Upvotes Survey Code: EM Algorithm

```
#EM algorithm works in conjunction with worker quality. We still need to
implement tracking of the qualities of workers that answer HITs, and this
is a rudimentary implementation of the EM algorithm on the data.
def matrix_conf(actual, predicted, normalize = True):
  #confusion matrix
   dot = lambda a, b: sum([i*j for (i, j) in zip(a, b)])
   TP = dot(actual,predicted)
   TN = dot([1- x for x in actual], [1- x for x in predicted])
   FP = dot(actual, [1- x for x in predicted])
   FN = dot([1- x for x in actual],predicted)
   firstrow = [TP, FP]
   sec_row = [FN, TN]

   if not normalize:
       return([firstrow,sec_row])

   firstrow = [float(i)/sum(firstrow) for i in firstrow]
   sec_row = [float(i)/sum(sec_row) for i in sec_row]
   return([firstrow,sec_row])
def df_from_data(rows):
   #takes in the rows of results read from CSV
   # dataframe that holds labels by workers
   frame = pd.read_csv(rows, delimiter = "\t", header=None)
   labels_by_workers = [[],[],[],[],[]]
   key = 0
   for i in range(frame.shape[0]):
       if i % 5 == 0:
           key += 1
       labels_by_workers[key - 1].append(frame[2][i])

   data = {
       'worker1' : labels_by_workers[0],
       'worker2' : labels_by_workers[1],
       'worker3' : labels_by_workers[2],
       'worker4' : labels_by_workers[3],
       'worker5' : labels_by_workers[4]
   }
```

```
locations = ["Salt Lake City","Topeka", "McKean County","Jackson","New
York City"]

labels_by_workers = pd.DataFrame(data=data, index=urls)
   labels_by_workers[labels_by_workers=='yes'] = 1
   labels_by_workers[labels_by_workers=='no'] = 0
   return labels_by_workers
def correctLabels_translation(labels_by_workers):
   #return: df containing labels by workers
   #initialize correctLabels

   locations = ["Salt Lake City","Topeka", "McKean County","Jackson","New
York City"]
   correctLabels = pd.DataFrame(data={},index=locations)
   correctLabels['yes'] = labels_by_workers.sum(axis = 1)
   correctLabels['no'] = 5 - correctLabels['yes']
   #calculate max votes and initialize true labels
   correctLabels = (correctLabels.T ==
correctLabels.T.max()).T.astype(int)
   correctLabels = correctLabels.astype('float32') #float conversion
   return correctLabels
def function_iteration(iters):
   #similar to em_vote(rows,iternum) iters= iternum
   #dataframe from dataset
   labels_by_workers = df_from_data(data)
   # max vote to initialize true labels
   correctLabels =
correctLabels_translation(labels_by_workers=labels_by_workers)
   # confusion matrices
   conf_matr_workers= [[],[],[],[],[]]

   # number iterations set to 3
   for i in range(iters):
       # update confusion_matrices pending completed 'true' labels
       conf_matr_workers[0] =
matrix_conf(correctLabels['yes'].values.tolist(),labels_by_workers['worker
1'].values.tolist())
       conf_matr_workers[1] =
matrix_conf(correctLabels['yes'].values.tolist(),labels_by_workers['worker
2'].values.tolist())
```

```
        conf_matr_workers[2] =
matrix_conf(correctLabels['yes'].values.tolist(),labels_by_workers['worker
3'].values.tolist())
        conf_matr_workers[3] =
matrix_conf(correctLabels['yes'].values.tolist(),labels_by_workers['worker
4'].values.tolist())
        conf_matr_workers[4] =
matrix_conf(correctLabels['yes'].values.tolist(),labels_by_workers['worker
5'].values.tolist())
       # print(conf_matr_workers)
        # update correctLabels (accord to weighted majority vote)

        #'yes' column
        cumul_count = 0
        for i in range(5): # num of urls
            for j in range(5): # num of workers
                cumul_count += conf_matr_workers[j][0][1 -
labels_by_workers.iat[i,j]]
            correctLabels['yes'][i] = cumul_count
            cumul_count = 0

        # 'no' column
        cumul_count = 0
        for i in range(5): # num of urls
            for j in range(5): # num of workers
                cumul_count += conf_matr_workers[j][1][1 -
labels_by_workers.iat[i,j]]
            correctLabels['no'][i] = cumul_count
            cumul_count = 0

        # normalize each row by sum of labels
        correctLabels = correctLabels.div(correctLabels.sum(axis=1),
axis=0)
        correctLabels = (correctLabels.T ==
correctLabels.T.max()).T.astype(int) # max value updated with 1 and min
with 0
        correctLabels = correctLabels.astype('float32') # float conversion

   correctLabels['labels'] = correctLabels.idxmax(axis='columns')
   truest_labels = correctLabels['labels'].sort_index()
```

```
    df = truest_labels.to_frame()
    return df
def main():

aggregate = function_iteration(iters)
    aggregate.to_csv(csv_file)
```