Question 1 (higher similarity ==> lower distance)

1. distance = 1-similarity
2. distance = 1/similarity
3. distance = -1*similarity

In [79]:

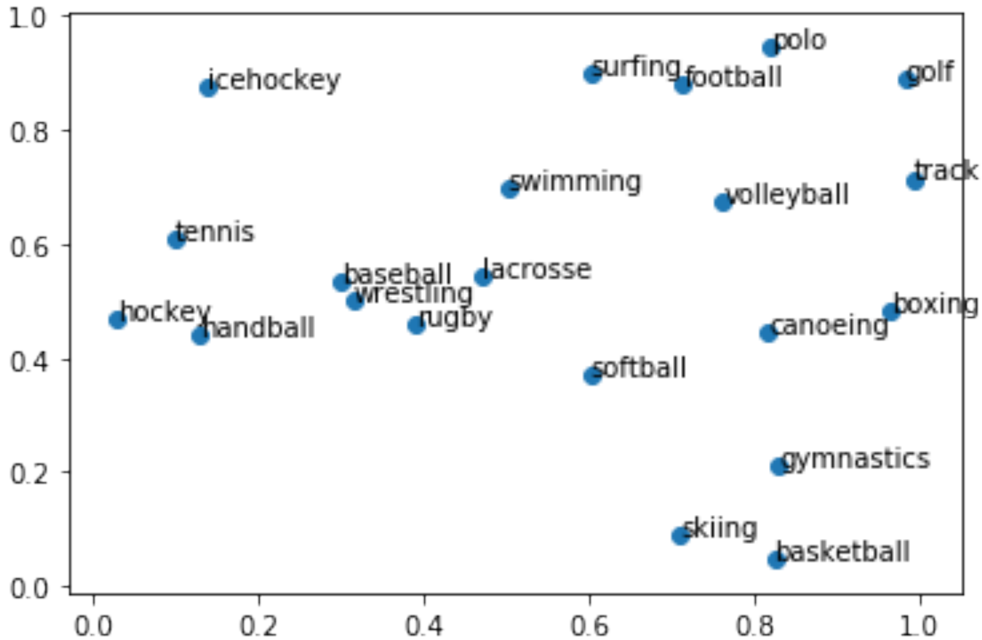
```
#Question 2

import numpy as np
import matplotlib.pyplot as plt

#to compute partial derivatives
delta = 0.005
scaling = 0.01

#other needed variables
gradient = []
stress = 0

#read in similarities
psychsimilarities = np.loadtxt(open("Assignment5-similarities.cs
v", "rb"), delimiter=",", skiprows=1)

#assign random values to the starting points on MDS grid
mdsdistances = np.random.uniform(0, 1, size=(21,2))

#matrix ==> array of MDS locations (x,y coordinates)
#[(football)[0.03,93.2], (soccer)[4,6.78], (basketball)[3.3,4.1]
]

#similarities ==> array from our data --> psychological distance
s
#* = startingsport
#[(football)[football-football, *football-soccer, football-bball
, football-golf],
#(soccer)[football-soccer, soccer-soccer, *soccer-bball, soccer-
golf]],
#(bball)[football-bball, bball-soccer, bball-bball, *bball-golf]
```

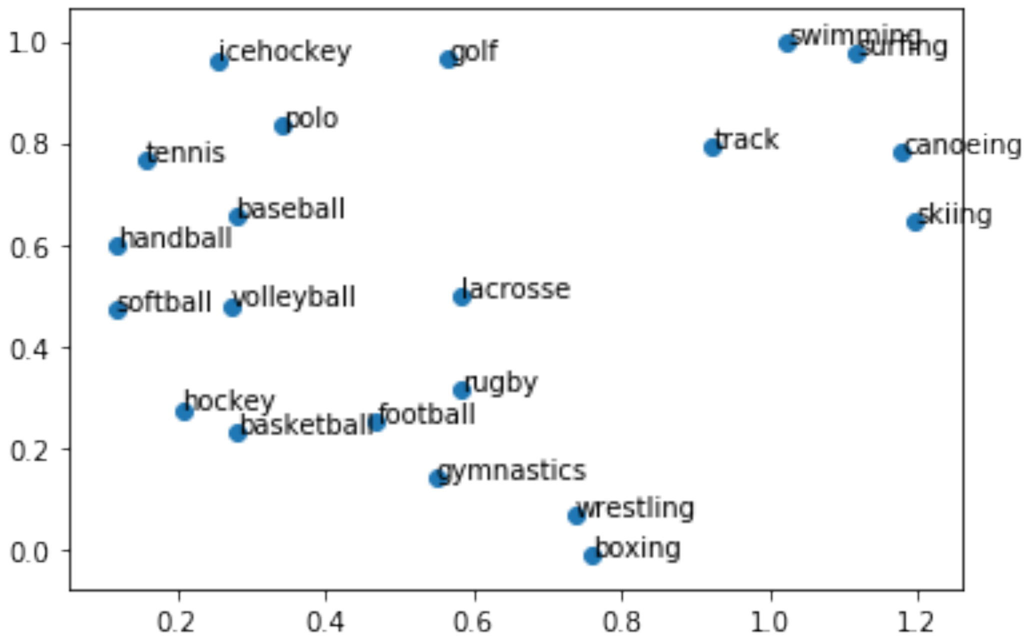
```
def stresscalc(mdsdistances, psychsimilarities):
    stress = 0
    mdsdistance = 0
    psychdistance = 0
    row = 0
    col = 1
    for row in range(0,20): #for each sport
        #compare sport to each remaining sport
        c = col
        for c in range(c,20):
            psychdistance = 1-psychsimilarities[row][c]
            mdsdistance = np.sqrt(np.square(mdsdistances[row][0]
- mdsdistances[c][0]) + np.square(mdsdistances[row][1] - mdsdist
ances[c][1]))
            stress += np.square(psychdistance-mdsdistance)
        col += 1
    return stress

#QUESTION 3
def findgradient(mdsdistances, psychsimilarities, delta):
    gradient= np.copy(mdsdistances)

    #Find the partial derivative by changing x and keeping y & c
hanging y and keeping x constant.
    for i in np.arange(len(mdsdistances) -1): #for all coordinat
es in our grid

        #FIRST SOLVE FOR X
        coordinatesmodplus = np.copy(mdsdistances)
        coordinatesmodminus = np.copy(mdsdistances)

        coordinatesmodplus[i][0]+= delta #changing x value, and
keeping y the same
        fpdx = stresscalc(coordinatesmodplus, psychsimilarities)
#f(changed concept x, y remains)

        coordinatesmodminus[i][0]-=delta #changing x value, keep
ing y the same
        fmdx=stresscalc(coordinatesmodminus, psychsimilarities)
#f(changed concept x, y remains)

        partialderivativex=(fpdx-fmdx)/(2*delta)

        gradient[i][0]=partialderivativex #update that part of t
```

```
he gradient with the partial derivative found for element i of v
alues

        #THEN SOLVE FOR Y
        coordinatesmodplus = np.copy(mdsdistances)
        coordinatesmodminus = np.copy(mdsdistances)

        coordinatesmodplus[i][1]+= delta #changing y value, and
keeping x the same
        fpdy = stresscalc(coordinatesmodplus, psychsimilarities)
#f(changed concept y, x remains)

        coordinatesmodminus[i][1]-=delta #changing y value, keep
ing x the same
        fmdy=stresscalc(coordinatesmodminus, psychsimilarities)
#f(changed concept y, x remains)

        partialderivativey=(fpdy-fmdy)/(2*delta)

        gradient[i][1]=partialderivativey #update that part of t
he gradient with the partial derivative found for element i of v
alues

    return gradient

# def gradientcheck(gradient):
#     passed = []
#     for i in range(len(gradient)):
#         if gradient[i][0] <= 0.1:
#             passed.append(1)
#         if gradient[i][1] <= 0.1:
#             passed.append(1)
#     return sum(passed)

#QUESTION 4
names = ['football', 'baseball', 'basketball', 'tennis',
         'softball', 'canoeing', 'handball', 'rugby',
         'hockey', 'icehockey', 'swimming', 'track',
         'boxing', 'volleyball', 'lacrosse', 'skiing',
         'golf', 'polo', 'surfing', 'wrestling', 'gymnastics']

xpoints = []
ypoints = []
stressvals = []
iterations= 0
```

```
#PLOT RANDOM POINTS FIRST
for i in range(len(mdsdistances)):
    xpoints.append(mdsdistances[i][0])
    ypoints.append(mdsdistances[i][1])

fig, ax = plt.subplots()
ax.scatter(xpoints,ypoints)
for i, txt in enumerate(names):
    ax.annotate(txt, (xpoints[i], ypoints[i]))


gradient = findgradient(mdsdistances, psychsimilarities, delta)
stress = stresscalc(mdsdistances, psychsimilarities)
print(stress)
# checkval = gradientcheck(gradient)
while stress >= 10:
    for i in range(len(gradient)):
        mdsdistances[i][0] += (-scaling * gradient[i][0])
        mdsdistances[i][1] += (-scaling * gradient[i][1])
        xpoints[i] = (mdsdistances[i][0])
        ypoints[i] = (mdsdistances[i][1])
    gradient = findgradient(mdsdistances, psychsimilarities, del
ta)
    stress = stresscalc(mdsdistances, psychsimilarities)
    stressvals.append(stress)
    iterations += 1
#     checkval = gradientcheck(gradient)
#     print("checkval", checkval)
    print(stress)

fig, ax = plt.subplots()
ax.scatter(xpoints,ypoints)
for i, txt in enumerate(names):
    ax.annotate(txt, (xpoints[i], ypoints[i]), fontsize = 10)
```

