Лабораторная работа №6:

"Разработка системы предсказания поведения на основании графовых моделей"

Студент: Кривцов Н.А.
Группа: ИУ5-22М

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-c0RwVOKreMrZ3bfSmCwI2y-?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

In [1]:

```
# 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
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-sparse in /usr/local/lib/python3.7/dist-packages (0.6.13)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-sparse) (1.4
.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->tor
ch-sparse) (1.21.6)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-cluster in /usr/local/lib/python3.7/dist-packages (1.6.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-spline-conv in /usr/local/lib/python3.7/dist-packages (1.2.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://pytorch-geometric.com/whl/torch-1.11.0%2Bcu113.html
Requirement already satisfied: torch-geometric in /usr/local/lib/python3.7/dist-packages (2.0.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (
1.21.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from torch-geometric)
(2.11.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from torch-geometric
) (2.23.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from torch-geometric)
(1.3.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from torch-geome
tric) (1.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (
1.4.1)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.7/dist-packages (from torch-geometri
c) (3.0.9)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-geometric) (4
.64.0)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2-
>torch-geometric) (2.0.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from p
andas->torch-geometric) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->tor
ch-geometric) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil
>=2.7.3->pandas->torch-geometric) (1.15.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist
-packages (from requests->torch-geometric) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->t
orch-geometric) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from reque
sts->torch-geometric) (2022.5.18.1)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from reques
ts->torch-geometric) (3.0.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from sci
kit-learn->torch-geometric) (3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-lear
n->torch-geometric) (1.1.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://data.pyg.org/whl/torch-1.11.0%2Bcu113.html
Collecting torch-scatter==2.0.8
  Using cached 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.whl size=322201
6 sha256=fd219fb4f6d94ae12055c88de126b1423740ab32400e1ff75ed5b63ca96d6135
  Stored in directory: /root/.cache/pip/wheels/96/e4/4e/2bcc6de6a801960aedbca43f7106d268f766c3f9f8ab49b
3a5
Successfully built torch-scatter
Installing collected packages: torch-scatter
Successfully installed torch-scatter-2.0.8
```

In [2]:

```
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 = 125 #@param { type: "integer" }
BASE_DIR = '/content/' #@param { type: "string" }
np.random.seed(RANDOM_SEED)
```

In [3]:

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

Out[3]:

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

In [5]:

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

Анализ исходных данных

In [6]:

```
# Read dataset of items in store
df = pd.read_csv(BASE_DIR + '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: Columns (3) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[6]:

	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

In [7]:

```
# Read dataset of purchases
buy_df = pd.read_csv(BASE_DIR + 'yoochoose-buys-lite.dat')
# buy_df.columns = ['session_id', 'timestamp', 'item_id', 'price', 'quantity']
buy_df.head()
```

Out[7]:

	session_id	timestamp	item_id	price	quantity

	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
4	489758	2014-04-06T09:59:52.578Z	214827026	1046	1

In [8]:

```
# 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()
```

Out[8]:

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

In [9]:

```
# 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()
```

Out[9]:

```
session_id     50000
timestamp     279522
item_id        18732
category         103
dtype: int64
```

In [10]:

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

Out[10]:

```
5.5907
```

In [11]:

```
# Encode item and category id in item dataset so that ids will be in range (0,len(df.item.unique()))
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()
```

Out[11]:

	session_id	timestamp	item_id	category
27	26	2014-04-06T16:42:55.741Z	3639	0
28	26	2014-04-06T16:44:58.482Z	11053	0
29	26	2014-04-06T16:45:11.344Z	7533	0
30	26	2014-04-06T16:46:19.569Z	4866	0
105	187	2014-04-02T18:05:22.418Z	2395	0

In [12]:

```
# 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/indexing.
html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[12]:

	session_id	timestamp	item_id	price	quantity
0	420374	2014-04-06T18:44:58.314Z	1193	12462	1
1	420374	2014-04-06T18:44:58.325Z	1186	10471	1
57	396	2014-04-06T17:53:45.147Z	13004	523	1
105	351689	2014-04-03T07:29:02.313Z	12598	2092	1
141	420229	2014-04-02T18:51:54.172Z	14742	1883	2

In [13]:

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

Out[13]:

```
{396: [13004],
 5332: [1284, 9643],
 5717: [11286],
 8427: [694, 694],
 10019: [15409, 11598],
 11527: [11395],
 11718: [13042, 13044, 11952],
 14007: [772, 772],
 14484: [11938, 11940, 11992, 11938, 11940, 11992],
 19094: [5240, 10440, 5247],
 22427: [13029, 13017],
 22908: [15458],
 28667: [10764, 10761],
 28834: [4728, 13008, 1924, 8286, 1910, 1909, 14973],
 33131: [4366, 8255, 8255, 4366],
 35436: [12742, 12490],
 35699: [1668],
 37807: [12819, 12819],
 39237: [12692, 4146],
 41849: [13008],
 46922: [2982, 2971],
 47917: [14973, 4999, 3717],
 51736: [12679, 12819],
 56174: [12666, 10518],
 60584: [4881],
 63126: [12980, 12827],
 63799: [12793],
 64477: [12981],
 64682: [9682, 9683],
 65192: [9006],
 68052: [13053, 12971, 12683],
 69884: [12666],
 72136: [13053, 12971, 13033],
 74741: [15954, 12682],
 78072: [13958],
```

```
78373: [15745, 15729, 8762, 15499],
80051: [12820, 12667, 13908],
83842: [15192],
84542: [10568],
86688: [14973],
88254: [14973],
89226: [14973, 12970, 12961, 12985],
89312: [13004, 1425, 11030, 12853, 9967, 10761, 4, 410],
90297: [5179, 6964, 873],
93282: [15092,
 5752,
 5749,
 426,
 4286,
 4511,
 7000,
 7003,
 5754,
 4522,
 7004,
 4532,
 7097,
 2082,
 2093,
 2083],
94207: [13028, 13317, 13017],
95924: [3249, 13876],
98536: [13070, 13069, 13041, 13881, 13067],
101512: [15302],
101738: [12617, 11943, 11990, 12739, 10662],
105556: [4584, 4585],
105578: [12822],
107523: [6065, 12683],
109338: [13004, 13004, 13315, 10357, 15853],
110593: [7633, 7633],
111192: [8705],
111502: [1810],
115778: [12965, 12967, 13020, 13024, 12966, 12844],
117627: [980],
117927: [12679],
123974: [10375],
126413: [12823, 13011, 12980],
126924: [980],
127178: [11880, 12736],
128031: [9224],
128061: [15356, 15849],
129084: [11953, 11953],
131027: [8601],
133909: [15045, 14880, 10128],
134728: [11939, 10404],
134736: [7147],
135014: [3221],
138967: [13019, 12961],
145959: [13041, 13881, 13067, 13067],
150246: [10546, 11412],
150557: [8705, 8685],
151018: [4996],
151649: [8480, 3856, 11825],
152851: [13970, 13970],
153982: [4996],
154761: [7493, 12828],
155141: [11954],
155747: [12617, 12616, 11757, 12459, 12967, 11635, 12806, 11953, 3038],
158846: [4034],
159064: [842],
163997: [12819],
164679: [12560],
172282: [757, 12752, 12971, 9860],
173691: [13008, 12985],
175897: [11307],
176227: [60, 1883],
177492: [13076, 12817],
185741: [12821, 12734],
186433: [12490, 6618],
187838: [3427],
190934: [12979, 13004, 12979, 13004],
196102: [248],
```

