Assignment Ten

Jacob Berlin

CS432 – Spring 2016

1. Using the data from A8:

- Consider each row in the blog-term matrix as a 500 dimension vector, corresponding to a blog.

- From chapter 8, replace numpredict.euclidean() with cosine as the distance metric. In other words, you'll be computing the cosine between vectors of 500 dimensions.

- Use knnestimate() to compute the nearest neighbors for both:

http://f-measure.blogspot.com/

http://ws-dl.blogspot.com/

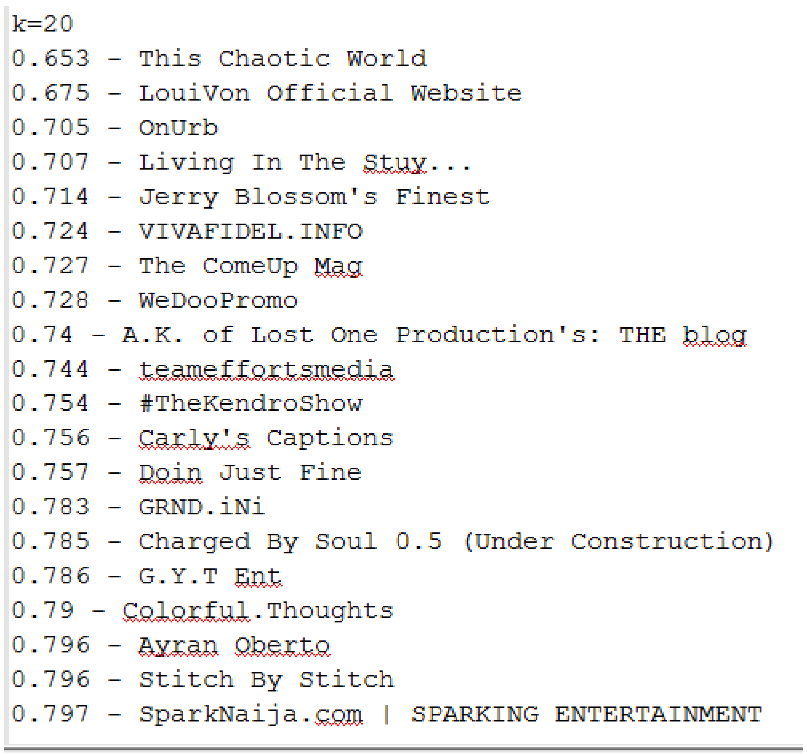
for k={1,2,5,10,20}.

To start this problem, I modified Programming Collective Intelligence's version of 'numpredict.py' and placed that code into 'clusters.py'. To make this work, I modified the Euclidean function to calculate the cosine distance and then modified the knnestimate to allow blog entries to be calculated. This is the output and code that I received/made:

```
blog 86 - f-measure
k=1
0.691 - Three Dudes Write a Blog
k=2
0.691 - Three Dudes Write a Blog
0.697 - MATH ROCK
k=5
0.691 - Three Dudes Write a Blog
0.697 - MATH ROCK
0.706 - A Taste of Vinum
0.716 - Double Trouble Mixtapes
0.727 - KrazySwagDJs
k=10
0.691 - Three Dudes Write a Blog
0.697 - MATH ROCK
0.706 - A Taste of Vinum
0.716 - Double Trouble Mixtapes
0.727 - KrazySwagDJs
0.736 - Hip Hop 2016
0.739 - Hip-Hop Overdose
0.745 - DJ A_DOG
0.75 - Hood Supastar
0.752 - WELCOME TO ABJ RICH TEENS BLOG
```

```
k=20
0.691 - Three Dudes Write a Blog
0.697 - MATH ROCK
0.706 - A Taste of Vinum
0.716 - Double Trouble Mixtapes
0.727 - KrazySwagDJs
0.736 - Hip Hop 2016
0.739 - Hip-Hop Overdose
0.745 - DJ A_DOG
0.75 - Hood Supastar
0.752 - WELCOME TO ABJ RICH TEENS BLOG
0.755 - Dispikable Mix Sessions
0.764 - Beth Sturgess' Blog
0.765 - Double Trouble Mixtapes
0.77 - To The Break Of Dawn
0.779 - BEATS OF ART
0.787 - Gang$ta Life
0.788 - -Dope Music Daily-
0.791 - MagVit
0.792 - Fresh: Hip-Hop & R&B
0.795 - SOLO138
```

```
Blog 6 – Web Science and Digital Libraries Research Group
k=1
0.653 – This Chaotic World
k=2
0.653 – This Chaotic World
0.675 – LouiVon Official Website
k=5
0.653 – This Chaotic World
0.675 – LouiVon Official Website
0.705 – OnUrb
0.707 – Living In The Stuy...
0.714 – Jerry Blossom's Finest
k=10
0.653 – This Chaotic World
0.675 – LouiVon Official Website
0.705 – OnUrb
0.707 – Living In The Stuy...
0.714 – Jerry Blossom's Finest
0.724 – VIVAFIDEL.INFO
0.727 – The ComeUp Mag
0.728 – WeDooPromo
0.74 – A.K. of Lost One Production's: THE blog
0.744 – teameffortsmedia
```



```
k=20
0.653 – This Chaotic World
0.675 – LouiVon Official Website
0.705 – OnUrb
0.707 – Living In The Stuy...
0.714 – Jerry Blossom's Finest
0.724 – VIVAFIDEL.INFO
0.727 – The ComeUp Mag
0.728 – WeDooPromo
0.74 – A.K. of Lost One Production's: THE blog
0.744 – teameffortsmedia
0.754 – #TheKendroShow
0.756 – Carly's Captions
0.757 – Doin Just Fine
0.783 – GRND.iNi
0.785 – Charged By Soul 0.5 (Under Construction)
0.786 – G.Y.T Ent
0.79 – Colorful.Thoughts
0.796 – Ayran Oberto
0.796 – Stitch By Stitch
0.797 – SparkNaija.com | SPARKING ENTERTAINMENT
```

```
def euclidean(v1,v2):
  numerator = sum(a*b for a,b in zip(v1,v2))
  denominator = square_rooted(v1)*square_rooted(v2)
  return 1 - round(numerator/float(denominator),3)


def getdistances(data,vec1):
  distancelist=[]

  # Loop over every item in the dataset
  for i in range(len(data)):
    vec2=data[i]

    # Add the distance and the index
    distancelist.append((euclidean(vec1,vec2),i))

  # Sort by distance
  distancelist.sort()
  return distancelist

def knnestimate(data,vec1,k=5):
  # Get sorted distances
  dlist=getdistances(data,vec1)
  avg=0.0
  # Take the average of the top k results
  for i in range(k):
    idx=dlist[i][0]
    print(dlist[i][0])
    #avg+=data[idx][1]
  #avg=avg/k
  return avg
```

2. Rerun A9, Q2 but this time using LIBSVM. If you have n categories, you'll have to run it n times. For example, if you're classifying music and have the categories: metal, electronic, ambient, folk, hip-hop, pop you'll have to classify things as: metal / not-metal electronic / not-electronic ambient / not-ambient etc.

Use the 500 term vectors describing each blog as the features, and your manually assigned classifications as the true values. Use 10-fold cross-validation (as per slide 46, which shows 4-fold cross-validation) and report the percentage correct for each of your categories.

For this question, I ran the four categories that I had through LIBSVM –

	Percent Cross-Validation	Percent Correct
1. HowTo –	48%	32%
2. Competitions –	83%	85%
3. Sales –	64%	58%
4. Information –	76%	52%