30.7756268582338
29.2340216796795
27.950080655913332
26.797115332417903
25.7273829643099
24.753166713246255
23.899104262761337
23.149025005780473
22.464751510929926
21.82702609440978
21.245103733498063
20.729425551870214
20.27616324728105
19.871781088943944
19.500433616240073
19.148348445214474
18.805494981843612
18.46539979021808
18.1234927155003
17.77387645426199
17.404145117891545
16.990280785698285
16.507569884263116
15.963318548250355
15.393851918303206
14.832622896517655
14.297015673402099
13.790391489399273
13.312754270345803
12.867907450615105
12.458351663800737
12.083811203507201
11.742988033710164
11.434044704289366
11.154896928787386
10.90357157851574
10.678330974627256
10.477555133536619
10.299579050905338
10.142638826810842
10.004938501072862
9.88474301386693





4.[20pts, HELP] Write the code that follows a gradient in order to find positions that minimize the stress – be sure to take small steps in the direction of the gradient (e.g. 0.01*gradient). Plot the sport names at the resulting coordinates. Do the results agree with your intuitions about how this domain might be organized? Why or why not?

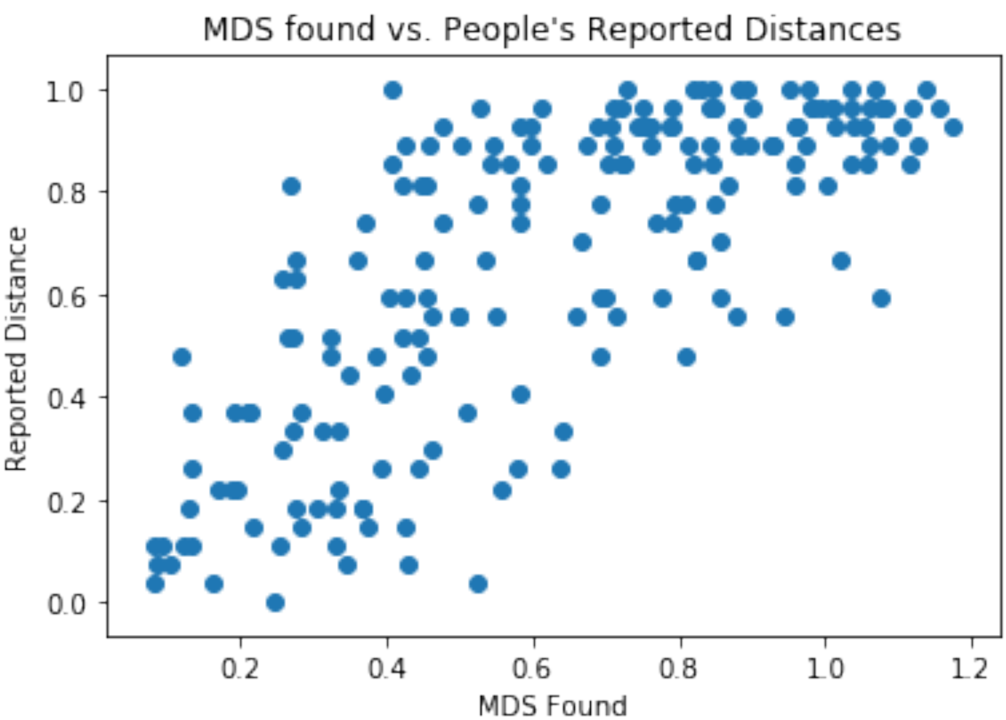
The results are in line with what my intuition is about how sports might be organized. For example, all of the water-related (or snow-related) sports (i.e. canoeing, surfing, swimming, skiing) are clustered together. In addition, the "fighting" sports (i.e. wrestling, boxing) are clustered together, and the sports that require a ball are also clustered in the same region of the graph.

1. [5pts, SOLO] Make a scatter plot of the the pairwise distances MDS found vs. people's reported distances. Briefly describe what good and bad plots would look like and whether yours is good or bad.

In [48]:

```
#QUESTION 5
psydist = []
mdsdist = []

col = 1
for row in range(0,20): #for each sport
    #compare sport to each remaining sport
    c = col
    for c in range(c,20):
        psydist.append(1-psychsimilarities[row][c])
    col += 1

col = 1
for row in range(0,20): #for each sport
    #compare sport to each remaining sport
    c = col
    for c in range(c,20):
        mdsdist.append(np.sqrt(np.square(mdsdistances[row][0] - mdsdistances[c][0]) + np.square(mdsdistances[row][1] - mdsdistances[c][1])))
    col += 1

print(len(psydist))
print(len(mdsdist))
plt.scatter(mdsdist, psydist)
plt.title("MDS found vs. People's Reported Distances")
plt.xlabel("MDS Found")
plt.ylabel("Reported Distance")
plt.show()
```

```
[[ 0.45059588  0.93361779]
 [ 0.81299194  0.65675009]
 [ 0.71540539  1.00355779]
 [ 1.04467129  0.5215493 ]
 [ 0.87695297  0.80640709]
 [ 0.22328299  0.00331391]
 [ 0.99462255  0.82237336]
 [ 0.40094204  0.77095533]
 [ 0.53313084  0.93879966]
 [ 0.64772708  0.42566199]
 [ 0.35188643 -0.01920809]
 [ 0.22421169  0.27792082]
 [ 0.07756792  0.95150369]
 [ 0.79692732  0.91311267]
 [ 0.44537676  0.67625818]
 [ 0.4174001  -0.07519789]
 [ 1.01292908  0.24162153]
 [ 0.91208671  0.49659232]
 [ 0.28573642 -0.06717961]
 [ 0.06969944  0.75515676]
 [ 0.02962239  0.03530635]]
190
190
```

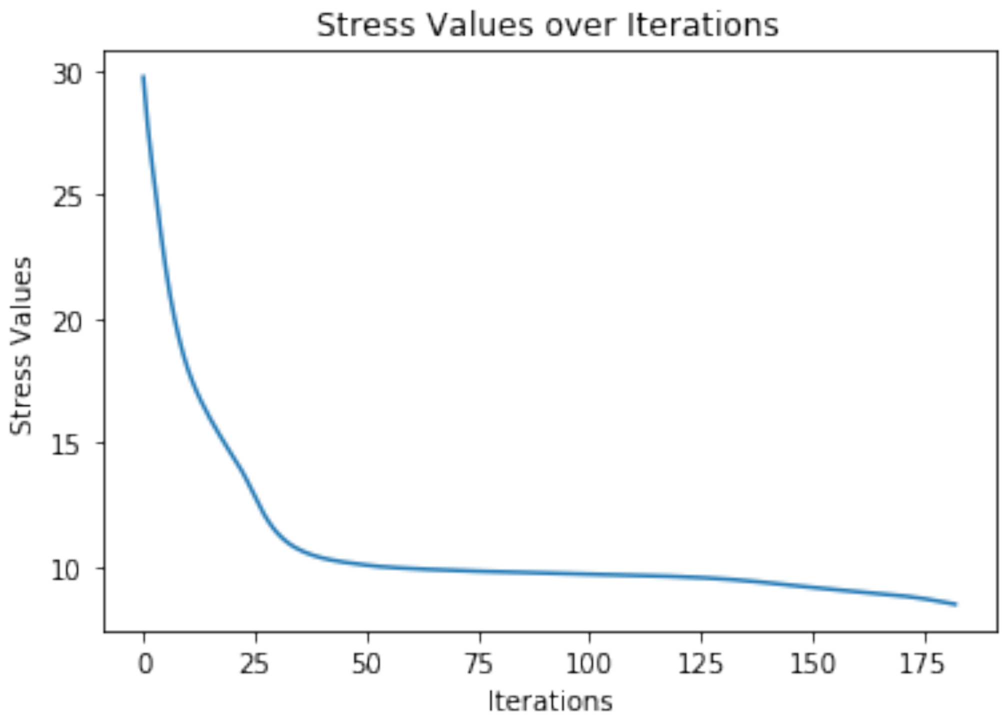


A good plot has a trend, meaning the x values and y values increase or decrease with some sort of consistent relationship. In this case, a good plot would have a positive correlation, meaning as people's reported distances increased, my MDS plots found should have also increased. This is somewhat represented in my graph but there are definitely some outliers. For example, one point was reported at a near 1 distance by people psychologically, but my MDS found it to be at ~0.4 distance value.

