Bo Zhang
01063214

1 KNN

Using the data from A8:
- Consider each row in the blog-term matrix as a 1000 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 1000 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}.

Algorithm:

1. Open "blogdata(allFilter).txt" to read the Blog data.
2. Modify the script from "Programming Collective Intelligence" to replace euclidean() with cosine as the distance metric and replace the return of knnestimate() with the names of K "nearest neighbors".
3. Use the script from "Programming Collective Intelligence" to get K "nearest neighbors" of "F-Measure" and "Web Science and Digital Libraries Research Group".

Source code:

Listing 1: The content of Q1.py

```python
import math

def cosine(v1,v2):
    x=0.0
    for i in range(len(v1)):
        x+=v1[i]**2

    y=0.0
    for i in range(len(v2)):
        y+=v2[i]**2

    z=0.0
    for i in range(len(v1)):
        z+=(v1[i]-v2[i])**2

    return (x+y-z)/2/math.sqrt(x*y)

def getdistances(data,vec1):
    distancelist=[]
    for i in range(len(data)):
        vec2=data[i]['input']
        distancelist.append((cosine(vec1,vec2),i))
    distancelist.sort(reverse=True)
    return distancelist

def knnestimate(data,vec1,k=3):
    # Get sorted distances
    dlist=getdistances(data,vec1)

    result=[]
    for i in range(k):
        idx=dlist[i][1]
        result.append(data[idx]['Blog'])
    return result


f=open('blogdata(allFilter).txt','r',encoding='utf-8')
lines=f.readlines()
f.close()
data=[]
for i in range(1,len(lines)):
    line=lines[i].strip().split('\t')
    row={}
    row.setdefault('Blog',line[0])
    row.setdefault('input',[])
    for j in range(1,len(line)):
        row['input'].append(int(line[j]))
    data.append(row)
    if row['Blog']=='F-Measure':
        fMeasure=row['input']
    if row['Blog']=='Web Science and Digital Libraries Research Group':
        WSDLRG=row['input']

print(knnestimate(data,fMeasure,1))
print(knnestimate(data,fMeasure,2))
print(knnestimate(data,fMeasure,5))
print(knnestimate(data,fMeasure,10))
print(knnestimate(data,fMeasure,20))
print(knnestimate(data,WSDLRG,1))
print(knnestimate(data,WSDLRG,2))
print(knnestimate(data,WSDLRG,5))
print(knnestimate(data,WSDLRG,10))
print(knnestimate(data,WSDLRG,20))
```

Results:

Table 1: KNN Result

K	F-Measure	Web Science and Digital Libraries Research Group
1	F-Measure	Web Science and Digital Libraries Research Group
2	F-Measure	Web Science and Digital Libraries Research Group
	My Name Is Blue Canary	Anthems and Atleticos
5	F-Measure	Web Science and Digital Libraries Research Group
	My Name Is Blue Canary	Anthems and Atleticos
	KiDCHAIR	The Ideal Copy
	One Base on an Overthrow	Pithy Title Here
	SunStock Music	Sonology
10	F-Measure	Web Science and Digital Libraries Research Group
	My Name Is Blue Canary	Anthems and Atleticos
	KiDCHAIR	The Ideal Copy
	One Base on an Overthrow	Pithy Title Here
	SunStock Music	Sonology
	Everything Flows	ELLIA TOWNSEND A2
	Some Call It Noise....	SEVEN1878
	Indie Top 20 - The Blog!	macthemost
	.	Indie Top 20 - The Blog!
	Eli Jace — The Mind Is A Terrible Thing To Paste	The Great Adventure 2016
20	F-Measure	Web Science and Digital Libraries Research Group
	My Name Is Blue Canary	Anthems and Atleticos
	KiDCHAIR	The Ideal Copy
	One Base on an Overthrow	Pithy Title Here
	SunStock Music	Sonology
	Everything Flows	ELLIA TOWNSEND A2
	Some Call It Noise....	SEVEN1878
	Indie Top 20 - The Blog!	macthemost
	.	Indie Top 20 - The Blog!
	Eli Jace — The Mind Is A Terrible Thing To Paste	The Great Adventure 2016
	Swinging Singles Club	SunStock Music
	www.doginasweater.com Live Show Review Archive	Steel City Rust
	the roofy leak	Hip In Detroit
	no gift for the gab	STANLEY SAYS
	Bleak Bliss	Mile In Mine
	ALL MY BROTHERS AND SISTERS	hello my name is justin.
	Steel City Rust	from a voice plantation
	Myopiamuse	.
	Hip In Detroit	no gift for the gab
	SPACE KiDZ a GoGo	Everything Flows

2 SVM

Rerun A9, Q2 but this time using LIBSVM. If you have n categories, you'll have to run it n times.

Use the 1000 term vectors describing each blog as the features, and your mannally assigned classifications as the true values. Use 10-fold cross-validation and report the percentage correct for each of your categories.

Algorithm:

1. Use the script from A8 to generate entries vectors, but do not filter the wordlist with frequencies and only generate entries in "entries.txt". Save the entries vectors into "feeddata.txt".