196342: [9578, 3374],
197479: [13067, 13070, 13069, 13041, 13881],
197779: [6214, 2467, 13879],
202982: [11285],
204098: [12966, 12965, 12967, 12968, 12964],
206082: [12981],
207774: [12434, 12692],
211401: [14085, 14085],
213341: [5986, 6755, 12808, 12526],
214854: [12341],
216247: [14878, 5612],
217242: [13315],
217686: [11311],
218844: [4506, 4506],
218862: [15302],
220063: [8646, 8824, 8796, 8797],
220646: [12692, 12692],
223324: [11288],
225208: [11969],
226539: [1958],
229378: [8255],
231444: [5879],
234683: [12980, 10342],
236168: [12829],
236428: [14973],
241449: [12803],
244573: [13033, 8771, 15838, 15959],
246302: [12815, 12815],
251804: [5457],
255326: [13673, 13672, 15215],
256353: [13730],
258637: [1961, 14869, 14626],
262872: [11543],
263691: [13707, 9021],
265263: [13881, 13069, 13020, 13067],
265737: [7493, 7493],
266467: [10544, 12980, 11991, 15215],
267512: [13053],
270216: [13041],
270578: [12670],
273423: [11991, 11941, 11992, 11990],
276147: [12541, 9483, 12630, 10096, 3147, 9338, 9424, 3100, 9409],
276874: [14762],
278534: [12741, 12742, 12526],
280613: [13706],
286332: [1790],
289272: [9092, 9094, 9095],
292608: [8398, 8398],
294352: [12823, 12819, 12823, 12819, 12819, 12823],
295199: [12828],
300213: [12535, 12535],
301831: [15409],
302784: [9521, 8572],
305913: [10676],
307228: [11285],
307623: [4810],
310447: [4505, 12692],
310541: [3416],
310676: [15302],
314784: [11286, 11286, 276],
315249: [13672],
315631: [14157, 4179],
320191: [10507],
324746: [911],
325364: [14095],
330223: [8348, 13004],
338369: [3690],
343699: [12632, 12633],
344271: [11285, 11285],
345242: [10093, 10103],
346742: [11938, 11992, 11938, 11992],
346879: [13004, 12979, 12682, 15954, 12683],
348831: [12861],
349709: [15215],
350736: [15077],
351689: [12598],
352466: [14973],

```
354984: [8282],
355008: [13036, 12547, 11550, 13020, 13008, 12529, 13020, 13042, 12681],
356134: [10716, 12670, 10716, 12670],
357234: [11310, 11310],
371792: [5667],
374406: [12829, 13005],
375712: [12970, 10723],
381501: [5315, 7150],
383181: [12742, 12526],
383853: [12760, 11696, 7481, 11645],
384137: [4912, 4756],
387179: [4810, 4811],
388664: [13069, 13020, 13071, 11305, 6065],
396416: [11071],
401958: [9968, 16],
403286: [10342, 6250],
408041: [12971, 12971],
420229: [14742],
420332: [15744, 6333],
420374: [1193, 1186],
421377: [11321, 11321],
422691: [12738, 11163, 12636],
423222: [12745, 12745],
425119: [8276, 12969, 8276, 12969],
425214: [15215],
426721: [13020, 14626, 13024],
427381: [12970, 11287, 7163, 10072],
431052: [3543, 3542],
437398: [4730, 4226],
438401: [13871],
442227: [12819, 12819],
445898: [13070, 13024, 13881, 13066, 13176, 13043, 14869, 14626, 13067],
446123: [12969, 12971],
446379: [11953],
446584: [10180],
447256: [1999],
449953: [15511,
 15950,
 15379,
 15351,
 15500,
 15352,
 8771,
 15366,
 8790,
 15507,
 15752,
 15743],
451234: [10210, 15049],
452924: [10059],
453208: [980],
455962: [1137],
460141: [12528, 12558, 13013],
460686: [4810, 4811],
460814: [12670, 12804],
461892: [11278, 8572],
462753: [12692, 12680],
463256: [12736, 12994],
463476: [12806, 11503],
467711: [1739, 13035, 1739, 13035],
471204: [2799, 8532, 14000, 8570],
471851: [8255],
472826: [34, 4779, 8361, 9969, 5, 4778],
475899: [12961, 12985, 12752, 11283],
476967: [12979, 13004, 12962, 2733, 12981],
477611: [6061],
480829: [2744],
481199: [13019, 13053],
487722: [9891],
488596: [12688],
491606: [922],
494591: [4505],
495012: [13069, 13070],
495336: [9199],
498628: [8172],
503127: [8172],
505444: [19961,
```

```
509481: [13317, 8935, 11253, 11254, 12967, 11852, 6087],
509978: [3927, 3923],
511618: [13032, 11209],
512691: [15529, 2191, 15529, 2191],
517243: [11320],
519443: [13021],
520334: [13071, 5306],
522853: [11908],
529602: [8380, 9869, 1810],
531309: [2190, 12757, 12757, 12813, 2188, 12863, 12369],
533472: [12803],
533819: [8255, 12752, 8255, 12752],
535199: [12670, 12617, 10662, 13176],
535934: [12819],
536469: [10134],
537563: [12834, 12845, 12829, 15981],
538594: [7112],
540283: [12844, 12834, 4034, 4034, 12844, 12834],
540353: [13073, 13071],
545092: [8718, 15770, 15323],
548388: [12828],
551414: [12833],
553806: [4843, 9006],
555682: [13069, 13315],
556053: [911],
559939: [10387],
561764: [5785, 5784],
565973: [15365, 15367],
572511: [15264, 15409],
573286: [1189],
573426: [8458],
574229: [14882],
576054: [13245, 12965, 13071],
578304: [5910, 13020, 13024],
584607: [12612],
584826: [4811, 4810],
587296: [13176, 12829, 11254, 11055],
587832: [11306],
589002: [12829, 12845],
590188: [11053],
595811: [2003],
596502: [4208],
596624: [12845],
600082: [12825],
601402: [13856],
603458: [8255],
603822: [13176, 13037, 13041, 11594],
604042: [13030, 13017, 7454],
605882: [15841, 15796],
605904: [13229],
607408: [13073],
610674: [12540, 3875, 7813, 12633, 11161],
611642: [12344],
613112: [373],
614178: [4836],
614849: [12602, 2899, 12602, 2899],
616894: [13020, 12737],
620966: [11499, 11286, 12831, 12830],
621842: [10725],
623669: [3889, 13053],
625244: [8648, 15314, 8822, 15806, 15549, 15711, 8736, 8761],
627312: [9869],
628053: [15124],
634447: [5296],
634789: [5289],
636363: [13253, 13250],
637039: [9869],
638118: [5667],
642869: [12981, 12962, 12962, 12981],
643909: [371],
644896: [13032],
648904: [12829],
657701: [11545],
666032: [5296],
667656: [8282],
668204: [3639],
668991: [5427, 9617],
```