1. [5pts, SOLO] Plot the stress over iterations of your MDS. How should you use this plot in order to figure out how many iterations are needed?

In [54]:

```
#QUESTION 6

plt.plot((np.arange(iterations)), stressvals)
plt.title("Stress Values over Iterations")
plt.xlabel("Iterations")
plt.ylabel("Stress Values")
plt.show()
```



This plot can be used to figure out how many iterations are needed by looking at where the stress values begin to plateau, which is around a stress value of ~9. By the looks of my graph, my stress values continue to decrease even after 175 iterations, meaning I should probably run my program for longer. However, for the purpose of this assignment and on piazza/from advice from GSI's, I was told a stress value of around 10 is sufficient.

1. [5pts, SOLO] Run the MDS code you wrote 10 times and show small plots, starting from random initial positions. Are they all the same or not? Why?

After running MDS 10 times it can be seen that the graphs are not all the same. This is due to a couple factors, including that the points are all starting from random initial positions. This means that their locations on the graph will vary depending on where they started that round of MDS. In addition, the method of gradient descent we are using does not account for local vs global minima of the stress function. If the gradient descent method finds a local minimum, this will also cause the resulting graph to look differently.

In [80]:

```
#Question 7

import numpy as np
import matplotlib.pyplot as plt

#to compute partial derivatives
delta = 0.005
scaling = 0.01

#other needed variables
gradient = []
stress = 0

#read in similarities
psychsimilarities = np.loadtxt(open("Assignment5-similarities.csv", "rb"), delimiter=",", skiprows=1)

#assign random values to the starting points on MDS grid
mdsdistances = np.random.uniform(0, 1, size=(21,2))

def stresscalc(mdsdistances, psychsimilarities):
```

```
def stresscalc(mdsdistances, psychsimilarities):
    stress = 0
    mdsdistance = 0
    psychdistance = 0
    row = 0
    col = 1
    for row in range(0,20): #for each sport
        #compare sport to each remaining sport
        c = col
        for c in range(c,20):
            psychdistance = 1-psychsimilarities[row][c]
            mdsdistance = np.sqrt(np.square(mdsdistances[row][0] - mdsdistances[c][0]) + np.square(mdsdistances[row][1] - mdsdistances[c][1]))
            stress += np.square(psychdistance-mdsdistance)
        col += 1
    return stress

def findgradient(mdsdistances, psychsimilarities, delta):
    gradient= np.copy(mdsdistances)

    #Find the partial derivative by changing x and keeping y & changing y and keeping x constant.
    for i in np.arange(len(mdsdistances) -1): #for all coordinates in our grid

        #FIRST SOLVE FOR X
        coordinatesmodplus = np.copy(mdsdistances)
        coordinatesmodminus = np.copy(mdsdistances)

        coordinatesmodplus[i][0]+= delta #changing x value, and keeping y the same
        fpdx = stresscalc(coordinatesmodplus, psychsimilarities) #f(changed concept x, y remains)

        coordinatesmodminus[i][0]-=delta #changing x value, keeping y the same
        fmdx=stresscalc(coordinatesmodminus, psychsimilarities) #f(changed concept x, y remains)

        partialderivativex=(fpdx-fmdx)/(2*delta)

        gradient[i][0]=partialderivativex #update that part of the gradient with the partial derivative found for element i of values
```

```
        #THEN SOLVE FOR Y
        coordinatesmodplus = np.copy(mdsdistances)
        coordinatesmodminus = np.copy(mdsdistances)

        coordinatesmodplus[i][1]+= delta #changing y value, and
keeping x the same
        fpdy = stresscalc(coordinatesmodplus, psychsimilarities)
#f(changed concept y, x remains)

        coordinatesmodminus[i][1]-=delta #changing y value, keep
ing x the same
        fmdy=stresscalc(coordinatesmodminus, psychsimilarities)
#f(changed concept y, x remains)

        partialderivativey=(fpdy-fmdy)/(2*delta)

        gradient[i][1]=partialderivativey #update that part of t
he gradient with the partial derivative found for element i of v
alues

    return gradient

def gradientcheck(gradient):
    passed = []
    for i in range(len(gradient)):
        if gradient[i][0] <= 0.1:
            passed.append(1)
        if gradient[i][1] <= 0.1:
            passed.append(1)
    return sum(passed)

names = ['football', 'baseball', 'basketball', 'tennis',
         'softball', 'canoeing', 'handball', 'rugby',
         'hockey', 'icehockey', 'swimming', 'track',
         'boxing', 'volleyball', 'lacrosse', 'skiing',
         'golf', 'polo', 'surfing', 'wrestling', 'gymnastics']

xpoints = []
ypoints = []
stressvals = []
iterations= 0

#PLOT RANDOM POINTS FIRST
for i in range(len(mdsdistances)):
```

```
    xpoints.append(mdsdistances[i][0])
    ypoints.append(mdsdistances[i][1])

# fig, ax = plt.subplots()
# ax.scatter(xpoints,ypoints)
# for i, txt in enumerate(names):
#     ax.annotate(txt, (xpoints[i], ypoints[i]))
n = 0

while n < 10:
    mdsdistances = np.random.uniform(0, 1, size=(21,2))
    gradient = findgradient(mdsdistances, psychsimilarities, del
ta)
    stress = stresscalc(mdsdistances, psychsimilarities)
    print(stress)
    checkval = gradientcheck(gradient)
    while stress >= 10:
        for i in range(len(gradient)):
            mdsdistances[i][0] += (-scaling * gradient[i][0])
            mdsdistances[i][1] += (-scaling * gradient[i][1])
            xpoints[i] = (mdsdistances[i][0])
            ypoints[i] = (mdsdistances[i][1])
        gradient = findgradient(mdsdistances, psychsimilarities,
delta)
        stress = stresscalc(mdsdistances, psychsimilarities)
       # checkval = gradientcheck(gradient)
        print(stress)
    print('NEW STRESS:', stress)

    fig, ax = plt.subplots()
    ax.scatter(xpoints,ypoints)
    for i, txt in enumerate(names):
        ax.annotate(txt, (xpoints[i], ypoints[i]), fontsize = 10
)
    n += 1
```

```
31.342938590553423
28.149608085375842
25.87853143113200
24.10320061974803
22.47113474020452
20.856952093264567
19.63727693994673
18.731238250451458
```

18.02571557251666
17.4266267667201
16.888479855793516
16.425117315752036
16.032557816133224
15.694566306692632
15.397127246681586
15.131205935041729
14.892048241942275
14.67664637799215
14.482034957346178
14.304930220336084
14.141849308117278
13.989599451462965
13.846026014805355
13.710259041870891
13.58208042497738
13.461277128602639
13.347517806472581
13.24046805607119
13.139856585812977
13.045447953952998
12.956984609905986
12.874153302472605
12.796591966172132
12.72393058414721
12.655849907656764
12.592129308220379
12.53265184118435
12.477359055262701
12.426182582670727
12.378990262922645
12.335566921243462
12.295626263704028
12.258838488723358
12.224859204990546
12.193351756280578
12.16400099022353
12.136519770784888
12.110650533414486
12.086163975570056
12.062856376356427
12.040546461297426
12.019072307950475
11.998288526456012

