CS 440/ECE 448 - 3 Hours
Spring 2015
Assignment 3: Naive Bayes Classification

Instructor: Svetlana Lazebnik

Harrison Kiang - hkiang2, Jonathan Park - jgpark2, Gabriella Quirini - gquirin2

Part 1: Digit Classification

In this section, we explore using the Naive Bayes model for classifying images. The images used were examples of numbers 0-9 written in different styles of handwriting. Most of them were easily classifiable to the human eye, but as we know, accurate image classification is a challenge to implement effectively even on the most powerful computer systems. Both parts of this section involve breaking down a given image into a set of features. These features define the image as a whole, but individually are nothing more than a binary value representing foreground (the area of the digit itself), and background (the rest of the 28x28 pixel space). Part 1.1 explores using single pixels as features. Later on, for extra credit we expanded the images to handle ternary values (foreground, background, middleground) for an overall improved accuracy with interesting results.

Part 1.1 - Single Pixels as Features

Using this approach, each image is looked at pixel by pixel and broken down into a 2-dimensional list of binary values. In order to achieve this, we created a Digit class. Each Digit holds its own feature list, as well as the ASCII “image” (#/+) representation of itself. Additionally, a ‘proper class’ attribute holds the classification given to it post-initialization. In order to obtain the feature list, we simply parse the ASCII list and append 1’s and 0’s to our feature list according to foreground and background characters respectively. We use this method to set up both our training and testing data for use. For the ternary version implementation, we assign 1’s to #, 0’s to blanks, and 0.8’s to +’s. We have experimented with different middle ground values, but found 0.8 to improve the overall accuracy more so than 0.1~0.7 or 0.9.

Training

Training data refers to the set of images we already know the classifications for. We “train” our algorithm to recognize which class an unknown digit belongs to this way. We first set up a “digit database” of sorts, which is a 2-dimensional list of Digit objects. It’s first dimension is of size 10, with each index corresponding to a digit class 0-9. Our training occurs by populating this database with the known Digit objects. Our resulting list is used to calculate feature likelihoods for each digit. The result is 10 28x28 lists whose values correspond to that pixel’s likelihood of being in the foreground for the given digit. Laplace smoothing is used to ensure no zero-counts, and the priors are estimated based on the occurrences of training samples for each class. For the k constant of Laplace smoothing, we chose k=1 to get the best accuracy.

Testing

We can now use our trained data set to classify unknown images using maximum a posteriori classification. This method utilizes the likelihood matrix and priors for each number class to generate a probability that the test image belongs to a given class. Looking at each pixel of the test image individually, we determine whether or not the test image fits a class based on the probability that the current pixel matches the corresponding pixel of that class. The prior term comes into play by acting as an overall likelihood of that digit occurring at all. Omitting this term turns it into a maximum likelihood classification. Using the same training and testing data, MAP

classification yields a 77.1% accuracy rating, while maximum likelihood yields 77.0%. The difference comes from accounting for the prior, and this gap would be larger had the training set been less balanced in terms of occurrences for each digit, and the testing set adjusted to match. However, as seen from the overall accuracy, the removal of prior terms for Maximum Likelihood seems to present little difference than MAP classification.

Evaluation		
Classification Accuracy		Priors
Class 0	84.44%	0.0958
Class 1	96.30%	0.1126
Class 2	77.67%	0.0976
Class 3	79.00%	0.0986
Class 4	76.64%	0.1070
Class 5	67.39%	0.0868
Class 6	75.82%	0.1002
Class 7	72.64%	0.1100
Class 8	60.19%	0.0924
Class 9	80.00%	0.099

