session-based recommender system

# example code(Main.py)

|  |
| --- |
| *from* model.MKM\_SR *import* MKM\_SR  *from* model.utils *import* trans\_to\_cuda,MKM\_DATA,train\_predict\_mkm  *import* pandas *as* pd  *import* time  *import* pickle  *from* datetime *import* datetime  *import* argparse  def get\_n\_entity\_relation\_item(*dataset*,*opt*):      entity2id = pd.read\_csv(*dataset*+'entity2id',*sep*='\t')      item2id =entity2id[entity2id['type']=='item']  *if* *opt*.mode =='MKM\_SR':          relation2id = pd.read\_csv(*dataset*+'relation\_operation2id',*sep*='\t')      n\_entity = len(entity2id)+1      n\_relation = len(relation2id)+1      n\_item = len(item2id)+1  *return* n\_entity,n\_relation,n\_item    *if* \_\_name\_\_ == '\_\_main\_\_':  *# 인자값을 받을 수 있는 인스턴스 생성*      parser = argparse.ArgumentParser()  *# 입력받을 인자값 등록*      parser.add\_argument('--dataset', *default*='demo', *help*='dataset name:jdata2018/kkbox/demo')      parser.add\_argument('--batch\_size', *type*=int, *default*=128, *help*='input batch size')      parser.add\_argument('--hidden\_size', *type*=int, *default*=100, *help*='hidden state size')      parser.add\_argument('--epoch', *type*=int, *default*=50, *help*='the number of epochs to train for')      parser.add\_argument('--kg\_loss\_rate',*type*=float,*default*=0.001,*help*=' the rate of kg\_loss')      parser.add\_argument('--lr', *type*=float, *default*=0.001, *help*='learning rate')  *# [0.001, 0.0005, 0.0001]*      parser.add\_argument('--l2', *type*=float, *default*=1e-3, *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('--remove\_new\_items',*action*='store\_true',*help*='whether remove new items:if add argument,it will be true') *#if add argument,it will be true,no new item*      parser.add\_argument('--mode',*default*='MKM\_SR',*help*='which mode:MKM\_SR')  *# 입력받을 인자값을 opt에 저장 (type: namespace)*      opt = parser.parse\_args(['--remove\_new\_items'])  *# 입력받은 인자값 출력*      print('the mode is :',opt.mode)      print("the dataset is :",opt.dataset)      print('remove new item',opt.remove\_new\_items)  *if* opt.dataset == 'kkbox':          dataset = './data/kkbox/'  *elif* opt.dataset == 'jdata2018':          dataset = './data/jdata2018/'  *elif* opt.dataset == 'demo':          dataset = './data/demo/'  *if* opt.remove\_new\_items:          dataset += 'no\_new\_item/'  *else*:          dataset += 'with\_new\_item/'  *if* opt.mode == 'MKM\_SR':          print('the mode is :', opt.mode)          data\_path = dataset + 'MKM\_SR/'          kg2id = pd.read\_csv(dataset + 'kg2id', *sep*='\t')          item\_ids = kg2id['head'].unique()          print('len(kg items):', len(item\_ids))          itemid2index = {}  *for* index, item\_id *in* enumerate(item\_ids):              itemid2index[item\_id] = index          train\_processed = MKM\_DATA(pickle.load(open(data\_path + 'train\_processed.pkl', 'rb')))          test\_processed = MKM\_DATA(pickle.load(open(data\_path + 'test\_processed.pkl', 'rb')))          n\_entity, n\_relation, n\_item = get\_n\_entity\_relation\_item(dataset,opt)          print('n\_entity:{}, n\_relation:{}, n\_item:{}'.format(n\_entity, n\_relation, n\_item))          model = trans\_to\_cuda(MKM\_SR(opt, n\_entity, n\_relation, n\_item))          start = time.time()          print('start time: ', datetime.now())          best\_result = [0, 0]          best\_epoch = [0, 0]          bad\_counter = 0  *for* epoch *in* range(opt.epoch):              epoch\_start\_time = time.time()              print('-------------------------------------------------------')              print('epoch: ' + str(epoch))              hit, mrr = train\_predict\_mkm(model, train\_processed, test\_processed, item\_ids, itemid2index)              print('the epoch \tRecall@20:\t%.4f\tMMR@20:\t%.4f\t' % (hit, mrr))              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              print('the single epoch time is :%d s' % (time.time() - epoch\_start\_time))  *if* bad\_counter >= opt.patience:  *break*          print('-------------------------------------------------------')          print("now time: {} ,running time:{}".format(datetime.now(), time.time() - start)) |

# testing result

|  |
| --- |
| epoch: 26  [0/165] Loss: 1.7602 2022-02-13 13:22:24.0  [100/165] Loss: 2.2120 2022-02-13 13:22:30  Loss: 375.299  start predicting: 2022-02-13 13:22:34.670  the epoch Recall@20: 17.2330 MM  Best Result:  Recall@20: 18.2646 MMR@20: 7.  the single epoch time is :11 s  ------------------------------------------  epoch: 27  [0/165] Loss: 1.6613 2022-02-13 13:22:35.1  [100/165] Loss: 2.0937 2022-02-13 13:22:41  Loss: 340.263  start predicting: 2022-02-13 13:22:46.224  the epoch Recall@20: 16.3228 MM  Best Result:  Recall@20: 18.2646 MMR@20: 7.  the single epoch time is :11 s  ------------------------------------------  epoch: 28  [0/165] Loss: 1.3601 2022-02-13 13:22:46.7  [100/165] Loss: 1.8730 2022-02-13 13:22:53  Loss: 306.053  start predicting: 2022-02-13 13:22:57.365  the epoch Recall@20: 16.2015 MM  04.827887  Loss: 277.073  start predicting: 2022-02-13 13:23:09.035639  the epoch Recall@20: 16.3835 MMR@20: 6.5698  Best Result:  Recall@20: 18.2646 MMR@20: 7.3558 Epoch: 23, 23  the single epoch time is :11 s |