11.978063803781929
11.958278827519882
11.938824572508395
11.919600921339189
11.900515586767604
11.881483304377737
11.862425264730824
11.843268754430458
11.823946974740062
11.804399004804685
11.784569874772071
11.764410713051122
11.743878932564481
11.722938423975462
11.701559729876344
11.679720182443578
11.65740399676909
11.634602320702768
11.6113132466961
11.587541788919872
11.563299817761642
11.538605923478801
11.513485153473262
11.487968538673751
11.46209230145042
11.435896629380723
11.40942391420283
11.382716398161943
11.355813238860998
11.328747087288221
11.301540349167677
11.274201333864282
11.246720445441483
11.219066385764675
11.191181985311712
11.162978849759737
11.134329863753761
11.105059127221484
11.074929955474287
11.043632346701115
11.010771288946819
10.975856847464996
10.938296862155575
10.897393519810716
10.852345711372962

```
10.80225940712196
10.746167923799518
10.683063063564887
10.611937501359968
10.531839634360896
10.44194480018293
10.341649868264028
10.230698321205493
10.109336198966925
9.978483820819106
NEW STRESS: 9.978483820819106
33.6219566024969
31.096891838182067
29.1736315747589
27.615607273434406
26.359881624843485
25.35143396452544
24.544111670843613
23.85687741850006
23.2995511559155
22.855877231201685
22.493043691865807
22.18597085945151
21.91729814431701
21.675216210870236
21.450671503258793
21.22963027191719
20.96330220053948
20.612384419221424
20.262113684468762
19.962565436211378
19.71816272626383
19.518470419633346
19.351474433900915
19.207339225912076
19.0788929221613
18.961255922091564
18.851251339919752
18.746782943648267
18.64635075176573
18.548785083157703
18.453139182398328
18.3586484232336 8
18.264704864201235
18.170831858386258
```

18.07665761679936
17.981889124410955
17.886286976748146
17.78964074561194
17.691744078483705
17.592368828660245
17.49123794266224
17.387997478663276
17.28218890715827
17.173223678410515
17.06036285616698
16.942705216115677
16.819186897620067
16.688593224560343
16.5495789502434
16.400692486443013
16.240410097930642
16.067201888621327
15.879648801853467
15.676591419778331
15.457240030961229
15.221181721806259
14.968347409751914
14.699192344955009
14.415369477806678
14.120792956453451
13.822281431741358
13.528720186319449
13.248833304114747
12.989082601985317
12.752812747656565
12.540470841132421
12.350246505804021
12.178663764666474
12.020911926295428
11.870862892315156
11.72088370955047
11.561899500322037
11.384651861419501
11.182759682568518
10.955999486300412
10.710793094266151
10.457498371803114
10.206987335332583
9.968411308634911

```
NEW STRESS: 9.968411308634911
28.970014787935384
27.10980580156949
25.257032442363073
23.550682137087303
22.022190606002344
20.555028397451235
19.456929804967896
18.591332519787535
17.745779150286207
16.9869002891223
16.341438425146425
15.754233252742072
15.181288578249436
14.63268825888342
14.128963913628033
13.664256702290512
13.233447306168504
12.833318357167101
12.459335286297005
12.108038928210997
11.77766072032106
11.467776843398031
11.178967173924747
10.91229860065104
10.668355725729038
10.446095991757407
10.243125134503677
10.058746998195742
9.892534737573737
NEW STRESS: 9.892534737573737
35.76211835466345
33.11980888773937
30.978629755821416
29.13606640660183
27.475398428309664
25.943552689085433
24.458037742139425
22.937769784472675
21.387551281523177
19.7734197066562
17.82267207307888
15.857745652629276
14.495707078751481
13.608171251609802
```

```
13.022797497706359
12.62255470943862
12.33588750398776
12.120769174292056
11.950236091636711
11.807157894857504
11.680318933610724
11.564092231618435
11.456624219713083
11.356384470612207
11.261986023832437
11.172508837722301
11.087398731978341
11.00628750648913
10.92889821714553
10.854998567773363
10.78435379817086
10.716673110713751
10.651564172907744
10.588510247548076
10.526889272470182
10.466068218077035
10.4056022766445
10.345499381826224
10.28638608231048
10.229389990794369
10.175760919897572
10.126476282042384
10.082041144325135
10.042496095530751
10.007528958672477
9.976593590419844
NEW STRESS: 9.976593590419844
32.39945088281318
30.307103092172603
28.741219014846394
27.36670014980305
25.966664132232125
24.410909250032105
22.792963637092722
21.34052173747163
20.156273649013478
19.216611029626097
18.448078260957935
17.816359936683583
```

```
17.27387153722492
16.78548912901293
16.337367573944604
15.925113255835418
15.544184148434054
15.189463312025882
14.85540622521142
14.535019290567202
14.219555231133912
13.899567876502243
13.56694421554414
13.217048424035882
12.85038959225064
12.473539739452134
12.098069931014468
11.736801151624618
11.399848901242896
11.092925553794007
10.817674999782291
10.572856423620685
10.355624202521641
10.162605992507848
9.990685266477412
NEW STRESS: 9.990685266477412
28.70236579204507
26.099572982894834
23.960908549417585
22.037176477292046
20.19864266526784
18.493299765665395
17.03070128765523
15.880290691497104
15.012511699452116
14.358107282439592
13.851408159917098
13.443580780144522
13.101739666150008
12.804656254578514
12.538762717954487
12.295107985599962
12.067401300384851
11.851426592069869
11.645461226017051
11.450503475800092
11.269381493982394
```

```
11.104509086602775
10.956258472414769
10.82367366654262
10.706650805216515
10.60501298233219
10.517191077035065
10.440752060860461
10.37302000394872
10.311433392701808
10.253784821829962
10.198451241617313
10.144644616657017
10.09256860449164
10.043250464713594
9.997985342500396
NEW STRESS: 9.997985342500396
34.601263156410134
30.641070883397177
27.551525281002363
24.829912868213906
22.260413795030892
20.036923626413337
18.302453984464364
16.966519673871005
15.819219583908987
14.853399935504939
14.133070984566292
13.60647530740161
13.21607864645299
12.91763977342773
12.678189102552414
12.473074944665786
12.284928013411635
12.10902479767178
11.950767243588169
11.812188651208572
11.691396262332663
11.585895706063154
11.493799912291752
11.413237623368683
11.34155830703044
11.279059471730077
11.226181920029124
11.181848939287542
11.144625085453573
```

11.113190465996839
11.08643729261266
11.063461794084176
11.043532992237928
11.026061871323021
11.010576205449354
10.996700717481803
10.984141048737213
10.97267020306075
10.962116594535484
10.952353237750245
10.943287910973748
10.934854307999542
10.927004291957232
10.91970138711493
10.912915606894126
10.906619638019453
10.900786306365557
10.895387163484493
10.890391971942137
10.885768841183003
10.881484772878311
10.87750640768091
10.8738008127915
10.870336200829295
10.867082516971923
10.864011868485695
10.861098797095464
10.858320410877472
10.855656400439276
10.853088966387439
10.850602683617794
10.848184324508807
10.845822658878372
10.843508244358121
10.841233217081706
10.838991089462484
10.836776559388506
10.834585333333493
10.83241396457855
10.830259706853386
10.828120383137312
10.825994269020308
10.82387998984236
10.821776430749221

10.819682658783892
10.817597856148126
10.815521263796922
10.813452134561786
10.811389695028595
10.80933311542186
10.807281486768241
10.805233804631667
10.803188958729924
10.801145727763352
10.799102778808923
10.797058670663395
10.795011860556352
10.79296071370103
10.790903515211093
10.788838483986469
10.786763788265157
10.78467756265465
10.78257792659901
10.780463004409112
10.778330947184859
10.776179957183071
10.77400831542375
10.771814413554601
10.76959679117179
10.767354179865503
10.765085555144957
10.762790197009679
10.760467759181731
10.758118345834179
10.755742593049971
10.75334175033222
10.750917755520504
10.748473294846969
10.746011839082843
10.743537647240046
10.741055731372333
10.738571779585202
10.736092038926023
10.733623164563253
10.73117204563927
10.728745620617225
10.726350695447953
10.723993776534742
10.721680927741366