669426: [2771, 3078, 8352, 13246],
671531: [10172, 12827, 11741],
674377: [14625, 14998],
675243: [14449],
680172: [710],
682358: [195, 1131],
691586: [11938, 11938],
698267: [12851, 4957, 7750, 5616, 397],
698269: [10489, 213],
702769: [13033, 8352, 8398],
707761: [12819],
711561: [8718, 15308],
712451: [5555, 10076],
713507: [13176, 13037],
713648: [354],
714223: [11940, 11942],
718792: [5934],
721202: [12342, 8352, 1804, 8352, 1804, 12342],
723588: [2013],
725103: [4092, 13071],
730527: [12861, 13073],
734467: [10040],
735254: [12831, 13246],
736131: [13246],
736568: [8499],
739331: [12734, 12821],
739896: [15463],
740752: [4836],
747024: [12670],
752307: [12542],
753064: [15263],
756893: [12670, 13315],
759938: [12831, 12830],
762432: [8398, 8325],
765642: [3416, 3244, 3807, 7246],
765909: [8175],
767184: [11992, 11954, 11937, 11942],
769901: [12833, 12829, 12828],
770473: [11939, 11991, 11990],
779433: [13008],
779677: [13079, 8301, 13176, 11852, 11852, 12526, 12632],
779777: [2209],
782154: [13022, 13020, 12867],
783266: [9199, 9199],
784784: [9630, 10761, 10762],
786893: [12671, 11531, 11416, 12638],
792147: [12828, 12746],
796217: [12692, 12680],
797261: [12830, 12844, 12831],
797672: [12344, 3749, 1746],
801208: [12966, 12965, 12968, 12964],
803313: [8771, 94, 19],
807281: [12829, 12828],
813046: [15879],
817009: [13344, 14966],
817391: [13076],
818841: [12834, 12829, 12834, 12829],
820179: [13470, 13505, 13400],
821093: [11287],
823972: [12811],
828289: [11941, 11991],
832366: [1732, 1727],
841779: [13017],
846562: [11544],
850362: [952],
850763: [13254, 10076, 9111, 13199, 11888, 12342],
852289: [3639, 14038, 3796],
854031: [12962],
855106: [1412],
856169: [3087, 2053],
857413: [15162, 14973, 14973, 15162],
859348: [11682, 11681],
861412: [11689, 11189],
861716: [13068, 8572, 11278, 11993, 5910],
861913: [911],
863796: [11305, 11305, 12844, 11305, 11306, 11305, 12830, 11306],

```
866034: [12819, 12819],
873131: [13069, 13881, 13041, 13070],
873617: [4092],
874742: [8175, 8175],
876848: [13246],
881814: [3498, 7547],
884177: [3808, 3408, 3407],
886141: [13306, 13306],
887613: [10532, 11991],
889281: [13745, 5170, 16044, 3927],
891997: [10076, 12795],
893801: [1211],
901563: [11991, 11941],
902689: [15765, 12819, 15840, 15817],
906101: [12845, 12834],
906464: [11254, 11253, 12833],
910871: [13073, 9596, 13073],
911254: [13871, 13209, 13871, 13209],
914714: [7493, 2191, 7493, 2191],
916696: [6790],
918692: [10040],
920324: [8352],
921976: [12966, 12965],
924742: [3694],
927411: [14998],
930997: [13041, 13070, 3564, 13069, 13176],
931176: [13072, 13073, 10543],
935032: [13071, 12829],
939588: [12829, 12845, 12831, 13066],
939603: [8729, 8734],
944312: [13008, 969],
947223: [12829],
948518: [14600, 10170, 2669, 244],
949759: [8172],
954497: [12570, 14926],
956229: [3927],
956628: [11545, 1907],
958817: [4208],
961416: [12831],
963548: [10563, 7918, 11596, 10546, 7918, 10546, 11596, 10563],
964747: [15734],
965579: [357],
967409: [13864, 13888, 12871, 13887, 12963, 13264],
968462: [561],
972238: [13183],
974711: [5448, 8276, 7918, 5448, 7918, 8276],
976893: [12754],
977862: [13686, 2130, 12831],
978009: [13871],
980199: [13248, 15001, 13743, 13071],
987121: [1988],
991674: [15001, 14998],
992026: [13080],
992956: [991],
995923: [12831, 12830, 2123],
996123: [12864, 12785],
1001102: [4994, 10078, 12786],
1004494: [13080, 12680],
1004773: [10440, 5222, 5247, 5210],
1007119: [11289, 11289],
1007878: [13871],
1009081: [13712],
1010708: [1746, 90],
1016318: [11540],
1018546: [3687],
1019248: [13245],
1020346: [2223],
1021156: [12831, 12829, 12831, 12829],
1021768: [13887, 13264],
1026056: [12829, 8179],
1026246: [13248, 13248],
1027453: [12819],
1028494: [12828, 12831, 16014],
1028974: [16097, 13080, 13591],
1029161: [11286],
1035807: [12845, 15504, 8657],
1036908: [12496]
```

```
1036906: [12496],
1037224: [12788, 15028, 13007, 8822, 13249, 13252],
1040397: [11937, 12830, 15215],
1043108: [9910],
1044328: [12739],
1046943: [11826],
1047291: [10344, 2645, 11834, 2645, 2506],
1049669: [13320, 13180],
1050158: [13707],
1051661: [5612, 13200],
1052752: [2179, 13190],
1053378: [11852, 13887, 11954],
1055831: [12560],
1056069: [3856],
1058621: [153, 8255],
1060999: [12827, 13042, 13043],
1061927: [2175, 2180, 2165, 2178, 2167],
1062786: [13888, 8474, 13864],
1065912: [13887, 12963, 13888, 12960, 13887, 12963, 13888, 12960],
1066542: [7834],
1069409: [13180, 13213, 13320],
1069469: [10681, 9073],
1069832: [4208, 12819],
