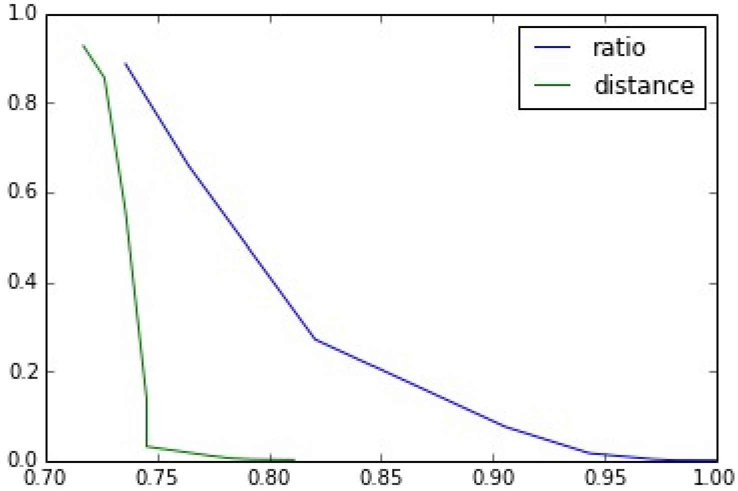
Question 5:
The python Levenshtein library provides another metric of string similarity called "ratio" (use `L.ratio(s1, s1)`). `ratio` gives a similarity score between 0 and 1, with higher meaning more similar. Add a column to "prod" with the `ratio` similarities of the **name** columns, and redo the precision/recall tradeoff analysis with the new metric. (Note: you will have to alter the `accuracy` method and the threshold range.) On this data, does `Levenshtein.ratio` do better than `Levenshtein.distance`? (Plot the two precision-recall curves together in one graph to compare them)

Answer 5:

```
%matplotlib inline
import pylab

def accuracy(max_ratio):
    similar = prod[(prod.ratio*10) >= max_ratio]
    correct = float(sum(similar.cluster_x == similar.cluster_y))
    precision = correct / len(similar)
    recall = correct / len(clusters)
    return (precision, recall)

thresholds = range(1,10)
p = []
r = []

prod['ratio'] = prod.apply(lambda r: L.ratio(r['name_x'], r['name_y']), axis=1)
#prod[:10]

for t in thresholds:
    acc = accuracy(t)
    p.append(acc[0])
    r.append(acc[1])

pylab.plot(thresholds, p)
pylab.plot(thresholds, r)
pylab.legend(['precision', 'recall'], loc='upper left')

%matplotlib inline
import pylab

def accuracy(max_ratio):
    similar = prod[(prod.ratio*10) >= max_ratio]
    correct = float(sum(similar.cluster_x == similar.cluster_y))
    precision = correct / len(similar)
    recall = correct / len(clusters)
    return (precision, recall)

thresholds = range(1,10)
p = []
r = []

prod['ratio'] = prod.apply(lambda r: L.ratio(r['name_x'], r['name_y']), axis=1)

for t in thresholds:
    acc = accuracy(t)
    p.append(acc[0])
    r.append(acc[1])
```

```
def accuracy(max_distance):
    similar = prod[prod.distance < max_distance]
    correct = float(sum(similar.cluster_x == similar.cluster_y))

    precision = correct / len(similar)
    recall = correct / len(clusters)
    return (precision, recall)

thresholds2 = range(1, 11)
p2 = []
r2 = []

for t in thresholds:
    acc = accuracy(t)
    p2.append(acc[0])
    r2.append(acc[1])

pylab.plot(r,p)
pylab.plot(r2,p2)
pylab.legend(['ratio', 'distance'], loc='upper right')
```

Conclusion: Ration is better than distance.


ratio
distance