10.7194176572472
10.717208836573903
10.715058651110088
10.712970579233257
10.710947395751315
10.708991194777363
10.707103427142261
10.705284947847005
10.703536069681753
10.7018566198517
10.700245997153035
10.698703227877399
10.69722701916217
10.695815808939674
10.694467811976358
10.69318106174699
10.691953448071795
10.690782750573126
10.689666668093919
10.688602844274731
10.68758888951763
10.68662239958048
10.685700971048577
10.684822213925996
10.68398376157884
10.683183278247858
10.682418464330796
10.681687059614669
10.68098684461699
10.680315640171411
10.67967130536841
10.679051733935486
10.678454849113233
10.67787859705279
10.677320938729133
10.676779840327855
10.676253262025277
10.675739145038573
10.675235396773802
10.674739873845507
10.674250362678142
10.67376455732731
10.67328003407367
10.672794222242711
10.672304370586588

10.671807508424031
10.671300400569699
10.670779494886704
10.67024086106178
10.669680118923484
10.669092354293307
10.668472019968512
10.667812818977257
10.667107566713241
10.666348027944663
10.665524724001653
10.66462670468639
10.663641278646585
10.662553695152857
10.661346769508791
10.660000443837083
10.6584912749399
10.656791841647342
10.65487006600174
10.652688446406813
10.650203207313101
10.647363380095776
10.64410984455191
10.640374380763728
10.636078807153487
10.631134311165889
10.625441110365768
10.618888606131263
10.611356196822685
10.602714884581273
10.592829719635017
10.581562962337397
10.568777605804778
10.554340619340982
10.538125010510472
10.520009658767773
10.499875946703655
10.477600565393509
10.45304447326544
10.426038725084696
10.396368589312129
10.363757898488089
10.32785589698222
10.288229067304853
10.24436072176419

```
10.195661708215635
10.141496392310577
10.081228806666369
10.01429369139611
9.940294791956934
NEW STRESS: 9.940294791956934
26.896217980244458
25.351317934981992
23.95227489975143
22.77160526665307
21.633913600824243
20.507928931115554
19.706693496856946
19.172172248978363
18.80315316865545
18.526080174521148
18.291594048461423
18.067336092030796
17.828794020531546
17.566810953247675
17.30147806833801
17.05189249961135
16.823546192046248
16.613521615562036
16.414192117000447
16.21003047866964
15.986787303631381
15.766960655771971
15.56712618670896
15.39080164348875
15.233489302618212
15.09518321058061
14.977007766751433
14.876176935361858
14.789463182241592
14.71404305640713
14.647574009394306
14.58808308630392
14.533804758678762
14.482989735515696
14.433659028186344
14.383256360679606
14.328183223320943
14.263410991902852
14.182902724483252
```

14.081873041814267
13.960188605194396
13.823520232064594
13.680624265051955
13.539529396668472
13.405502508131658
13.280970519313096
13.166329713798635
13.060832221388269
12.96323554853484
12.87218667169303
12.786411858169108
12.704790362814562
12.626367572918426
12.550339677486273
12.47602542935698
12.402830499230166
12.33020369660344
12.257579525459683
12.18429631983162
12.10947278585799
12.031821146737903
11.949388585493452
11.859299590497603
11.757791471460612
11.641082407089254
11.507258829007496
11.358050888134633
11.198750025636656
11.03620778687764
10.876626599786738
10.72443959720114
10.582263145576173
10.45133666780013
10.332005158213033
10.224076461749396
10.127044775000122
10.04022079358237
9.96280798747842
NEW STRESS: 9.96280798747842
33.214130358936394
30.850949672848103
29.25705229953264
28.086187306927663
27.080002324505635

26.188735772723444
25.35210759572363
24.409132911421022
23.268060213379293
22.04070890898187
20.90637922895061
19.975383279709106
19.210102470664182
18.467791834673374
17.694186720960953
17.01906337636115
16.428055761087325
15.88285150138305
15.367565003115248
14.881356906467278
14.427164833482754
14.011699389746807
13.64226950736082
13.321092329439884
13.044584986052111
12.80683348606905
12.602323495706889
12.426678575853016
12.27638660675175
12.148030324362296
12.034109564056218
11.927103556921418
11.835159867822323
11.760674162648632
11.70122743706483
11.653574816240802
11.614536664449195
11.581189756015704
11.55090789083838
11.521506426356535
11.491594696670054
11.460919571433784
11.430246555488283
11.400682546501121
11.37291118126235
11.34687256773553
11.322125353595547
11.298444072224099
11.275948381976631
11.254788679571197

11.235033351585354
11.216732126320643
11.199929052200382
11.18462713099956
11.170762584300197
11.158209790805461
11.146804846622404
11.136371007101138
11.126737479446449
11.117750198490647
11.109276339225852
11.10120485969331
11.093444912287906
11.085923334114453
11.078581918360616
11.071374827530176
11.064266304598982
11.057228725603624
11.050240981008226
11.043287149088336
11.036355417497793
11.029437210437262
11.02252648363807
11.01561915522485
11.008712646223351
11.001805509529884
10.994897130401112
10.987987484974894
10.981076946083958
10.974166127786892
10.967255761726408
10.960346599732326
10.953439338104102
10.946534559802073
10.93963269140818
10.932733972227673
10.92583843332894
10.918945884684097
10.91205590889674
10.905167860296558
10.898280868451304
10.891393845396802
10.884505496116127
10.877614332005127
10.87071868724117

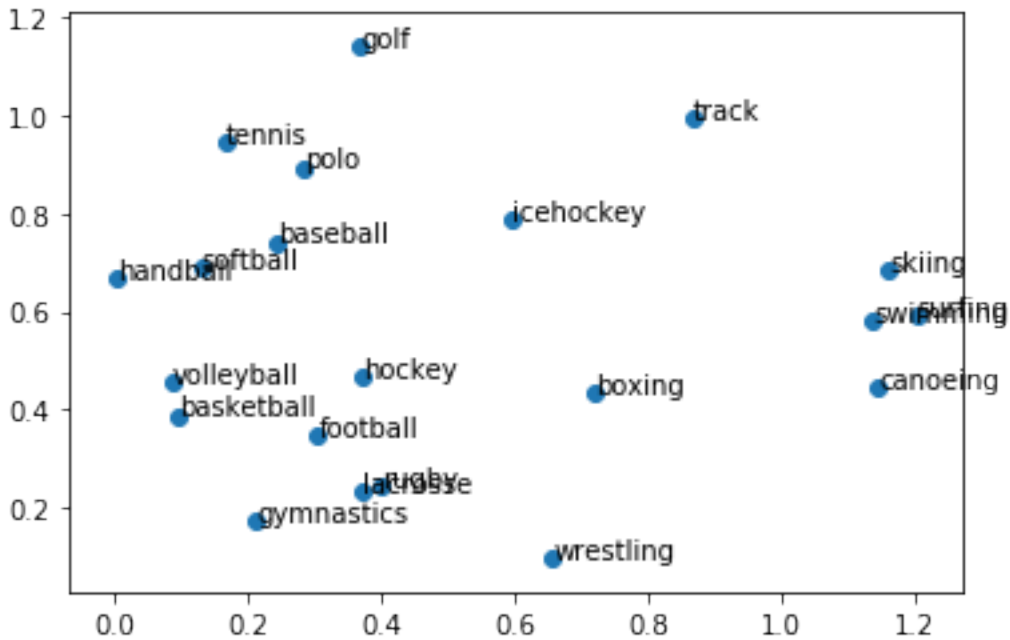
10.863816738116803
10.856906525506618
10.849985980696113
10.843052954812562
10.836105252055882
10.829140666831693
10.82215702474144
10.81515222718972
10.808124299137308
10.801071439268824
10.79399207157391
10.786884897075097
10.779748944192882
10.772583616036988
10.76538873276839
10.758164567102956
10.750911871032569
10.743631891928562
10.736326376358038
10.728997560180057
10.7216481437746
10.714281251571018
10.706900375354707
10.699509301106863
10.692112019337516
10.684712618966222
10.677315164749436
10.66992355800473
10.662541379893812
10.655171715744107
10.647816957744933
10.640478581760698
10.63315689183812
10.6258507230772
10.61855708964664
10.611270759513872
10.603983730454384
10.596684572447808
10.589357588771845
10.581981730806671
10.574529178323749
10.566963466267747
10.559236999452457
10.55128774831304
10.543034866078631

