1. Modifique el código de parsear blogs para agrupar entradas individuales en lugar de blogs enteros. ¿Las entradas del mismo blog se agrupan unas con otras? ¿Qué hay de las entradas de la misma fecha?

def getwords (html):

txt = re.compile(r'<[^>]+>').sub('', html)

words = re.compile(r'[^A-Z^a-z]+').split(txt)

return [word.lower() for word in words if word != '']

def getwordcounts (url):

d=feedparser.parse(url)

wc={}

for e in d.entries:

if 'summary' in e: summary=e.summary

else: summary=e.description

words=getwords(e.title+' '+summary)

for word in words:

wc.setdefault(word,0)

wc[word]+=1

return d.feed.title,wc

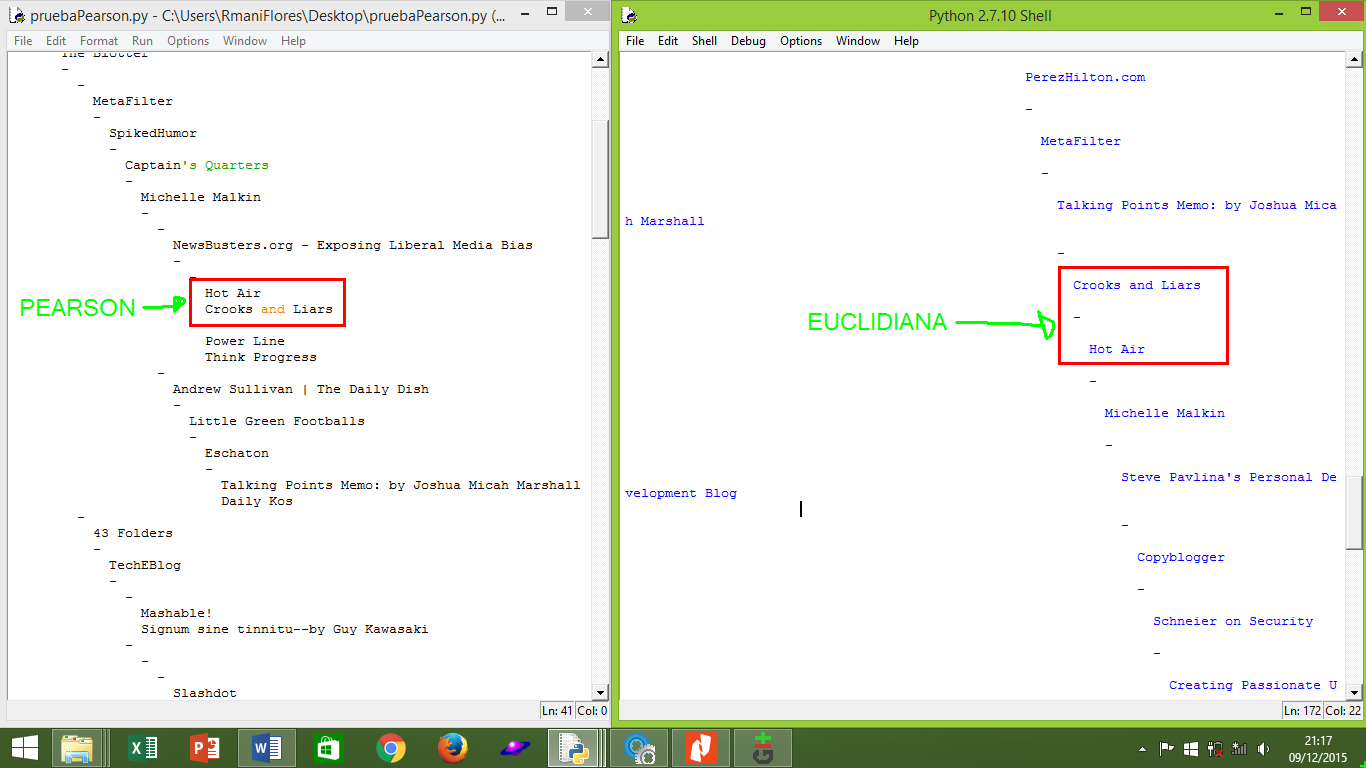
1. Pruebe usando la distancia euclidiana para clustering de los blogs. ¿Cómo cambia esto los resultados?

