[{"metadata":{"trusted":true},"cell\_type":"code","source":"# Cosine similarity!\n# user-user recc for rating with cold case baselines and return avg for 1 user-resub\nfrom collections import defaultdict\nimport math\nimport numpy as np\nfrom collections import Counter\ndataWimg = list()\nfor d in data:\n if (d['#image\_id']):\n if (d['number\_of\_upvotes']):\n if (d['total\_votes']):\n dataWimg.append(d)\n \ndef returnRating(upvotes, totVotes):\n if(int(totVotes)==0):\n return 0\n return int(upvotes)/int(totVotes)\n \nX = [(d['username'],d['#image\_id']) for d in dataWimg]\ny = [returnRating(d['number\_of\_upvotes'],d['total\_votes']) for d in dataWimg]\n\n\nhalfLenData = len(dataWimg)/2\n# print(halfLenData)\n\ntrnData = dataWimg [:66153]\nX\_train = X[:66153]\ny\_train = y[:66153]\n\nX\_test = X[66153:]\ny\_test = y[66153:]\n\nusersImageSubs = defaultdict(list)\nimageUserSubs = defaultdict(list)\n\nfor d in trnData:\n user = d['username']\n img = d['#image\_id']\n usersImageSubs[user].append(img)\n imageUserSubs[img].append(user)\n \nuserUserDict = defaultdict(list) \nuserSim = defaultdict(dict) \nfor user in usersImageSubs:\n for img in usersImageSubs[user]:\n userUserDict[user]+= [u for u in imageUserSubs[img] if ((u !=user) & (u!=''))]\n userSim[user] = Counter(userUserDict[user])\n \ndef printFirst10(arr):\n counter = 0\n for a in arr:\n print(a)\n counter+=1\n if(counter>10):\n break\n \n\n# printFirst10(userJaccard.items())\n\nratings = defaultdict(dict)\nuserCount = defaultdict(int)\nuserTotRating = defaultdict(int)\nimageCount = defaultdict(int)\nimageTotRating = defaultdict(int)\ntotRating =0\nusers = set()\nimages = set()\nfor d in trnData:\n ratings[d['username']][d['#image\_id']] = returnRating(d['number\_of\_upvotes'],d['total\_votes'])\n users.add(d['username'])\n images.add(d['#image\_id'])\n imageCount[d['#image\_id']] +=1 \n imageTotRating[d['#image\_id']] += returnRating(d['number\_of\_upvotes'],d['total\_votes'])\n userCount[d['username']] +=1 \n userTotRating[d['username']] += returnRating(d['number\_of\_upvotes'],d['total\_votes'])\n totRating+=returnRating(d['number\_of\_upvotes'],d['total\_votes'])\n \navgRating = totRating/len(trnData)\n \ncounterrr= 0\n \nuserCosine = defaultdict(dict)\nfor user in userSim:\n if(user):\n for u2 in userSim[user]:\n dotProd = 0\n if(u2):\n for i in usersImageSubs[user]:\n if (i in usersImageSubs[u2]):\n dotProd+= ratings[user][i]\*ratings[u2][i]# should be values of that image sim mult together\n counterrr+=1\n userCosine[user][u2] = dotProd/(math.sqrt(len(usersImageSubs[user]))\*math.sqrt(len(usersImageSubs[u2])))\n \n \n # if user not seen before, return img average \n# Calculate img averages\n\navgImageRating = defaultdict(int)\nfor i in imageTotRating:\n avgImageRating[i] = imageTotRating[i]/imageCount[i]\n \navgUserRating = defaultdict(int)\nfor i in userTotRating:\n avgUserRating[i] = userTotRating[i]/userCount[i]\n \n\n# if img not seen before, return user average\n\n# if none seen before, return overall average (for users with 1 post)\n\ndef weightedUserSimScore(user, img):\n rating = 0\n if((user in users) & (img in images)):\n weightedSum = 0\n sumSim = 0\n for (u,v) in userCosine[user].items():\n if(img in usersImageSubs[u]):\n weightedSum+= ratings[u][img]\*v\n sumSim+=v\n if(sumSim==0):\n sumSim = 1\n return (weightedSum/sumSim)\n elif (img in images):\n return avgImageRating[img]\n elif (user in users):\n return avgUserRating[user]\n else:\n return avgRating\n\npredictionsTrn = [weightedUserSimScore(user,img) for (user,img) in X\_train]\n\npredictionsTst = [weightedUserSimScore(user,img) for (user,img) in X\_test]\n\n# print(\"trn data pred\")\n# printFirst10(predictionsTrn)\n# print(\"trn data actual\")\n# printFirst10(y\_train)\n# print(\"test data pred\")\n# printFirst10(predictionsTst)\n# print(\"test data actual\")\n# printFirst10(y\_test)\n\nMSELinRegTrn = np.square(np.subtract(y\_train,predictionsTrn)).mean() \nMSELinRegTst = np.square(np.subtract(y\_test,predictionsTst)).mean() \nprint(\"User similarity Ratings (with cold case avg baseline): \")\nprint(\"MSE for Training \" +str(MSELinRegTrn))\nprint(\"MSE for Test \"+str(MSELinRegTst))","execution\_count":62,"outputs":[{"output\_type":"stream","text":"User similarity Ratings (with cold case avg baseline): \nMSE for Training 0.07363918536066838\nMSE for Test 0.028245667746484827\n","name":"stdout"}]}]