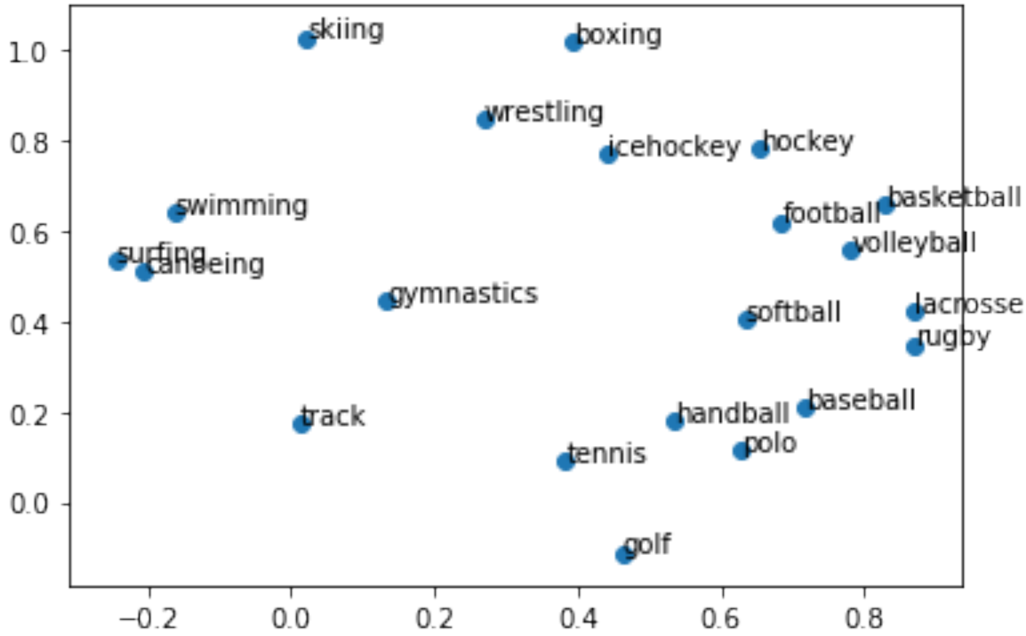
```
10.534372924271521
10.52516446379728
10.515230678210099
10.504340431170212
10.492198716718324
10.478437461714451
10.46261455963128
10.444230923573214
10.422778067522467
10.397825058478077
10.369136424702909
10.33678033117714
10.301155855575539
10.262873803073719
10.2224887444367
10.180180181338798
10.13557616950187
10.087961020580002
10.036977373340441
9.983436665457774
NEW STRESS: 9.983436665457774
31.91746312012702
29.32126613209557
27.185714923253183
25.272160388355097
23.52301460884494
21.93194448866228
20.315616432607023
18.646371591611377
17.093127679393856
15.818801353783353
14.897030695650985
14.277053247072057
13.868201740225146
13.595591514120752
13.40884794182969
13.275633430456283
13.175984241673133
13.097959996420371
13.035262603488778
12.98397925517204
12.940736757827272
12.902621641521824
12.86721486615443
12.832461731262095
```

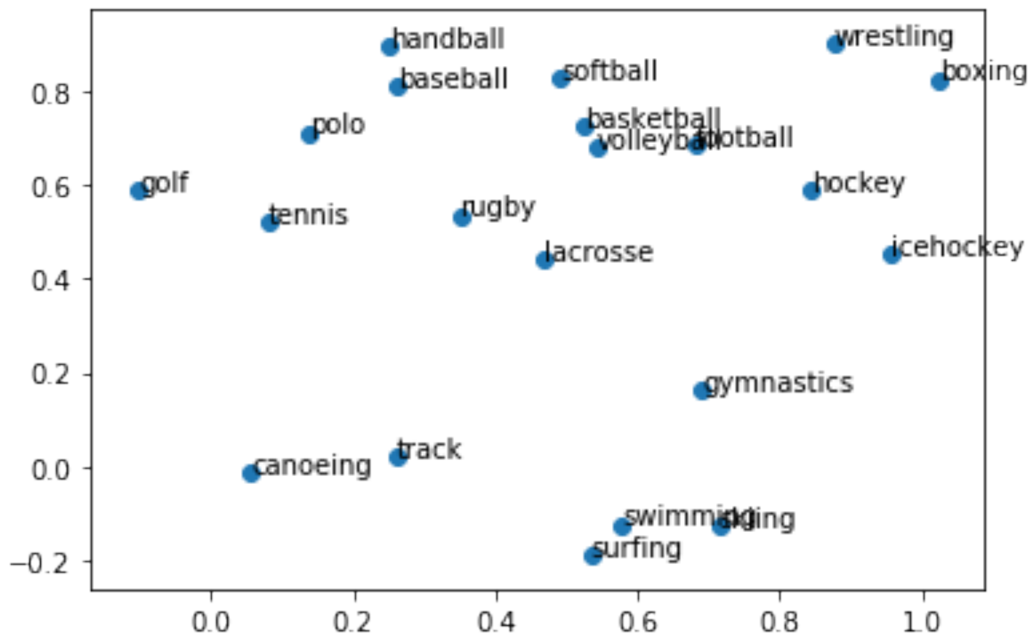
12.796495317914214
12.757502250270525
12.713697886632948
12.663451319407924
12.605540390899431
12.539507276377972
12.46612268724876
12.387592897796294
12.307015029679926
12.227489645929287
12.151512519644756
12.080740134204447
12.016017142759816
11.95755172254147
11.905134795867506
11.858330213390548
11.816606071072808
11.779410416351261
11.7462088665028
11.716501363864515
11.689829105217811
11.665776892183036
11.64397287320105
11.624086415008698
11.605824571884519
11.588927587416917
11.573163792171515
11.558324126194757
11.544216362841286
11.530658975129986
11.517474478808543
11.504482006020227
11.491488811425564
11.47828040525207
11.464609076925894
11.450180732116642
11.434640092713213
11.417553920660927
11.398390109123815
11.37648710192884
11.35100884413698
11.320908531379374
11.285026090471975
11.242593836444994
11.194232118585603

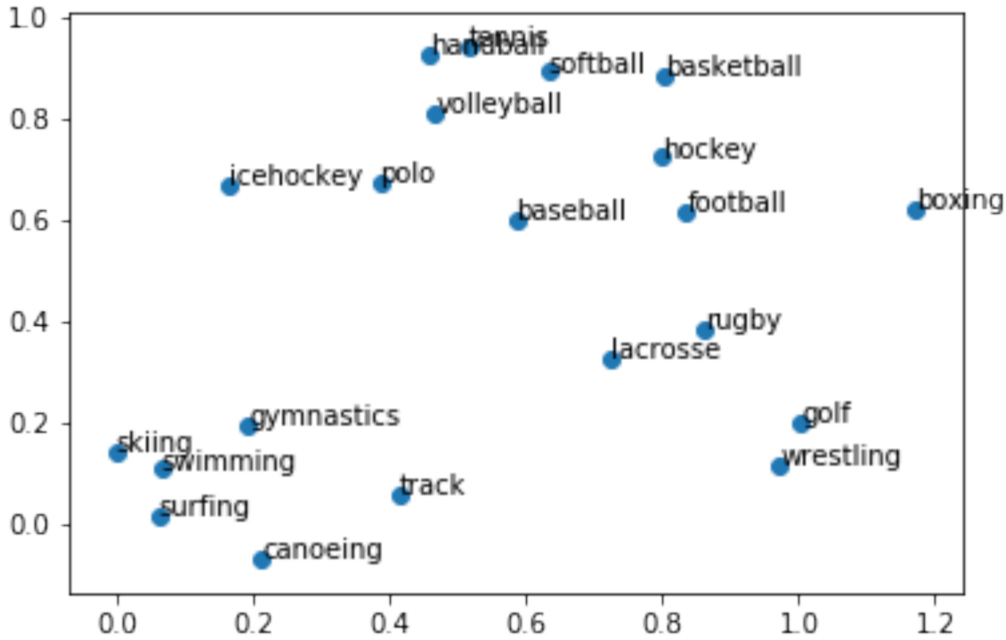
11.142592252018154
11.091530977595138
11.044416938019259
11.003138894494237
10.968176482915734
10.939120625224906
10.915138101790168
10.895209006645095
10.878043567984436
10.861390719686359
10.84197148261869
10.819897451930359
10.798174891575991
10.778655846146972
10.761805885701817
10.747452270616247
10.735213592324577
10.724684343449239
10.715504071034674
10.707375912135896
10.700064389876559
10.693386018143476
10.687198611933134
10.681391622948695
10.675878192165465
10.670588917998144
10.665467084904057
10.660465040048484
10.655541427986535
10.650659040751489
10.64578308971553
10.640879747409581
10.635914840487176
10.630852600941356
10.625654405448147
10.620277458470142
10.614673412813927
10.608786985092742
10.60255472979213
10.595904298280491
10.588754720852444
10.58101844667331
10.572605895806456
10.563432863102221
10.553430059439304