Confusion Matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	.84	.00	.01	.00	.01	.06	.03	.00	.04	.00
1	.00	.96	.01	.00	.00	.02	.01	.00	.00	.00
2	.01	.03	.78	.04	.01	.00	.06	.01	.05	.02
3	.00	.02	.00	.79	.00	.03	.01	.06	.02	.06
4	.00	.01	.00	.00	.77	.00	.03	.01	.02	.17
5	.02	.02	.01	.13	.03	.67	.01	.01	.02	.07
6	.01	.07	.04	.00	.04	.05	.76	.00	.02	.00
7	.00	.06	.03	.00	.03	.00	.00	.73	.03	.13
8	.02	.01	.03	.14	.02	.06	.00	.01	.60	.12
9	.01	.01	.01	.03	09.	.02	.00	.02	.01	.80

orange = pairs most likely confused

With the ternary version, MAP classification overall accuracy gives 77.8% and ML gives 78%. It is interesting to note the differences caused by adding the middleground.

Class Number	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Accuracy %	85.6	95.4	77.7	83.0	73.8	64.1	78.0	70.8	66.9	82.0

Certain classes get a boost in accuracy, whereas other classes tend to lose some. Overall, the accuracy is 0.7~1.0% better.

Odds Ratios:

One way of inspecting what our program has “learned” is by generating images based off of the likelihood matrix. Another is by using odds ratios. Given classes c_1, c_2, the odds ratio is a matrix formed by comparing corresponding pixels if the following manner:

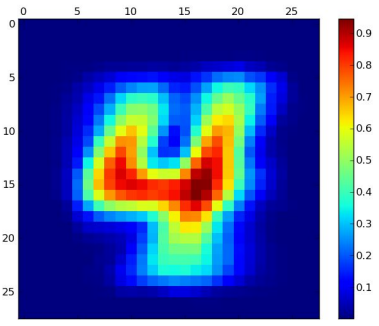
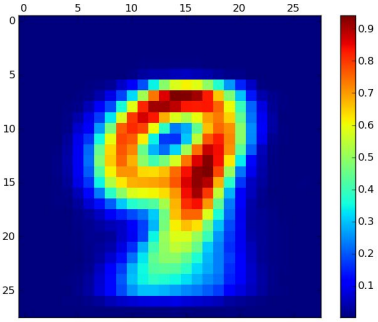
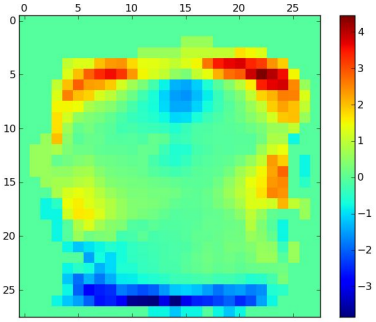
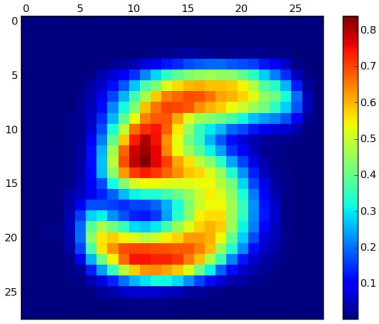
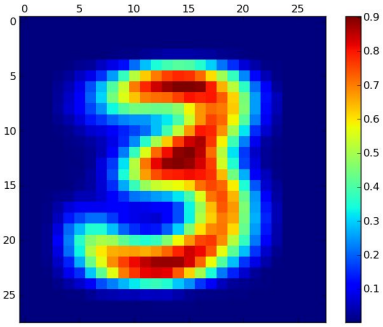
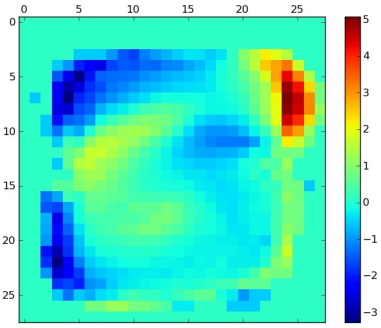
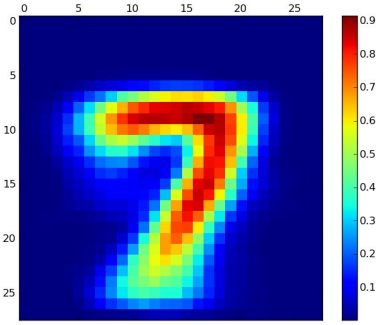
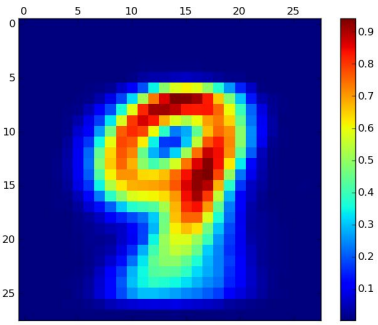
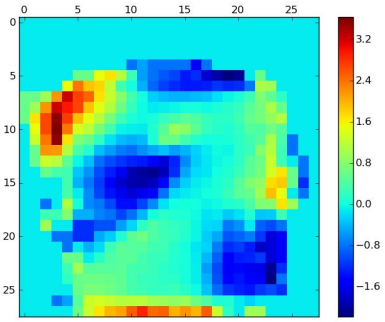
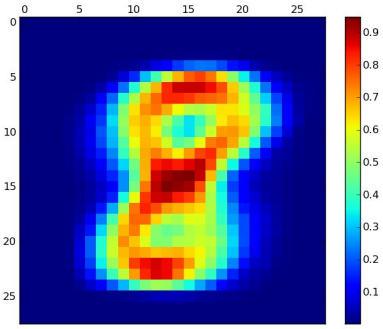
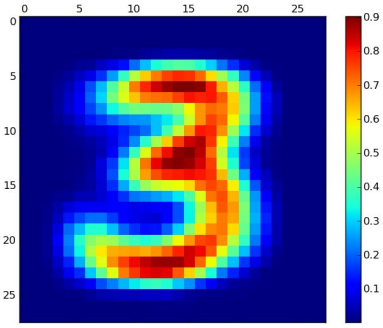
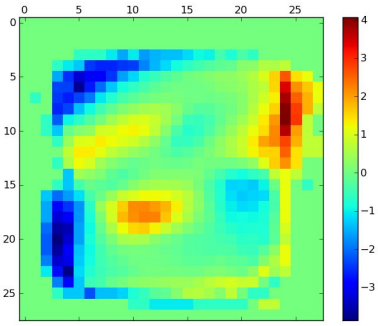
$odds(F_{ij}=1, c_1, c_2) = P(F_{ij}=1 \mid c_1) / P(F_{ij}=1 \mid c_2)$