1070276: [10725, 2123],
1083083: [6856],
1087189: [13180],
1087718: [12828],
1090759: [7246, 12786],
1093468: [13183, 10537, 10227],
1094188: [13320, 13320],
1094657: [10054],
1095159: [12828, 12831],
1101143: [1412],
1103679: [3560],
1105723: [9900],
1106573: [12833],
1108214: [12845, 12834],
1109444: [12831, 12828],
1113626: [14241],
1114579: [15409],
1116413: [13072],
1117458: [12831, 12829],
1120286: [13181],
1125028: [12829, 12831],
1128034: [11540],
1129263: [7382],
1132476: [11287],
1140522: [4150, 4039, 13173, 4150, 4039, 13173],
1141823: [12565],
1146526: [13183],
1147084: [13184, 413, 410, 11],
1148138: [8690,
 8824,
 8792,
 15719,
 15712,
 8814,
 8711,
 8817,
 8718,
 14848,
 8717,
 8792,
 8814,
 8817,
 8690,
 8824,
 15712,
 8711,
 15719,
 8718,
 8717,
 14848],
1150003: [4150, 13250, 13253, 13250, 4150, 13253],
1152353: [13007],
1152839: [12547, 15028, 13248],
1154238: [13743, 12787, 13191],
1154488: [13249, 13252],
```

```
1154488: [13249, 13252],
1154598: [12834, 12834],
1155618: [3007],
1157393: [13071],
1157667: [8550],
1157892: [13034, 13035],
1160709: [4150, 13248],
1164267: [6014],
1169649: [12845],
1172063: [12831, 12829],
1188157: [13180, 12787, 12788],
1191007: [5326],
1192017: [13230, 8566],
1192338: [10409, 8600],
1193279: [16002, 16002, 16002, 16002, 16002, 16002, 16002, 16002, 16002],
1194663: [334],
1195972: [694, 695],
1196533: [6512],
1199526: [4039, 4150],
1201352: [13172, 13172],
1201678: [13250],
1203947: [12788, 13199, 13319],
1204342: [2744, 2185],
1204593: [7248, 8794],
1206364: [2901],
1208187: [12981],
1212379: [11953, 12864],
1213511: [13183, 13250, 13251],
1214317: [4731],
1217261: [13210],
1218789: [13183],
1219619: [13250, 13251],
1220224: [13831, 13833],
1226622: [13320, 13319, 13180],
1226732: [12829],
1230617: [6184],
1233389: [13245, 15187],
1234512: [12742, 12741, 13864],
1240903: [8927, 8923],
1241361: [13872],
1247121: [917, 10660],
1247661: [195],
1248963: [12795],
1250611: [13197],
1254927: [12844, 15117],
1255914: [13183, 13194, 13198, 13745],
1257224: [13071, 10543, 6314],
1262859: [15215, 13042, 3004, 6540, 13670, 1993, 4848, 12496, 13671, 11981],
1266557: [1999, 2003],
1266993: [11071, 3687],
1268616: [11306, 11291, 11291],
1271254: [12344],
1271902: [12830],
1272566: [12834, 12845],
1273913: [42, 42],
1274317: [15301, 15301],
1275036: [13071],
1275722: [13251, 15193],
1279418: [15394, 5781],
1279517: [15320, 8774],
1283747: [10535],
1286217: [553],
1289662: [15302],
1290246: [11354],
1290678: [4843],
1295128: [402],
1295624: [8282, 11597, 10519, 11596, 10517],
1300609: [10562],
1300763: [10725, 10725],
1301071: [12828, 2128],
1302694: [13708],
1304803: [4799],
1305189: [12496, 12550, 12498],
1306784: [13183],
1313179: [13284, 12636, 13287, 13263, 12738, 13285, 13727, 12594, 11853],
1313759: [12872],
1327659: [13887, 12963, 13037],
1329204: [13888, 13206, 13258, 13887],
```

```
1329304: [13888, 12206, 13258, 13887],
1333992: [12087, 12087],
1336061: [1298, 5764],
1340183: [12963, 13887, 13888],
1342307: [11852, 13888],
1343049: [9902],
1344283: [11880, 15308, 8717],
1345186: [12872],
1346724: [13887],
1346867: [4882],
1350464: [13249, 13253],
1353898: [13210, 13210],
1354498: [15262],
1356696: [3927],
1357952: [13183, 13318],
1364034: [13171, 13258, 13887, 12963, 7867],
1368519: [3584],
1372062: [3930, 13168],
1372374: [12097, 10661, 10660, 13258],
1375016: [4927],
1378809: [12842, 13268, 12840],
1379949: [13888, 12963, 13072],
1380237: [13356, 13358],
1384331: [7919],
1384601: [1147, 1147, 1147],
1385634: [13355, 13726, 13171, 13169, 13259],
1386988: [11320, 11320],
1388589: [13198, 9719],
1388838: [5986],
1389754: [13888, 13040, 13064, 13864, 13268, 12954],
1389796: [12842, 13939, 10670],
1393129: [13888, 13887, 11427, 13888, 13887, 11427],
1394752: [13887],
1396122: [13171, 4134],
1398868: [10660],
1400673: [8324],
1400747: [2743],
1402812: [11943],
1404849: [13322],
1405983: [14972],
1406226: [11544, 13171, 13888, 3210, 3209],
1407802: [8705, 13180],
1410363: [11941],
1412014: [4134, 13171, 13265, 13266, 12842],
1415274: [13887, 13888, 12963],
1415611: [10754, 10727, 10727, 10754],
1416611: [13887, 13031],
1417508: [13888, 13887, 13258, 12963, 10661],
1419164: [15302],
1419909: [7002],
1421651: [11756],
1422497: [13180, 3838],
1422669: [13357, 13356, 13930],
1424182: [12982],
1424232: [13229, 2465, 13168],
1424364: [13171, 9070, 12871, 13888],
1424497: [13887, 13888, 4134, 12963],
1442411: [9242],
1444027: [1744, 13255, 8305],
1444364: [13351, 13356, 13358],
1445108: [13265, 10036],