Figura 1. Comparando distancias de Pearson y Euclidiana

1. Investigue acerca de la distancia de Manhattan. Cree una función para esta y vea cómo cambia los resultados.

def manhattan(v1, v2):

suma=0

for i in range(len(v1)):

suma+=pow(pow(v1[i]-v2[i], 2), 0.5)

return suma

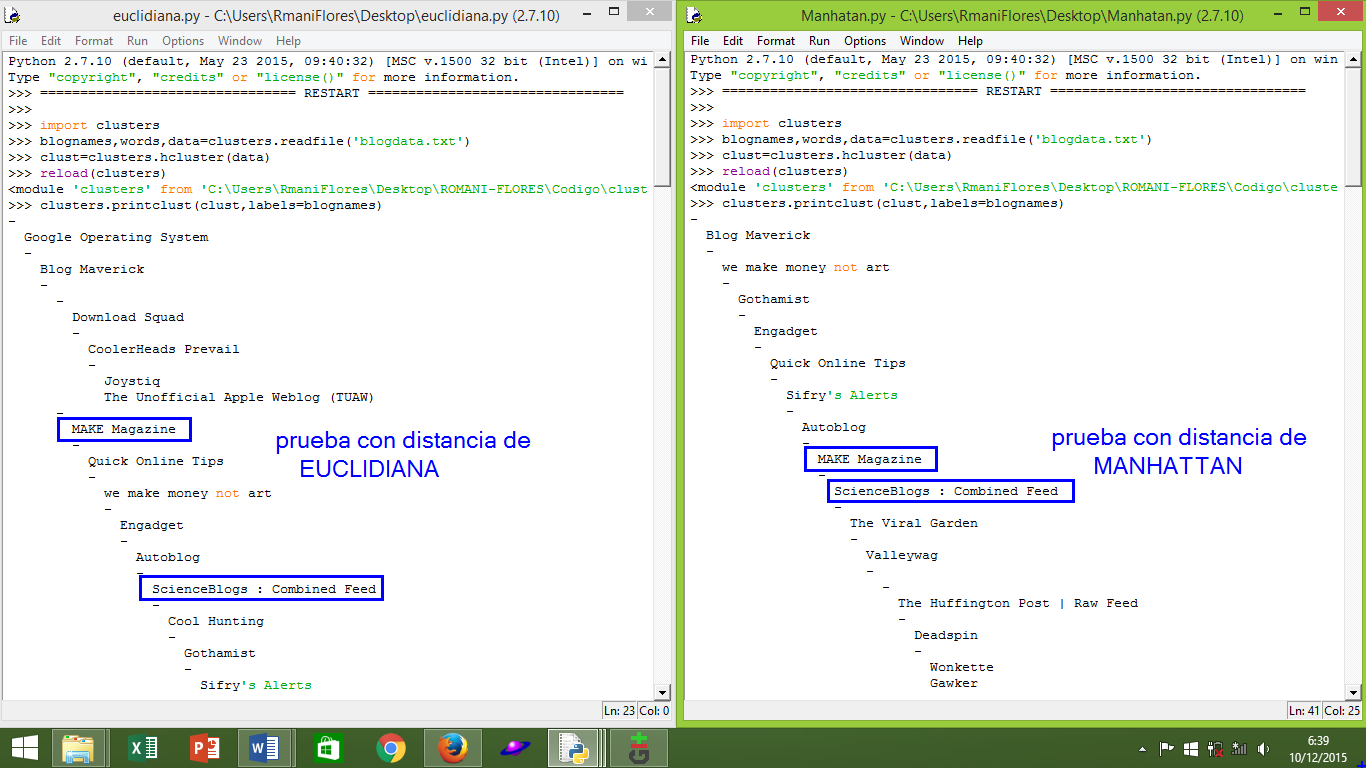
Como se observa en la Figura 2 hay diferencia de resultado entre las distancias Euclidiana y Manhattan.