Here we are essentially mapping the areas that indicate whether an unknown belongs to c_1 over c_2. If the unknown digit’s foreground is contained within these “hot spots” (illustrated below), we can say that it is more likely to be a member of c_1 over c_2.

The odds ratios printed in ASCII was easy to generate, but a bit unintuitive:
Matrix 2 comparing class 5 with class 3: (5 over 3)

```
++++++++++++++++++++++++++++
++++++++++++++++++++++++++++
++++++++++++++++++++++++++++
+++++--------------+ +++++++
+++----------------+ +++++++
+++----------------++ ++++++
+++----------------++ +++++
+-+----------------+  ++++
+++--------+++++----++ ++++
++------++    +------+ ++++
++-----   ++ ++-------+ ++ +
+++-++ +++  ++---------+++ +
+++++ +++  ++----------   ++
+++-++++  ++---------+++ +++
+++++ +  +++---------+-+ +++
+++++    ++++---------++  -+
++--++++++++++++-------+  ++
++----+++++++ +++------+  ++
++----++++++  +++------+  ++
++----+++++  +++-------+  ++
++-----++++++++--------+++++
++-------+++----------+++++
++--------------------+  +++
++-------------------+   +++
+++-------------++++ +++++++
+++------++++-++++-----+++++
++++++      ++--+++---++++++
++++++++++++++++++++++++++++
```

So we went through the extra effort of translating this into generating colored heat maps.
Below are heat maps corresponding to the feature likelihoods and odds ratios of the four most commonly confused digits based on our confusion matrix above.