1445544: [13265, 15019],
1450093: [9998, 4411],
1451177: [4134, 12870, 13166],
1451638: [10494],
1455419: [12526, 12789, 11035, 12805, 12461],
1458234: [12865, 12978],
1464933: [9276],
1467307: [15524, 15524],
1467916: [10661, 13887, 13888, 12960, 12963, 12871, 12872, 13264, 13170],
1469162: [12840, 13687],
1471534: [13836, 13065, 13835, 13888, 15993],
1474353: [13253],
1475487: [15192, 15190],
1475723: [15868],
1482714: [7919],
1484224: [11285, 11285],
1491527: [12982],
```

```
1491537: [12982],
1495173: [13888, 12835, 12457],
1495191: [5273],
1496703: [13210],
1497189: [13888, 12963],
1500172: [11412],
1502954: [13178],
1506813: [11762, 11761],
1514746: [13288, 13306],
1515564: [13249],
1518204: [430, 13011, 13888],
1519176: [12963, 13888],
1520349: [8184, 8184],
1523564: [15878],
1524154: [15728, 15947, 15492, 15847, 15509, 8664, 14827],
1529767: [9280],
1530477: [13887, 12963, 13887, 12963],
1531053: [10661, 12963, 11161],
1531199: [13669, 13669, 13669],
1531523: [15193, 13888],
1537016: [10659],
1540439: [13351, 5289, 13358],
1544581: [15751, 8771, 15729],
1547996: [13887, 13888],
1548134: [12819],
1550379: [13885, 13169, 16056, 11849, 11211],
1553861: [13320],
1555116: [13928, 13928],
1558427: [12673],
1561479: [11943],
1564289: [14557],
1566992: [12963, 13264, 13888, 13887, 13864, 12872],
1570407: [12819, 3416],
1572986: [13322],
1573738: [11539],
1575324: [13247, 12824, 13010],
1575922: [13887, 11953, 12963, 13888],
1578594: [13887, 12963],
1594539: [12963, 13887, 13888, 15808, 12872, 12854],
1598274: [13263, 13285, 13887, 13888, 13351, 13304],
1607083: [1203, 3808],
1610983: [13888],
1613338: [13168, 13886, 12824, 13010, 11161],
1613602: [15193],
1614782: [11285],
1618073: [17, 8172, 17, 8172],
1621613: [6241, 7970, 1121],
1622438: [13887, 13888, 12963, 13264, 13258],
1622681: [12845],
1622707: [12496],
1624556: [13177],
1627894: [12963, 13888],
1629933: [12963, 13887, 12871, 13888, 12960, 13864],
1632258: [12982, 9070],
1633193: [8255],
1638032: [13163],
1638806: [13887, 10539, 13888, 13024],
1639036: [7163],
1639317: [10568, 10568],
1639356: [13010, 13010],
1641436: [12340],
1642124: [13008, 13008],
1647363: [6482],
1649091: [4592, 11285, 11285],
1649104: [3830],
1649271: [12577],
1654466: [10568],
1655121: [12819],
1657034: [13887],
1658444: [12963, 13887],
1671547: [13166, 13163],
1676914: [13888, 13887],
1678244: [13887, 13258, 12960, 13264, 7504, 12805],
1678593: [12544, 1016],
1679318: [10660, 10660],
1679728: [15515, 15522],
1683766: [13358],
```

```
1684457: [4793],
1686556: [12976, 2479],
1692104: [7995],
1692851: [13177],
1699412: [13888, 13887],
1700194: [15360, 11761],
1700278: [12788],
1702116: [4134, 13887],
1702311: [1412],
1702871: [14155],
1705713: [12734],
1706839: [13357, 13725],
1711148: [13210, 3923],
1712444: [2932, 12736, 4849],
1715477: [7879],
1717336: [13183],
1718978: [15890, 15890],
1721048: [11030, 5799, 12815],
1723616: [12738, 13284, 12636, 13263],
1724166: [12979],
1727206: [10661],
1730696: [11596, 10095, 11554, 8285, 9073],
1733217: [5555, 153, 5555, 153],
1738896: [13887, 13888, 12963],
1742346: [13639],
1743199: [11039],
1743762: [4127, 13885, 13885, 4127],
1747468: [13196],
1747982: [5296, 6312],
1755202: [13883, 12840],
1755826: [13672],
1759453: [12955],
1759502: [13356],
1768952: [13888, 12963, 13887],
1769151: [12842, 12842],
1772997: [13258],
1776397: [13888, 12824, 13010],
1778003: [11091],
1781848: [13063, 13045, 13887, 13264],
1781978: [11030, 12325],
1783412: [15351],
1784727: [8536, 8536, 8536],
1786429: [4132],
1792084: [12824, 13886],
1793841: [13260, 13287],
1804952: [12959, 14093, 13972, 14094, 13057, 13057],
1808591: [15302],
1816368: [13071],
1817707: [13057, 12590],
1819412: [13307, 13675, 12983],
1820043: [13356],
1826998: [13356, 13356],
1828126: [13362, 327],
1828588: [13530],
1834257: [12529, 12788, 11416, 2520],
1841088: [11943, 11940, 11954, 13670],
1841749: [14870, 15547, 15703, 8734],
1846668: [9639],
1847111: [13885, 13885],
1851602: [13198],
1856142: [13730],
1856419: [13725, 13302, 13351, 13358],
1858184: [3560, 10757, 6093, 425],
1858256: [12974, 12974],
1861698: [13354, 8786, 8731, 8736],
1861751: [12819],
1863653: [13285, 12636, 13263, 13675, 13930, 280, 10663],
1866224: [13726, 13284, 12983, 13259, 13260],
1867351: [13358],
1869922: [13930, 12982],
1870113: [10517, 13529],
1872771: [3721],
1875431: [1810, 8161, 8152],
1879387: [8517, 10672, 13359, 8397, 10054, 8281, 10015, 11412],
1879923: [15834],
1891554: [1901],
1896192: [13357, 12693],
```

