There are two parts of the project. One is image classification and one is text classification.

Environment:

Windows 8.1

Python 2.7

libs:

nltk

pythonxy(contains scipy, numpy ….)

libsvm

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The first part includes following files:

crawler\_pic.py

index.py

ExtractSkinFeature.py

SkinDector.py

picture\_classify.py

skinmodel.bin

training-data.csv

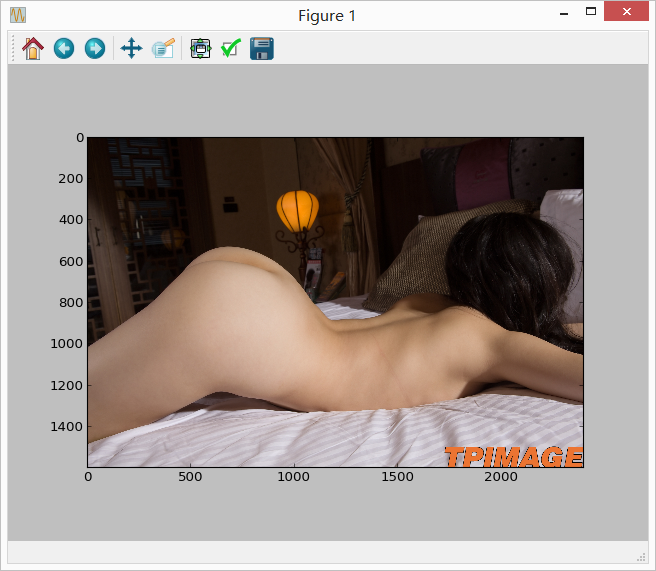
This part one, I use svm to classify the porn picture and text picture.

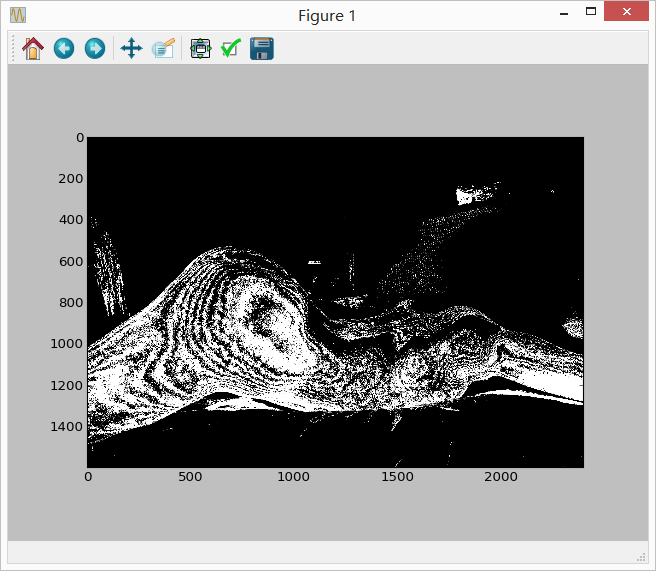
If you want to implement this program ,just double click picture\_classify.py then it will return result.

Now I would introduce each file’s function.

SkinDetector.py is used to detect skin in the picture. First read skinmodel.bin which is a skin model file I download from web. Then read the picture and check every color pixel. Then we can determine whether a color pixel is skin or not by RGB values.

ExtractSkinFeature.py is used to extract skin feature from color pixels. Those features including percentage of color pixels that are considered as skin; mean of RGB color value of skin pixels; maximum area of the skin section.

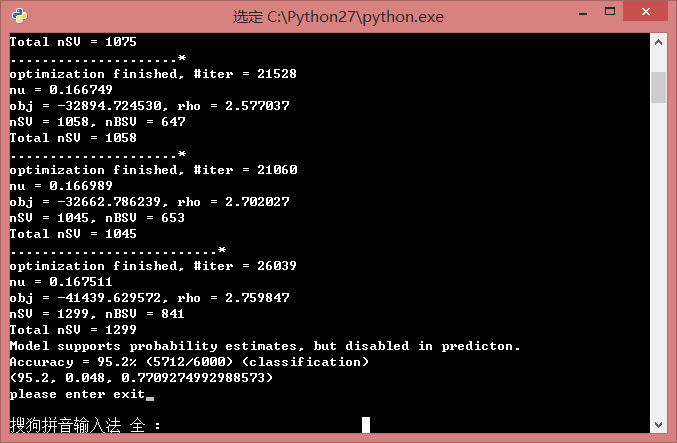




crawler\_pic.py is used to crawl images on the website. Use multiple threads to make it a little faster.