Likelihood Heat Map c_1	Likelihood Heat Map c_2	Log Odds for c_1 over c_2
		
		
		
		

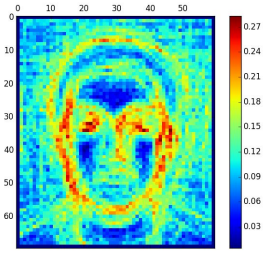
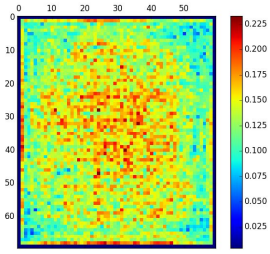
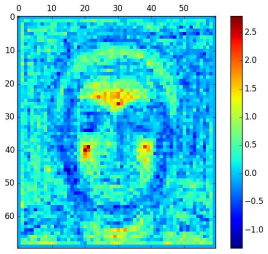
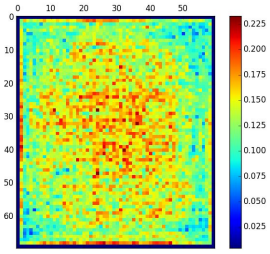
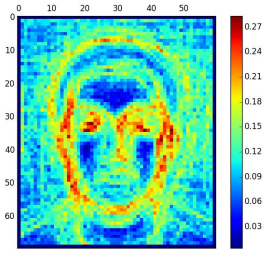
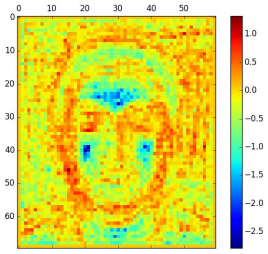
Faces:

Our above naive bayes classifier was modified to classify faces. Each face image is either a face (1) or a ‘non-face’ (0). Each face image is 70x60 and is populated by ‘#’ and ‘ ‘ characters, representing edge pixels non-edge pixels, respectively. The methods of training and classification are the same as above.

Training:

Number of training non-face images : 234
Number of training face images : 217
There are 451 numbers in training set
Prior for non-face class : 0.518847006652
Prior for face class : 0.481152993348
Overall accuracy(MAP): 90.6666666667 %
Overall accuracy (ML): 90.6666666667 %
Confusion Matrix:
0.8831 0.1169
0.0685 0.9315
Accuracy for non-face class : 88.31 %
Accuracy for face class : 93.15 %

Likelihood Heat Map c_1 **Likelihood Heat Map c_2** **Log Odds for c_1 over c_2**

Part 2: Text Document Classification

This section aims to apply Naive Bayes method for classifying written documents. The datasets provided are preprocessed into “bag of words” representation, each containing keywords, their labels, and number of occurrences. Labels correspond to the category that word belongs to, and are represented by numbers. In the email dataset, 0 and 1 refer to normal and spam respectively, while numbers 0-7 correspond to the categories of the newsgroup dataset. We begin by creating dictionaries from the training data. Then, we estimate the conditional probability tables over those dictionaries for each class. We use Laplace smoothing to help classify words not found in our dictionary. Overall, the training and testing process is very similar to part 1 above.