```
1897172: [13351, 13304, 13351, 13304],
1897289: [13353, 4730],
1898748: [7921],
1912998: [12983, 2494],
1913067: [12963, 13888, 14575, 8464, 14972],
1913188: [13351, 13358],
1914014: [12692, 12693, 12692, 12693],
1915461: [4506, 4506],
1920706: [9030, 12672],
1926017: [6927, 2189],
1927303: [15993],
1932354: [14155, 11547, 3924],
1933987: [9173, 7750, 9903],
1934593: [8500],
1937743: [4165, 7428],
1937874: [7082],
1944783: [12568, 13180, 4328],
1947126: [12831, 12829],
1947311: [13363, 13359, 15808],
1951211: [3657, 8458],
1957644: [12982],
1957877: [12745],
1960889: [2398, 12344],
1962818: [14155, 13261],
1965099: [10205],
1970227: [5803, 4732],
1976422: [16237],
1978459: [13176, 11246],
2097889: [917, 5142],
2099264: [13351],
2104028: [13306],
2106826: [13888],
2112117: [13888, 13258, 13042, 13042],
2119056: [13358, 13304, 13351],
2121251: [13352, 13353, 12737, 11953],
2123297: [13887, 12963, 12963, 13972, 13864, 13864, 13602],
2124812: [12829],
2126532: [15350, 3969],
2127257: [13673, 13670],
2128477: [13322],
2129241: [11503],
2130643: [12982],
2142503: [15262],
2152178: [15808, 15990, 15955, 15698, 15712],
2152722: [13354, 13356, 13726],
2154411: [12806, 12806],
2164999: [13133],
2169198: [13725, 13356, 13358, 13725, 13356, 13358],
2170198: [12981],
2170227: [5555],
2174786: [13170],
2177054: [9070, 13036, 13174, 13163],
2178962: [13354, 9372],
2185983: [13178],
2187176: [10775],
2192429: [5371, 5437],
2197913: [4894, 5738, 4894, 5738, 4894, 5738],
2201082: [13725],
2201789: [10098],
2202121: [13303, 13290],
2202274: [9030],
2204192: [10206, 11757],
2206498: [15323, 15837, 8718, 6678, 15340, 6678, 15340],
2208068: [12963, 13888],
2210166: [10101, 7918, 13363, 13533],
2211931: [13359],
2215102: [18731, 18731],
2215797: [5282],
2216316: [10129, 9501],
2216646: [12853],
2218893: [16004, 16547],
2219894: [13359, 13361, 13363],
2219914: [13359],
2219934: [10650, 13531],
2225124: [12547, 11161, 11953, 13537, 13057],
2225353: [12837, 10366, 11408, 5017, 10035],
2226053: [13360, 14352, 13363],
```

```
2226227: [12737, 13359, 3082],
2226289: [3721],
2229894: [13351],
2230077: [11954],
2232107: [680, 13083, 16202],
2233632: [13304, 13725, 13351],
2236533: [12976],
2239187: [15543],
2244143: [11283, 10626],
2244483: [13359, 13363, 13360, 13529],
2247088: [13359, 13363, 16004],
2247529: [13359, 13361, 13363],
2247661: [13529, 14153],
2248646: [12963],
2249398: [13529, 13532, 13059, 2407],
2250651: [14351],
2254001: [13359, 13529, 13530, 13361, 13362],
2255089: [13359, 13363, 13362, 13201],
2256081: [3807],
2259243: [16011],
2260331: [13363, 13359],
2261193: [3912, 15837],
2261538: [18731],
2271958: [13730, 55],
2274259: [13885, 13529, 13201],
2274617: [11247],
2277009: [13361, 13359, 10541],
2282693: [192],
2283124: [12496],
2286771: [13363, 300, 13360, 311, 10515],
2287464: [15215],
2290414: [13359, 13363],
2293176: [10673, 13361, 13529, 13359],
2296706: [13730],
2303272: [13359, 13361, 13529],
2309058: [13740, 13311, 13523],
2315361: [1496, 9727],
2317759: [15197, 7427, 7427, 15197],
2322311: [13882, 5142, 7949],
2322958: [13360, 13363],
2323802: [18731],
2324281: [9728, 9728],
2325487: [2965, 2861],
2328751: [13362, 13531, 13359, 13361],
2330694: [379],
2333213: [14351, 13981, 11045],
2336436: [13533, 13530, 13359],
2338316: [13360, 13363],
2339254: [13363, 13363],
2339763: [13360, 13363],
2340277: [248],
2343659: [878],
2343879: [13359, 13529, 13361],
2346614: [13359, 13529, 13359, 13361],
2349568: [18731],
2351931: [11633],
2366599: [13360, 13362, 13361, 13529],
2366986: [3721],
2371387: [13730],
2374387: [13361, 13362, 13534],
2375062: [13359, 13530],
2376401: [13361, 13362, 13360, 13359, 13531],
2379733: [13359, 13361, 13363],
2382834: [13361, 13359, 13959],
2383711: [13882, 4222, 12956],
2384824: [13359, 13530, 4150, 10396, 14093, 13175],
2387622: [13360, 13529, 13359, 4732, 9900],
2389119: [13530, 13361, 13534, 13360, 8397, 3201],
2390324: [13359, 13529, 13537],
2390959: [840],
2392367: [13359, 13530],
2394471: [13359, 13360, 13363, 13529],
2394851: [6678],
2397313: [13959, 11161],
2398109: [11940],
2403466: [11953, 13537, 13353],
2406964: [13363, 13362, 13359],
```

```
 2407859: [13363, 13359],
 2410176: [4097, 12570],
 2410758: [13258, 13972, 12737, 13530, 13727],
 2412556: [16015, 16015, 16015, 16015],
 2416093: [13360, 13363, 10371],
 2420051: [15261, 15261],
 2422761: [16015],
 2425284: [12981, 12962],
 2426211: [18731, 18731],
 2431522: [15993],
 2432049: [15984, 11763],
 2433002: [14351, 18731, 3930],
 ...}
```