index.py is used to create a index of the features of picture. It reads every picture and extract features of them then put the features into csv file.

picture\_classify.py is used to classify the pictures. First it reads data from csv file, separate them into test data and training set. Then normalize them and put them into svm model. Porn pictures are set as ‘1’ and normal pictures are set as ‘0’. The final accuracy is about 95%.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Second part is about text classification, which is described at the project idea #11. I implement 2 algorithms (KNN and NB) and compare the accuracy and speed of them.

Bayes\_text1.py is used to pre-processing the text data. It contains that remove stopwords, non-English words and non-number. Also it use Porter algorithm to reduce variant forms of a word to base form.

Bayes\_text2.py is used to calculate freq of every word and remove words which appear less than 4 times. Because I consider words appear more than 4 times as feature words.

Bayes\_text3.py is used to separate data set as test set and training set. The rate is about 1:9.

Bayes\_text4.py is the major work of NB algorithm. getCateWordProb() checks frequency of every word in every doc and the total number of words in a doc. it returns cateWordProb {category\_word: frequency} and cateWordNum {category: total number of words}. I use multinomial model to calculate. Conditional probability = (# of a word in a category + 0.0001)/(# of all words in this category + # of words in this doc) . Prior probability=(# of all words in a category)/(# of all words in this doc). After calculation,it will calculate accuracy.

output:

WordMap size32189

Cate\_Word: 162649

cate alt.atheism contains 130141.0

cate comp.graphics contains 145322.0

cate comp.os.ms-windows.misc contains 348719.0

cate comp.sys.ibm.pc.hardware contains 96505.0

cate comp.sys.mac.hardware contains 88902.0

cate comp.windows.x contains 131896.0

cate misc.forsale contains 75843.0

cate rec.autos contains 109281.0

cate rec.motorcycles contains 99047.0

cate rec.sport.baseball contains 111705.0

cate rec.sport.hockey contains 135429.0

cate sci.crypt contains 147705.0

cate sci.electronics contains 101945.0

cate sci.med contains 153708.0

cate sci.space contains 135170.0

cate soc.religion.christian contains 174490.0

cate talk.politics.guns contains 155503.0

cate talk.politics.mideast contains 219330.0

cate talk.politics.misc contains 162621.0

cate talk.religion.misc contains 103775.0

totalWordsNum: 2827037.0

rightCount: 1513.0resultCate: 1870

The accuracy for Naive Bayesian Classifier in 0th Exp is :0.8090909090909091

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

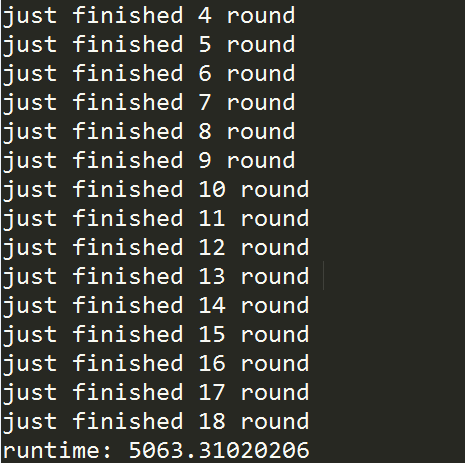
The other algorithm is knn.

knn1.py will use the data which is already pro-processed by Bayes\_text1.py. It will calculate IDF value of every word then write the value into file IDFPerWord.

knn2.py will create index: doc\_vector file. The form is like <category, doc, (word1, tdidf1), (word2,tdidf2),….> Then it will separate data sets as training set and test set. The rate is about 9:1.

knn3.py is main process of the knn algorithm. KNNComputeCate() return the category which has the closest distance from test file. This function use computeSim() to calculate sim value.

output:



wordMap size: 89937

newWordMap size: 27648

right count: 1472 , count: 1890 , accurancy: 0.778836