Sarin Madarasmi

Project 3 Report

Centroid-based Method

This method first represents the document vector into the specified feature-representation option. Then normalize each document vectors. After that, I compute the centroids from the training documents for each classifier. Since we are using one-vs-rest approach, I also compute centroids for each "rest" classifiers. Next, I compute threshold for each binary classifier by sorting objects in decreasing order according to prediction score. At each point in the ranked list, I compute the F1 score assuming that all objects above are +ve and all objects below are predicted -ve. The threshold is the value associated with the point that has the maximum F1 score. For each test document, I use the maximum prediction score against +ve centroids to decide which class each test document belongs to.

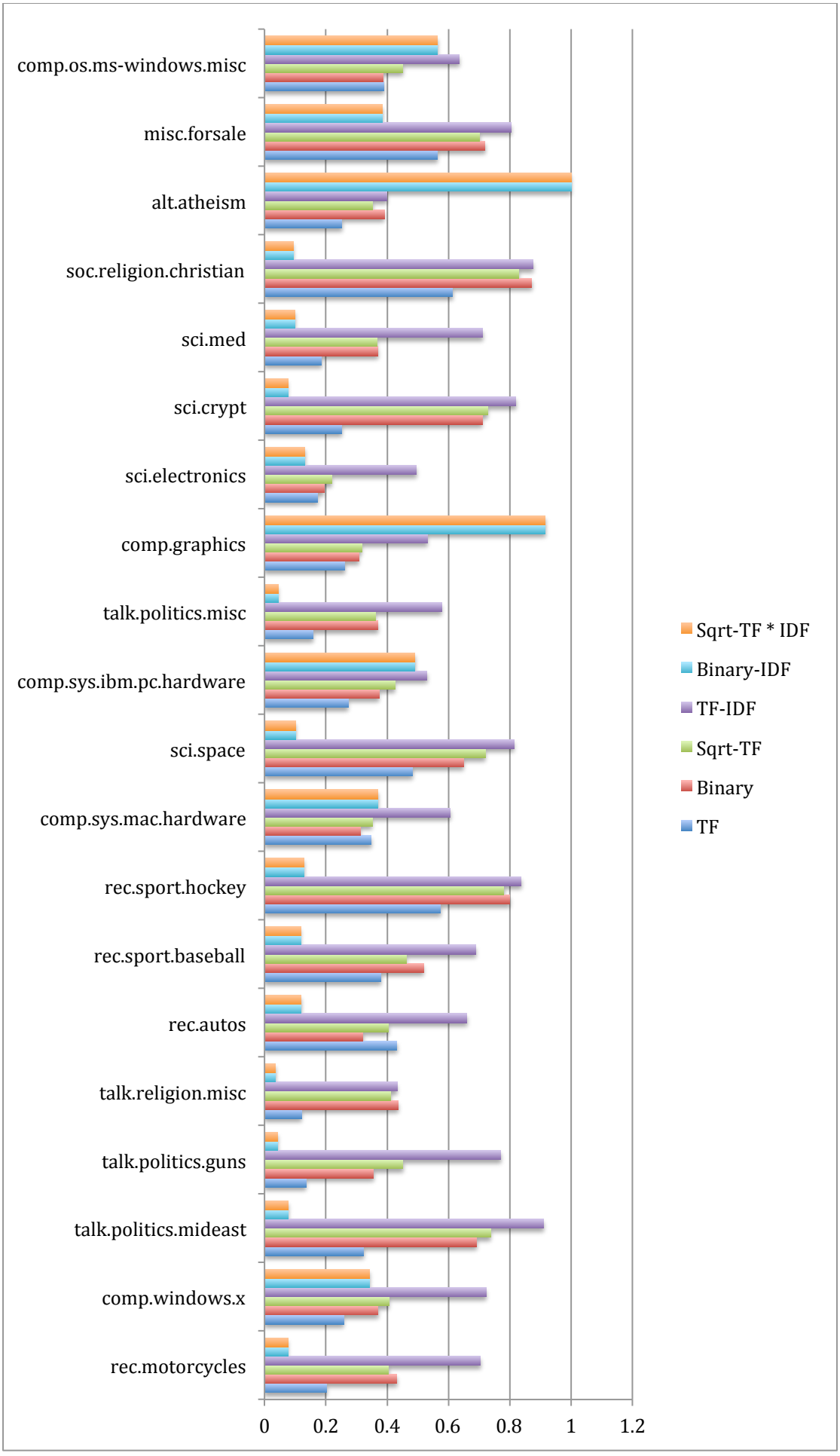
Example for running java program: java centroid input-file input-rlabel-file train-file test-file class-file features-label-file feature-representation-option output-file

I made all the representations, even the bonus ones.

