train a model
NUM_EPOCHS =  5 #@param { type: "integer" }
for epoch in tqdm(range(NUM_EPOCHS)):
        loss = train()
        train_acc = evaluate(train_loader)
        val_acc = evaluate(val_loader)
        test_acc = evaluate(test_loader)
```

NUM_EPOCHS: 5

```
        print('Epoch: {:03d}, Loss: {:.5f}, Train Auc: {:.5f}, Val Auc: {:.5f}, Test Auc
              format(epoch, loss, train_acc, val_acc, test_acc))
```

```
 20%|██        | 1/5 [00:43<02:53, 43.30s/it]Epoch: 000, Loss: 0.69328, Train Auc: 0.51851,
 40%|████      | 2/5 [01:23<02:05, 41.68s/it]Epoch: 001, Loss: 0.51542, Train Auc: 0.5576
 60%|██████    | 3/5 [02:03<01:21, 40.71s/it]Epoch: 002, Loss: 0.41113, Train Auc: 0.59
 80%|████████  | 4/5 [02:42<00:40, 40.16s/it]Epoch: 003, Loss: 0.37213, Train Auc: 0.
100%|██████████| 5/5 [03:21<00:00, 40.39s/it]Epoch: 004, Loss: 0.33969, Train Auc:
```

Проверка результата с помощью примеров

```
# Подход №1 - из датасета
evaluate(DataLoader(test_dataset[40:60], batch_size=10))
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.
  warnings.warn(out)
0.5180722891566265
```

```
# Подход №2 - через создание сессии покупок
test_df = pd.DataFrame([
            [-1, 15219, 0],
            [-1, 15431, 0],
            [-1, 14371, 0],
            [-1, 15745, 0],
            [-2, 14594, 0],
            [-2, 16972, 11],
            [-2, 16943, 0],
            [-3, 17284, 0]
], columns=['session_id', 'item_id', 'category'])

test_data = transform_dataset(test_df, buy_item_dict)
test_data = DataLoader(test_data, batch_size=1)

with torch.no_grad():
        model.eval()
        for data in test_data:
                data = data.to(device)
                pred = model(data).detach().cpu().numpy()

                print(data, pred)
```

```
100%|██████████| 3/3 [00:00<00:00, 211.70it/s]DataBatch(x=[1, 1, 2], edge_index=[2,
DataBatch(x=[3, 1, 2], edge_index=[2, 2], y=[3], batch=[3], ptr=[2]) [0.00142428 0.04503604 0
DataBatch(x=[4, 1, 2], edge_index=[2, 3], y=[4], batch=[4], ptr=[2]) [0.01907495 0.00566141 0

/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.
  warnings.warn(out)
```

✓ 0 秒 完成时间：19:23 ● ×