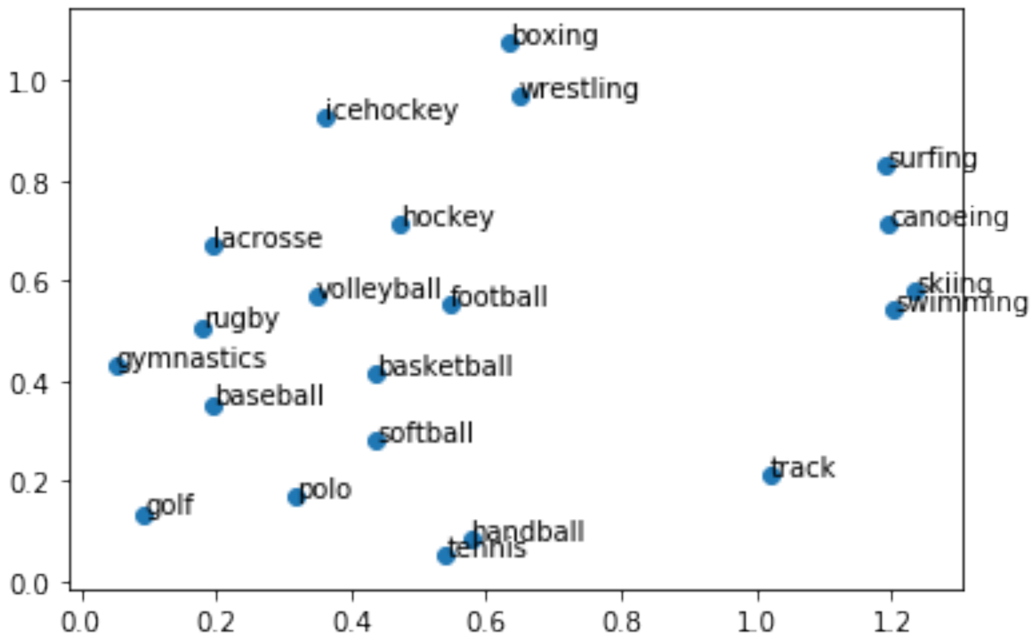
```
10.542552538198498
10.53078550935649
10.518143220239471
10.504659726854754
10.49037360702015
10.475311055173744
10.459471968006994
10.44282185497963
10.425290112400894
10.406773634086484
10.387144259475098
10.36625886266303
10.343971470490152
10.32014731032868
10.294678961822312
10.267504759044172
10.238629200194623
10.208144260190975
10.176249067297356
10.143263537173246
10.109629952900194
10.075896470883846
10.042679481513755
10.010607768750562
9.980258386164367
NEW STRESS: 9.980258386164367
```

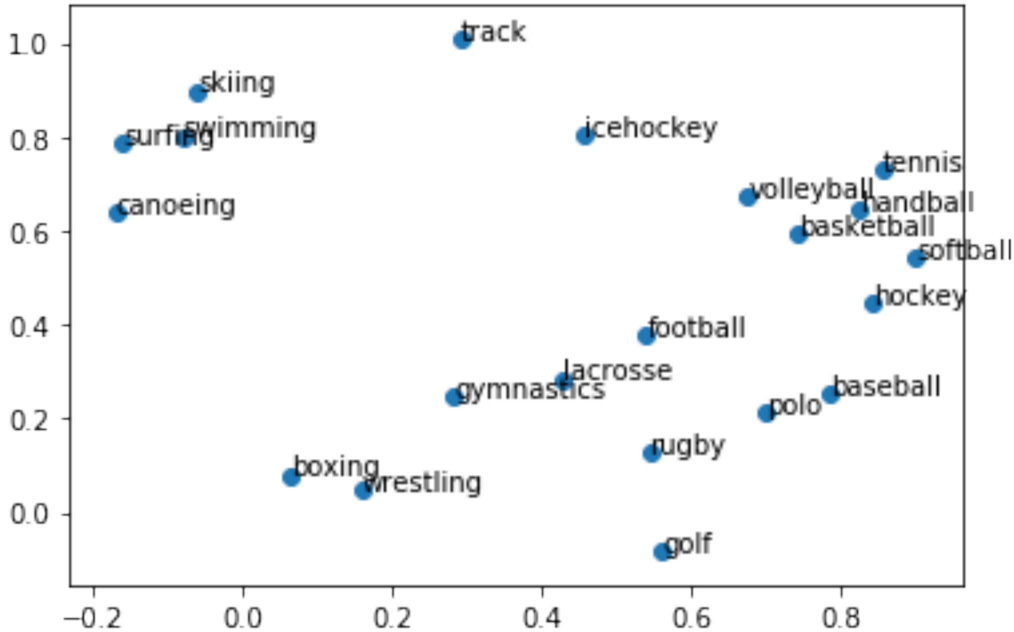


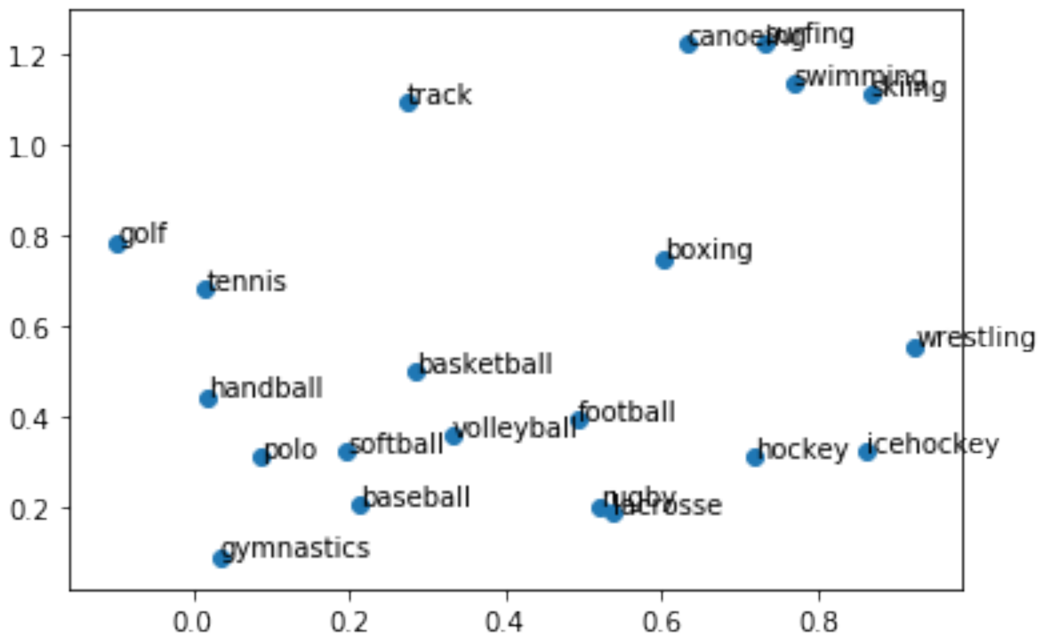

skiing
boxing
wrestling
icehockey
hockey
swimming
basketball
football
volleyball
surfing
canoeing
gymnastics
softball
lacrosse
rugby
track
handball
baseball
tennis
polo
golf
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8