Bag-of-words

**Below is the bar chart for bag of words centroid-based method. Below that will be all the tables and values that were used to construct the bar chart. **

Sarin Madarasmi



Sarin Madarasmi

TF Representation

rec.motorcycles	0.20359282
comp.windows.x	0.25787967
talk.politics.mideast	0.32360741
talk.politics.guns	0.13533837
talk.religion.misc	0.12121212
rec.autos	0.4312268
rec.sport.baseball	0.37974682
rec.sport.hockey	0.5740181
comp.sys.mac.hardware	0.34770516
sci.space	0.48360655
comp.sys.ibm.pc.hardware	0.27322403
talk.politics.misc	0.15811966
comp.graphics	0.26216215
sci.electronics	0.17370893
sci.crypt	0.25252524
sci.med	0.18579234
soc.religion.christian	0.61349696
alt.atheism	0.25210083
misc.forsale	0.5635739
comp.os.ms-windows.misc	0.38947368

Time of Execution: 7m12.689s

Binary Representation

rec.motorcycles	0.43076923
comp.windows.x	0.36995304
talk.politics.mideast	0.69135803
talk.politics.guns	0.35555556
talk.religion.misc	0.43609023
rec.autos	0.32075474
rec.sport.baseball	0.5189504
rec.sport.hockey	0.8
comp.sys.mac.hardware	0.3127572
sci.space	0.649635
comp.sys.ibm.pc.hardware	0.37378645
talk.politics.misc	0.3692308
comp.graphics	0.3068309
sci.electronics	0.19415645
sci.crypt	0.71005917
sci.med	0.36912748
soc.religion.christian	0.8716216
alt.atheism	0.3908046
misc.forsale	0.71910113
comp.os.ms-windows.misc	0.38563326

Sarin Madarasmi

Time of Execution: 7m18.127s

Square-root of TF Representation

rec.motorcycles	0.40366971
comp.windows.x	0.40591964
talk.politics.mideast	0.7376425
talk.politics.guns	0.45161292
talk.religion.misc	0.41095892
rec.autos	0.40316206
rec.sport.baseball	0.46249998
rec.sport.hockey	0.7801857
comp.sys.mac.hardware	0.35120642
sci.space	0.7202797
comp.sys.ibm.pc.hardware	0.4246988
talk.politics.misc	0.36111113
comp.graphics	0.31681246
sci.electronics	0.21882354
sci.crypt	0.7282051
sci.med	0.3668639
soc.religion.christian	0.8294314
alt.atheism	0.3529412
misc.forsale	0.7021041
comp.os.ms-windows.misc	0.45048547

Time of Execution: 7m43.640s

TF-IDF Representation

rec.motorcycles	0.7044025
comp.windows.x	0.72222227
talk.politics.mideast	0.911032
talk.politics.guns	0.7702702
talk.religion.misc	0.43209872
rec.autos	0.6603175
rec.sport.baseball	0.689441
rec.sport.hockey	0.8367953
comp.sys.mac.hardware	0.60645163
sci.space	0.81325305
comp.sys.ibm.pc.hardware	0.5301205
talk.politics.misc	0.57754016
comp.graphics	0.53177255
sci.electronics	0.49468085
sci.crypt	0.8191489
sci.med	0.7120743
soc.religion.christian	0.87539935
alt.atheism	0.40000004
misc.forsale	0.804301
comp.os.ms-windows.misc	0.6346154

Sarin Madarasmi

Time of Execution: 7m21.072s

Binary-IDF Representation

rec.motorcycles	0.07692308
comp.windows.x	0.3418903
talk.politics.mideast	0.07802038
talk.politics.guns	0.043741588
talk.religion.misc	0.036126737
rec.autos	0.119506165
rec.sport.baseball	0.11789653
rec.sport.hockey	0.12834224
comp.sys.mac.hardware	0.36949682
sci.space	0.101209424
comp.sys.ibm.pc.hardware	0.49056602
talk.politics.misc	0.04426683
comp.graphics	0.91525424
sci.electronics	0.13205458
sci.crypt	0.07599845
sci.med	0.09959072
soc.religion.christian	0.093183234
alt.atheism	1.0
misc.forsale	0.38482386
comp.os.ms-windows.misc	0.564433

Time of Execution: 8m0.782s

Square root of TF with IDF Representation

rec.motorcycles	0.07692308
comp.windows.x	0.3418903
talk.politics.mideast	0.07802038
talk.politics.guns	0.043741588
talk.religion.misc	0.036126737
rec.autos	0.119506165
rec.sport.baseball	0.11789653
rec.sport.hockey	0.12834224
comp.sys.mac.hardware	0.36949682
sci.space	0.101209424
comp.sys.ibm.pc.hardware	0.49056602
talk.politics.misc	0.04426683
comp.graphics	0.91525424
sci.electronics	0.13205458
sci.crypt	0.07599845
sci.med	0.09959072
soc.religion.christian	0.093183234
alt.atheism	1.0
misc.forsale	0.38482386

comp.os.ms-windows.misc	0.564433

Time of Execution: 7m47.694s

I could not do the 5-char representation because I got the error Java Out of Memory Error.

Ridge Regression Method:
I did this by first initializing w vector to have each dimension to be a random between 0 and 1. Then I computed to get the new w vector using the formula given for ridge regression. After finished, computing the new w, I check the difference between the old w vector and the new one. If it is less than 0.001 (it converges), I break the loop. If not, continue to compute new W vector again. I do this for each different lambda and compute the F1 Score and keep record of the highest F1 score. I do this for each of the binary models.

I could not get the report for the ridge regression because it took too long to run. So I cannot get the 10 highest words and the F1 score results because it was taking too long to test them.