Email Evaluation		
Overall Accuracy: 96.53%		
Category Classification Accuracy		Priors
Spam	94.61%	0.5493
Normal	98.46%	0.4507

Confusion Matrix

	0	**1**
0	.9461	.0538
1	.0153	.9846

8 Category Evaluation		
Overall Accuracy: 81.81%		
Category Classification Accuracy		Priors
sci.space	79.41%	0.1598
comp.sys.ibm.hardware	57.58%	0.0701
rec.sport.baseball	88.89%	0.1072
comp.windows.x	85.71%	0.1364
talk.politics.misc	97.87%	0.2034
misc.forsale	10.00%	0.0494
rec.sport.hockey	93.48%	0.1421
comp.graphics	82.76%	0.1313

Confusion Matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	.79	.00	.00	.00	.20	.00	.00	.00
1	.06	.57	.00	.12	.21	.00	.00	.03
2	.00	.00	.88	.00	.08	.00	.02	.00
3	.00	.00	.00	.85	.03	.00	.00	.10
4	.02	.00	.00	.00	.97	.00	.00	.00
5	.20	.10	.00	.00	.30	.10	.00	.30
6	.00	.00	.00	.00	.06	.05	.93	.00
7	.03	.00	.00	.03	.10	.00	.00	.82

orange = pairs most likely confused

Top 20 Words With Highest Log-Odds Ratios - Email

$c_1 = 0$ over $c_2 = 1$ (normal over spam)	$c_1 = 1$ over $c_2 = 0$ (spam over normal)
linguistic	sell
workshop	nbsp
abstract	ffa
theory	remove
syntax	capitalfm
grammar	bulk
chair	money
discourse	floodgate
translation	investment
movement	bonus
computational	aol
committee	profit
semantic	mailing
benjamin	save
verb	links
context	reports
programme	offshore
phonology	hundred
semantics	debt
nl	advertise

Top 20 Words With Highest Log-Odds Ratios - 8 Category

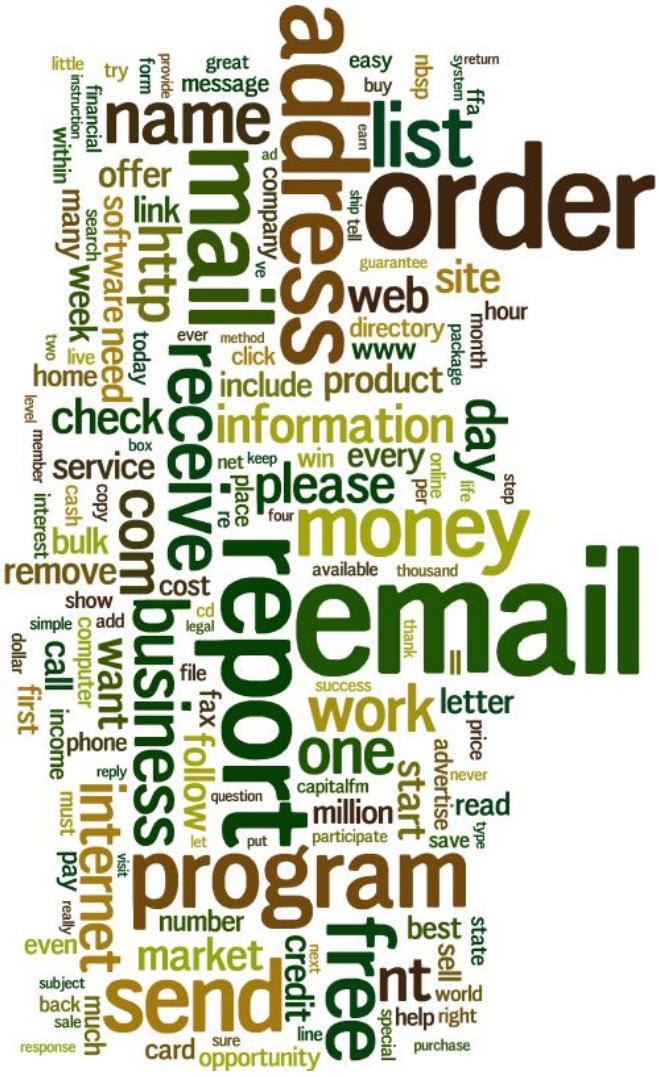
$c_1 = 5$ over $c_2 = 4$ (misc.forsale over talk.politics.misc)	$c_1 = 5$ over $c_2 = 7$ (misc.forsale over comp.graphics)	$c_1 = 1$ over $c_2 = 4$ (comp.sys.ibm.pc.hardware over talk.politics.misc)	$c_1 = 0$ over $c_2 = 4$ (sci.space over talk.politics.misc)
dos	wolverine	scsi	orbit
wolverine	comics	ide	shuttle
shipping	hulk	controller	spacecraft
comics	spiderman	bios	venus
spiderman	liefeld	disk	lunar
liefeld	sabretooth	dos	mars
sabretooth	app	isa	sky
rider	bagged	interface	satellites
bagged	hobgoblin	modem	solar
cd	xforce	floppy	hst
hobgoblin	punisher	motherboard	telescope
xforce	cable	hardware	propulsion
hulk	marvel	jumper	launch
disks	panther	rom	orbital
punisher	mutants	adaptec	orbiter
unix	pom	pc	jupiter
marvel	comic	vlb	moon
ufl	mint	card	saturn
panther	bwsmith	drive	kilometers
mutants	rider	cpu	probes