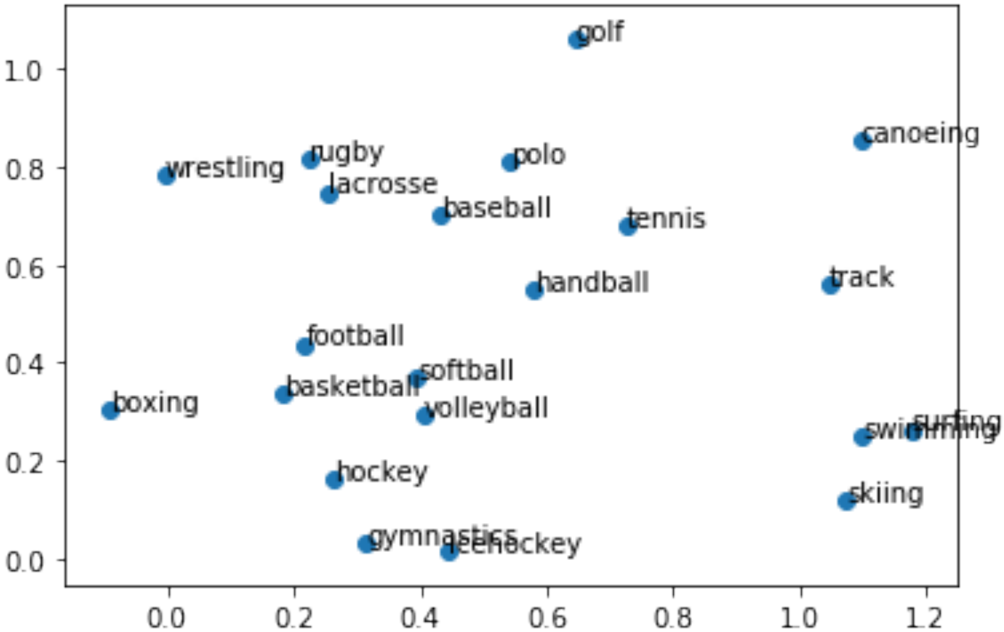

handball
wrestling
baseball
softball
boxing
polo
basketball
volleyball
football
golf
hockey
tennis
rugby
lacrosse
icehockey
gymnastics
track
canoeing
swimming
skiing
surfing
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

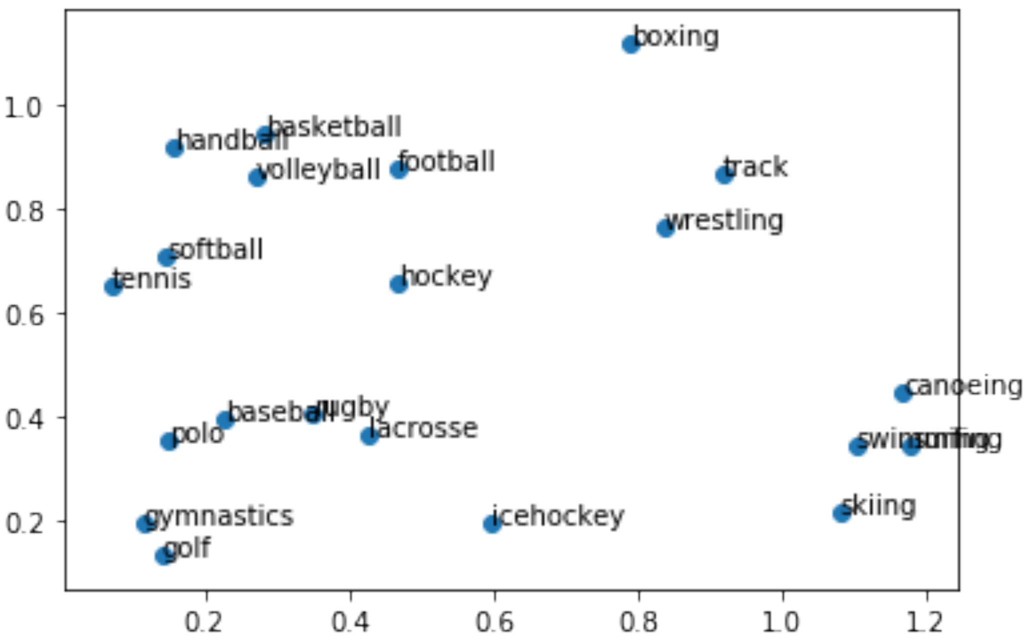

tennis
handball
softball
basketball
volleyball
hockey
icehockey
polo
baseball
football
boxing
rugby
lacrosse
golf
gymnastics
skiing
swimming
wrestling
track
surfing
canoeing

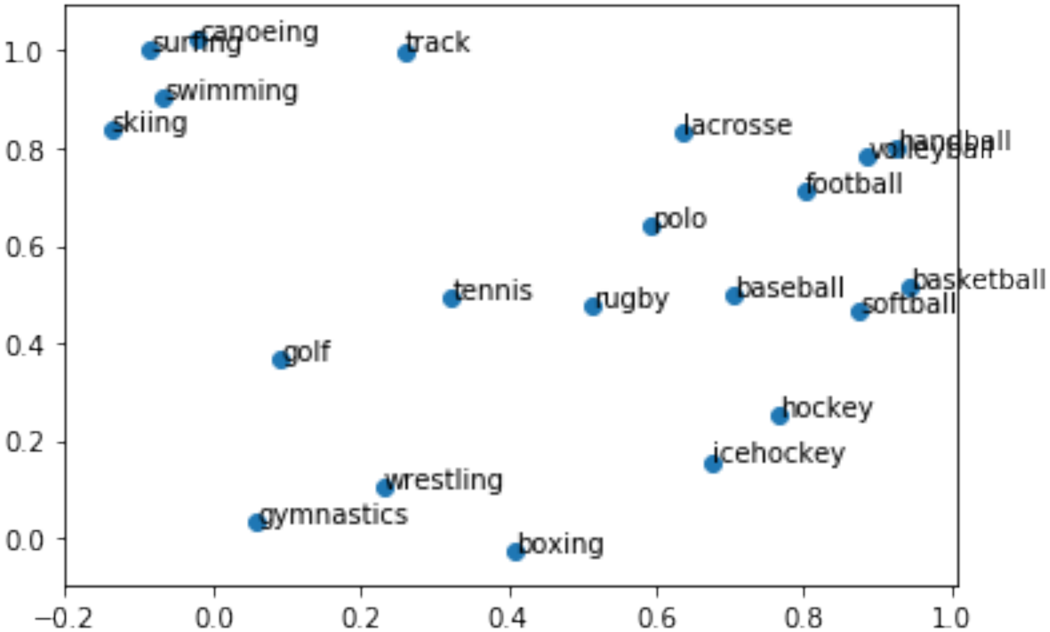

boxing
wrestling
icehockey
surfing
hockey
canoeing
lacrosse
skiing
volleyball
football
swimming
rugby
gymnastics
basketball
baseball
softball
track
polo
golf
handball
tennis


track
skiing
surfing
swimming
icehockey
tennis
volleyball
handball
canoeing
basketball
softball
hockey
football
lacrosse
gymnastics
baseball
polo
rugby
boxing
wrestling
golf
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8


canoeing
surfing
swimming
skiing
track
golf
boxing
tennis
wrestling
basketball
handball
football
volleyball
softball
polo
hockey
icehockey
baseball
rugby
lacrosse
gymnastics
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8


golf
canoeing
rugby
polo
wrestling
lacrosse
baseball
tennis
track
handball
football
softball
basketball
boxing
volleyball
surfing
swimming
hockey
skiing
gymnastics
icehockey
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2


boxing
basketball
handball
football
volleyball
track
wrestling
softball
tennis
hockey
canoeing
baseball
rugby
lacrosse
polo
swimming
skiing
gymnastics
icehockey
golf
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0
1.2



1. [5pts, SOLO] If you wanted to find one "best" answer but had run MDS 10 times, how would you pick the best? Why? Show a plot of the best and any code you used to find it.

In [ ]:

```
#QUESTION 8
psydist = []
mdsdist = []

mds1 = []
mds2 = []
mds3 = []
mds4 = []
mds5 = []
mds6 = []
mds7 = []
mds8 = []
mds9 = []
mds10 = []

def plot_distances(mdsdistances, psychsimilarities):
    col = 1
    for row in range(0,20): #for each sport
        #compare sport to each remaining sport
        c = col
        for c in range(c,20):
```

```
            psydist.append(1-psychsimilarities[row][c])
        col += 1

    col = 1
    for row in range(0,20): #for each sport
        #compare sport to each remaining sport
        c = col
        for c in range(c,20):
            mdsdist.append(np.sqrt(np.square(mdsdistances[row][0
] - mdsdistances[c][0]) + np.square(mdsdistances[row][1] - mdsdi
stances[c][1])))
        col += 1

    print(len(psydist))
    print(len(mdsdist))
    plt.scatter(mdsdist, psydist)
    plt.title("MDS found vs. People's Reported Distances")
    plt.xlabel("MDS Found")
    plt.ylabel("Reported Distance")
    plt.show()

# calc the trendline
z = numpy.polyfit(x, y, 1)
p = numpy.poly1d(z)
pylab.plot(x,p(x),"r--")
# the line equation:
print "y=%.6fx+(%.6f)"%(z[0],z[1])
```