session-based recommender system

# example code(Main.py)

|  |
| --- |
| *#!/usr/bin/env python36*  *# -\*- coding: utf-8 -\*-*  """  Created on July, 2018  @author: Tangrizzly  """  *import* argparse  *import* pickle  *import* time  *from* utils *import* build\_graph, Data, split\_validation  *from* model *import* \*  parser = argparse.ArgumentParser()  parser.add\_argument('--dataset', *default*='sample', *help*='dataset name: diginetica/yoochoose1\_4/yoochoose1\_64/sample')  parser.add\_argument('--batchSize', *type*=int, *default*=100, *help*='input batch size')  parser.add\_argument('--hiddenSize', *type*=int, *default*=100, *help*='hidden state size')  parser.add\_argument('--epoch', *type*=int, *default*=30, *help*='the number of epochs to train for')  parser.add\_argument('--lr', *type*=float, *default*=0.001, *help*='learning rate')  *# [0.001, 0.0005, 0.0001]*  parser.add\_argument('--lr\_dc', *type*=float, *default*=0.1, *help*='learning rate decay rate')  parser.add\_argument('--lr\_dc\_step', *type*=int, *default*=3, *help*='the number of steps after which the learning rate decay')  parser.add\_argument('--l2', *type*=float, *default*=1e-5, *help*='l2 penalty')  *# [0.001, 0.0005, 0.0001, 0.00005, 0.00001]*  parser.add\_argument('--step', *type*=int, *default*=1, *help*='gnn propogation steps')  parser.add\_argument('--patience', *type*=int, *default*=10, *help*='the number of epoch to wait before early stop ')  parser.add\_argument('--nonhybrid', *action*='store\_true', *help*='only use the global preference to predict')  parser.add\_argument('--validation', *action*='store\_true', *help*='validation')  parser.add\_argument('--valid\_portion', *type*=float, *default*=0.1, *help*='split the portion of training set as validation set')  opt = parser.parse\_args()  print(opt)  def main():      train\_data = pickle.load(open('../datasets/' + opt.dataset + '/train.txt', 'rb'))  *if* opt.validation:          train\_data, valid\_data = split\_validation(train\_data, opt.valid\_portion)          test\_data = valid\_data  *else*:          test\_data = pickle.load(open('../datasets/' + opt.dataset + '/test.txt', 'rb'))  *# all\_train\_seq = pickle.load(open('../datasets/' + opt.dataset + '/all\_train\_seq.txt', 'rb'))*  *# g = build\_graph(all\_train\_seq)*      train\_data = Data(train\_data, *shuffle*=True)      test\_data = Data(test\_data, *shuffle*=False)  *# del all\_train\_seq, g*  *if* opt.dataset == 'diginetica':          n\_node = 43098  *elif* opt.dataset == 'yoochoose1\_64' *or* opt.dataset == 'yoochoose1\_4':          n\_node = 37484  *else*:          n\_node = 310      model = trans\_to\_cuda(SessionGraph(opt, n\_node))      start = time.time()      best\_result = [0, 0]      best\_epoch = [0, 0]      bad\_counter = 0  *for* epoch *in* range(opt.epoch):          print('-------------------------------------------------------')          print('epoch: ', epoch)          hit, mrr = train\_test(model, train\_data, test\_data)          flag = 0  *if* hit >= best\_result[0]:              best\_result[0] = hit              best\_epoch[0] = epoch              flag = 1  *if* mrr >= best\_result[1]:              best\_result[1] = mrr              best\_epoch[1] = epoch              flag = 1          print('Best Result:')          print('\tRecall@20:\t%.4f\tMMR@20:\t%.4f\tEpoch:\t%d,\t%d'% (best\_result[0], best\_result[1], best\_epoch[0], best\_epoch[1]))          bad\_counter += 1 - flag  *if* bad\_counter >= opt.patience:  *break*      print('-------------------------------------------------------')      end = time.time()      print("Run time: %f s" % (end - start))  *if* \_\_name\_\_ == '\_\_main\_\_':      main() |

# testing result

|  |
| --- |
| epoch: 25  start training: 2022-02-13 22:28:02.557892  [0/13] Loss: 5.2257  [3/13] Loss: 5.3453  [6/13] Loss: 5.1560  [9/13] Loss: 5.3873  [12/13] Loss: 5.1533  Loss: 67.961  start predicting: 2022-02-13 22:28:03.031626  Best Result:  Recall@20: 7.0707 MMR@20:  1.3564 Epoch: 25, 25  -------------------------------------------------------  epoch: 26  start training: 2022-02-13 22:28:03.050575  [0/13] Loss: 5.1157  [3/13] Loss: 5.3555  [6/13] Loss: 5.2118  [9/13] Loss: 5.2416  [12/13] Loss: 5.6071  Loss: 68.393  start predicting: 2022-02-13 22:28:03.518323  Best Result:  Recall@20: 7.0707 MMR@20:  1.3564 Epoch: 26, 26  -------------------------------------------------------  epoch: 27  start training: 2022-02-13 22:28:03.535277  [0/13] Loss: 5.1427  [3/13] Loss: 5.0680  [6/13] Loss: 5.1106  [9/13] Loss: 5.2602  [12/13] Loss: 5.5889  Loss: 68.375  start predicting: 2022-02-13 22:28:04.015992  Best Result:  Recall@20: 7.0707 MMR@20:  1.3564 Epoch: 27, 27  -------------------------------------------------------  epoch: 28  start training: 2022-02-13 22:28:04.035939  [0/13] Loss: 5.3350  [3/13] Loss: 5.2369  [6/13] Loss: 5.3851  [9/13] Loss: 5.2038  [12/13] Loss: 4.7483  Loss: 67.577  start predicting: 2022-02-13 22:28:04.486733  Best Result:  Recall@20: 7.0707 MMR@20:  1.3564 Epoch: 28, 28  -------------------------------------------------------  epoch: 29  start training: 2022-02-13 22:28:04.507677  [0/13] Loss: 5.1630  [3/13] Loss: 5.2649  [6/13] Loss: 5.2858  [9/13] Loss: 5.3507  [12/13] Loss: 5.4910  Loss: 68.282  start predicting: 2022-02-13 22:28:04.988390  Best Result:  Recall@20: 7.0707 MMR@20:  1.3564 Epoch: 29, 29  -------------------------------------------------------  Run time: 15.415760 s |