Cloudmaps (http://www.wordle.net/create)

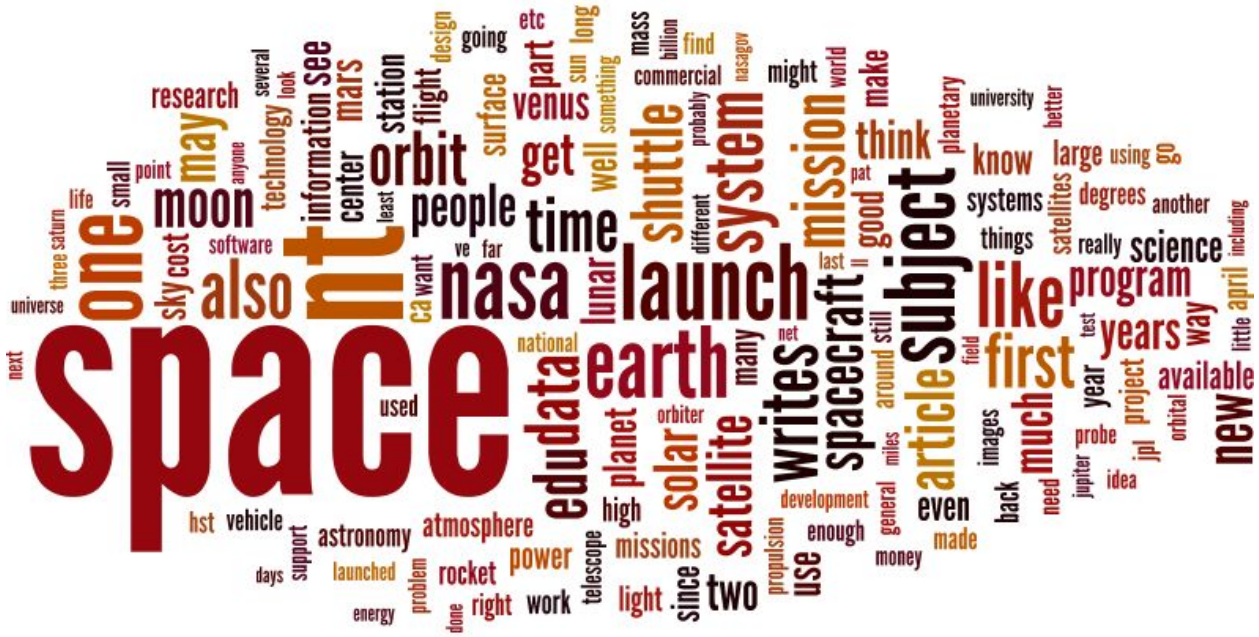
Each dictionary for each category (normal and spam for email, categories 0-7 for newsgroups) was traversed. Each word was added to a text file in a separate line and repeated on that same line according to the frequency. Each dictionary's word's frequency takes into account repeats and is based off of the given training data sets.