Figura 2. Comparando distancias Euclidiana y Manhattan

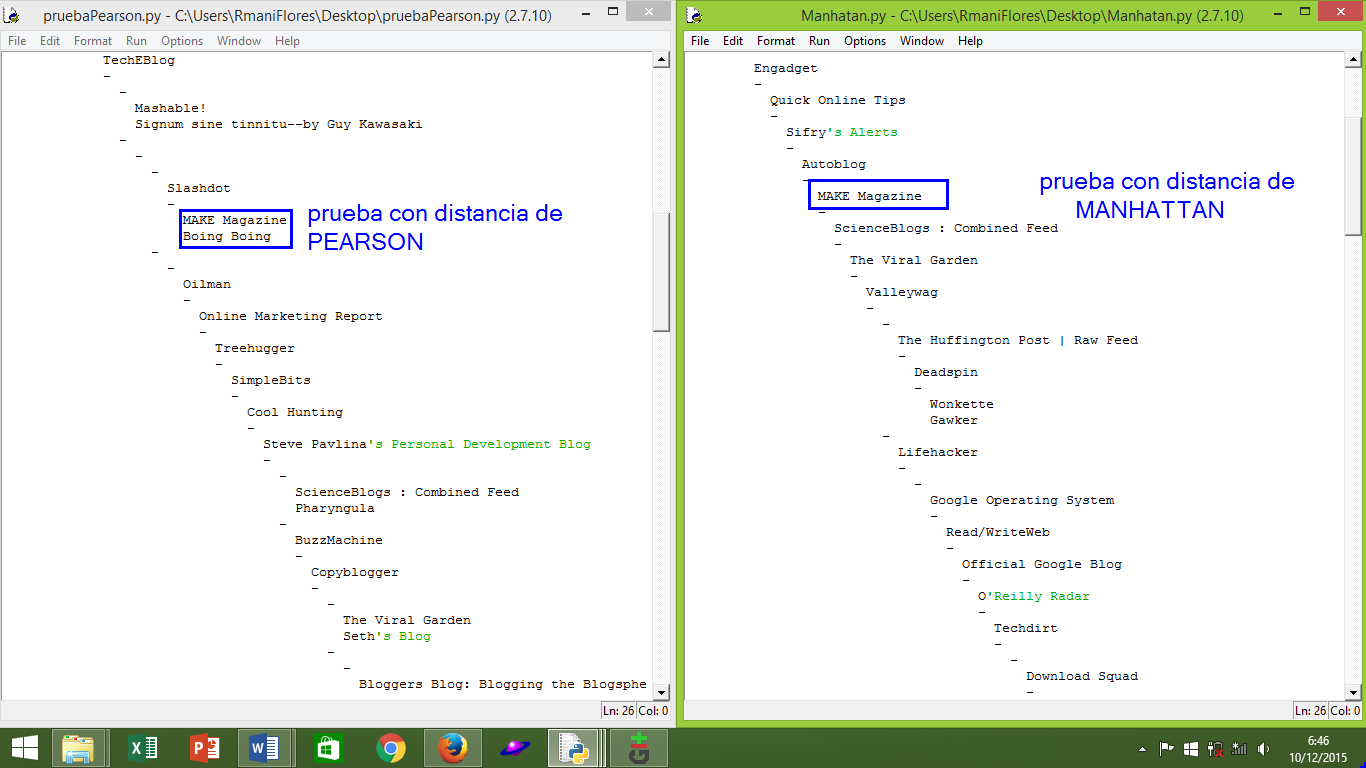
También en la figura 3 vemos la diferencia entre distancias de Pearson y Manhattan

Figura 3. Comparando distancias entre Pearson y Manhattan

1. Modifique la función del clustering K-means para que devuelva la distancia total ente todos los ítems de sus respectivos centroides.