2. Open "feeddata.txt" and "entries.txt" to read the title, word frequency vector and category of each entry.

3. Generate 4 lists of categories based on the category list. For each category, use 10-fold cross-validation to split the categories lists and word frequency vectors into training group and testing group.

4. For each split, use the script from "Programming Collective Intelligence" to train the SVM on the training group data, then use SVM to guess the classification of the testing group data.

5. For each category, compute the average accuracy of all the splits.

Source code:

Listing 2: The content of generatefeedvector.py

```
import feedparser
import re

# Returns title and dictionary of word counts for an RSS feed
def getwordcounts(url):
    # Parse the feed
    d=feedparser.parse(url)
    print (d.feed.title)
    feed=[]

    # Loop over all the entries
    for e in d.entries:
        print (e.title)

        wc={}
        summary=''
        if 'content' in e:
            for content in e.content:
                print (content)
                print ()
                summary=summary+content.value+'␣'
        elif 'summary' in e and e.summary!='':
            print (e.summary)
            print ()
            summary=summary+e.summary+'␣'
        elif 'description' in e and e.description!='':
            print (e.description)
            print ()
            summary=summary+e.description+'␣'
        else:
            print ('error!!!!!!!!!!!!!!')
            print ()

        # Extract a list of words
        words=getwords(e.title+'␣'+summary)
        for word in words:
            wc.setdefault(word,0)
            wc[word]+=1

        feed.append([e.title,wc])
    return feed

def getwords(html):
    # Remove all the HTML tags
    txt=re.compile(r'<[^>]+>').sub('',html)

    # Split words by all non-alpha characters
    words=re.compile(r'[^A-Z^a-z]+').split(txt)

    # Convert to lowercase
    return [word.lower( ) for word in words if word!='']

apcount={}
wordcounts={}
feedlist=[line.strip() for line in open('feedlist.txt','r')]
for feedurl in feedlist:
    print (feedurl)
    for feed in getwordcounts(feedurl):
        wordcounts[feed[0]]=feed[1]

        for word,count in feed[1].items():
            apcount.setdefault(word,0)
            if count>1:
                apcount[word]+=1

wordlist=[]
for w,bc in apcount.items( ):
    wordlist.append([bc,w])

wordlist.sort(reverse=True)

f=open('entries.txt','r',encoding='utf-8')
lines=f.readlines()
```

```
f.close()
entries=[]
for i in range(1,len(lines)):
    entries.append(lines[i].strip().split('\t')[0])

out=open('feeddata.txt','a',encoding='utf-8')
out.write('Title')
for i in range(min(len(wordlist),1000)): out.write('\t%s' % wordlist[i][1])
out.write('\n')
for blog,wc in wordcounts.items( ):
    if blog in entries:
        out.write(blog)
        for i in range(min(len(wordlist),1000)):
            if wordlist[i][1] in wc: out.write('\t%d' % wc[wordlist[i][1]])
            else: out.write('\t0')
        out.write('\n')
out.close()
```

Listing 3: The content of Q2.py

```
from svm import *
from svmutil import *

f=open('entries.txt','r',encoding='utf-8')
lines=f.readlines()
f.close()
entries=[]
for i in range(1,len(lines)):
    entries.append(lines[i].strip().split('\t'))
entries.sort()

f=open('feeddata.txt','r',encoding='utf-8')
lines=f.readlines()
f.close()
feeddata=[]
for i in range(1,len(lines)):
    feeddata.append(lines[i].strip().split('\t'))
feeddata.sort()

answers=[]
inputs=[]
categories=[]
for i in range(len(entries)):
    answers.append(entries[i][1])
    if entries[i][1] not in categories:
        categories.append(entries[i][1])
    inputs.append(list(map(int,feeddata[i][1:len(feeddata[i])])))

cat=[]
for i in range(len(categories)):
    cat.append([])
    for ans in answers:
        if ans==categories[i]:
            cat[i].append(1)
        else:
            cat[i].append(0)


svm_model.predict = lambda self, x: svm_predict([0], [x], self)[0][0]
param = svm_parameter()
param.kernel_type = RBF

for i in range(len(categories)):
    print ('\n'+categories[i]+':')
    acc=0
    for j in range(10):
        trainCat=[]
        trainInputs=[]
        testCat=[]
        testInputs=[]
        for k in range(len(cat[i])):
            if k<(j+1)*len(cat[i])/10 and k>=j*len(cat[i])/10:
                testCat.append(cat[i][k])
                testInputs.append(inputs[k])
            else:
                trainCat.append(cat[i][k])
                trainInputs.append(inputs[k])

        prob = svm_problem(trainCat,trainInputs)
        m = svm_train(prob, param)
        p_labels, p_acc, p_vals = svm_predict(testCat,testInputs, m)
        acc+=p_acc[0]
    print ('Average␣Accuracy␣=␣{}'.format(acc/10)+'%')
```

Results:

Table 2: SVM Prediction Performance Assessment

Dance	Music	Books	Movies
83.0%	75.0%	70.0%	72.0%