Сборка выборки для обучения

In [14]:

```
# 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','category']].sort_values('sess_item_id')[
['item_id','category']].drop_duplicates().values
        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
```

```
        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])
```

In [15]:

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

```
Processing...
  0%|          | 0/50000 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2
1: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider conv
erting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered
internally at  ../torch/csrc/utils/tensor_new.cpp:210.)
100%|██████████| 50000/50000 [02:56<00:00, 283.83it/s]
Done!
```

Разделение выборки

In [16]:

```
# 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)
```

Out[16]:

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

In [17]:

```
# 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.DataLoader
' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

In [18]:

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

Out[18]:

```
(18732, 102)
```

Настройка модели для обучения

In [19]:

```
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=embed_dim)
        self.category_embedding = torch.nn.Embedding(num_embeddings=num_categories, embedding_dim=embed
_dim)
        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

In [20]:

```
# 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()
```

In [21]:

```
# 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)
```

In [22]:

```
# 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)
```

In [23]:

```
# Train a model
NUM_EPOCHS =  15 #@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)
    print('Epoch: {:03d}, Loss: {:.5f}, Train Auc: {:.5f}, Val Auc: {:.5f}, Test Auc: {:.5f}'.
          format(epoch, loss, train_acc, val_acc, test_acc))
```

```
  7%|▋         | 1/15 [00:40<09:25, 40.41s/it]
```

```
Epoch: 000, Loss: 0.67669, Train Auc: 0.52441, Val Auc: 0.53001, Test Auc: 0.52498
```

13%|[redacted] | 2/15 [01:17<08:19, 38.46s/it]

Epoch: 001, Loss: 0.48399, Train Auc: 0.56859, Val Auc: 0.55746, Test Auc: 0.55136

20%|[redacted] | 3/15 [01:54<07:34, 37.86s/it]

Epoch: 002, Loss: 0.40272, Train Auc: 0.60059, Val Auc: 0.56321, Test Auc: 0.56091

27%|[redacted] | 4/15 [02:31<06:54, 37.64s/it]

Epoch: 003, Loss: 0.36485, Train Auc: 0.62781, Val Auc: 0.57539, Test Auc: 0.57249

33%|[redacted] | 5/15 [03:08<06:12, 37.27s/it]

Epoch: 004, Loss: 0.34037, Train Auc: 0.65900, Val Auc: 0.58490, Test Auc: 0.59073

40%|[redacted] | 6/15 [03:44<05:32, 36.99s/it]

Epoch: 005, Loss: 0.32349, Train Auc: 0.68062, Val Auc: 0.58279, Test Auc: 0.58951

47%|[redacted] | 7/15 [04:21<04:54, 36.75s/it]

Epoch: 006, Loss: 0.31069, Train Auc: 0.71464, Val Auc: 0.59796, Test Auc: 0.59507

53%|[redacted] | 8/15 [04:57<04:15, 36.54s/it]

Epoch: 007, Loss: 0.29609, Train Auc: 0.73714, Val Auc: 0.60001, Test Auc: 0.60194

60%|[redacted] | 9/15 [05:33<03:37, 36.32s/it]

Epoch: 008, Loss: 0.28586, Train Auc: 0.77141, Val Auc: 0.60433, Test Auc: 0.61670

67%|[redacted] | 10/15 [06:09<03:01, 36.22s/it]

Epoch: 009, Loss: 0.27251, Train Auc: 0.78842, Val Auc: 0.61848, Test Auc: 0.62342

73%|[redacted] | 11/15 [06:45<02:24, 36.18s/it]

Epoch: 010, Loss: 0.26538, Train Auc: 0.82290, Val Auc: 0.61437, Test Auc: 0.61963

80%|[redacted] | 12/15 [07:21<01:48, 36.14s/it]

Epoch: 011, Loss: 0.24979, Train Auc: 0.85696, Val Auc: 0.61783, Test Auc: 0.62906

87%|[redacted] | 13/15 [07:57<01:12, 36.06s/it]

Epoch: 012, Loss: 0.23585, Train Auc: 0.87090, Val Auc: 0.61519, Test Auc: 0.63079

93%|[redacted] | 14/15 [08:33<00:35, 36.00s/it]

Epoch: 013, Loss: 0.22456, Train Auc: 0.89360, Val Auc: 0.61035, Test Auc: 0.62640

```
100%|██████████| 15/15 [09:09<00:00, 36.63s/it]
```

```
Epoch: 014, Loss: 0.20975, Train Auc: 0.91882, Val Auc: 0.61920, Test Auc: 0.63559
```

Проверка результата с помощью примеров

In [24]:

```
# 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.DataLoader
' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

Out[24]:

```
0.5656565656565656
```

In [25]:

```
# 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, 199.34it/s]
```

```
DataBatch(x=[1, 1, 2], edge_index=[2, 0], y=[1], batch=[1], ptr=[2]) [0.00087506]
DataBatch(x=[3, 1, 2], edge_index=[2, 2], y=[3], batch=[3], ptr=[2]) [0.000564   0.01551606 0.07247568]
DataBatch(x=[4, 1, 2], edge_index=[2, 3], y=[4], batch=[4], ptr=[2]) [2.5984054e-05 1.0836762e-06 1.749
2797e-06 3.5187886e-06]
```

```
/usr/local/lib/python3.7/dist-packages/torch_geometric/deprecation.py:12: UserWarning: 'data.DataLoader
' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```