Email normal

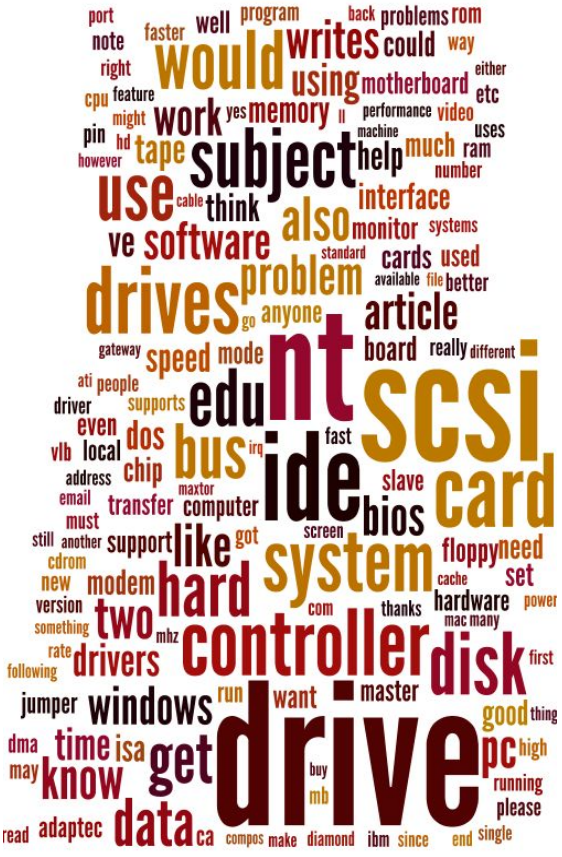
Email spam



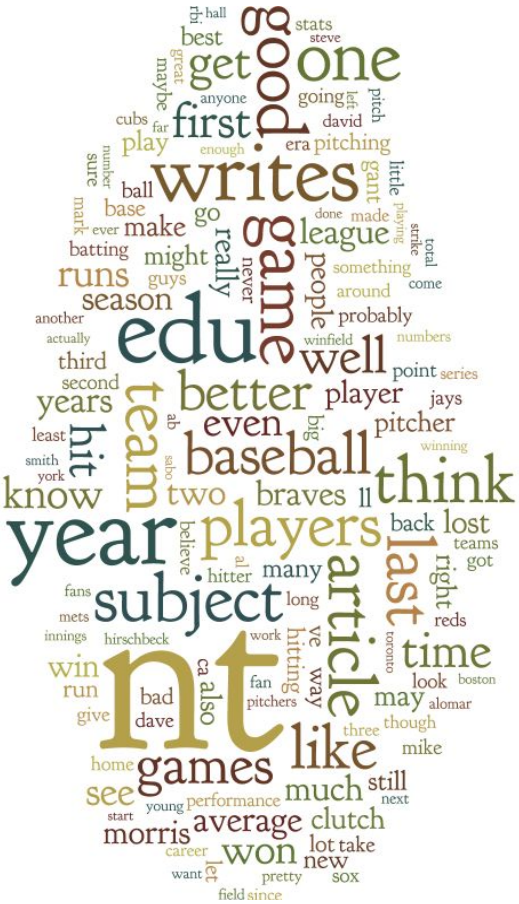
Category 0: sci.space



Category 1: comp.sys.ibm.pc.hardware



Category 2: rec.sport.baseball



Category 3: comp.windows.x

Category 4: talk.politics.misc

Category 5: misc.forsale

Category 6: rec.sport.hockey

Category 7: comp.graphics

Statements of Contribution:

Harrison: coding: parts 1 and 2, faces, cloudmaps, some report

Jon: debugging, coding part1: ternary, heatmaps, some report

Gabriella: coding part1, debugging, report