Classification Of Book Genres

By its Cover and Title

Team -33
Project - 10

Members:

Yash Goyal (201502181)
Dhruval Jain (201530109)
Mayank Garg(201530097)
Kumar Abhishek(201502172)

Motivation

The motivation for solving this problem is for designing covers of new books that want to come onto market with a relatively unknown author. This study would show what types of features concerning covers and titles are most important for determining a book's genre, and subsequently how a consumer perceives such a book.

Problem Addressed

In our project the problem we are addressing is to use book cover and its title to classify the book to a genre.

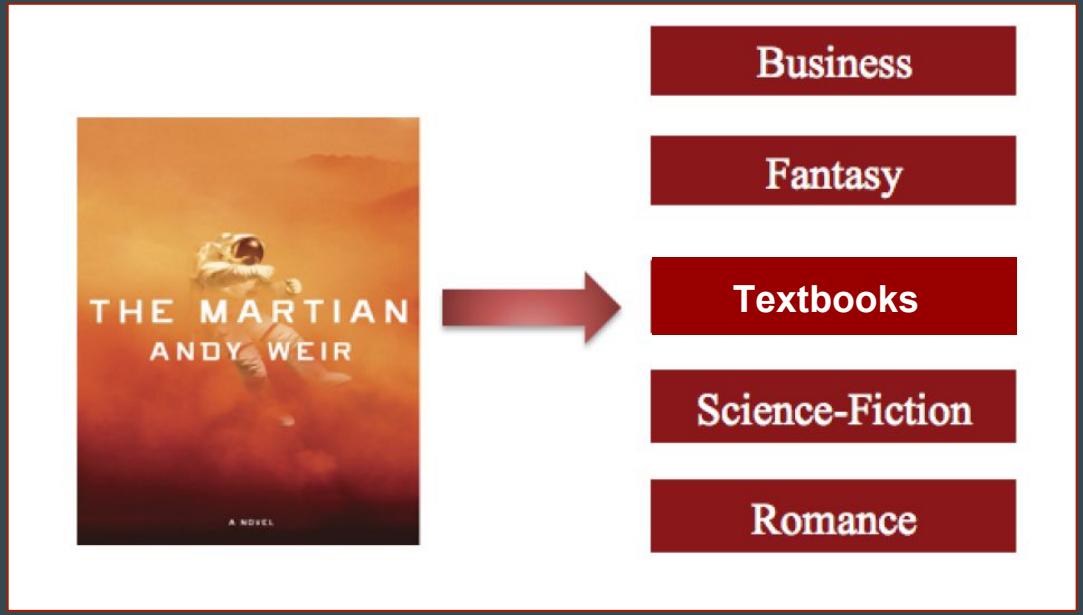
It has been shown that the cover design has a significant impact on the sales of a book, with book sales often shooting up after a change in design.

Our goal is to create a model that can determine how representative a cover is of its genre, as a method to later evaluate if the more a book cover resembles others in its genre, the higher the book sales.

Book Genres

We classify books into five genres: Business, Fantasy, Textbooks, Science-Fiction, and Romance.

Assumption : Non - overlapping.



Flow of work

1.) Dataset generation.

2.) Pre-processing

3.) Extracting Image (Book Cover) Features ~ ImageNet

4.) Extracting Features from Book Title ~ Word2vec

5.) Combining both the features to feed them into a classifier for final prediction.

Dataset Generation

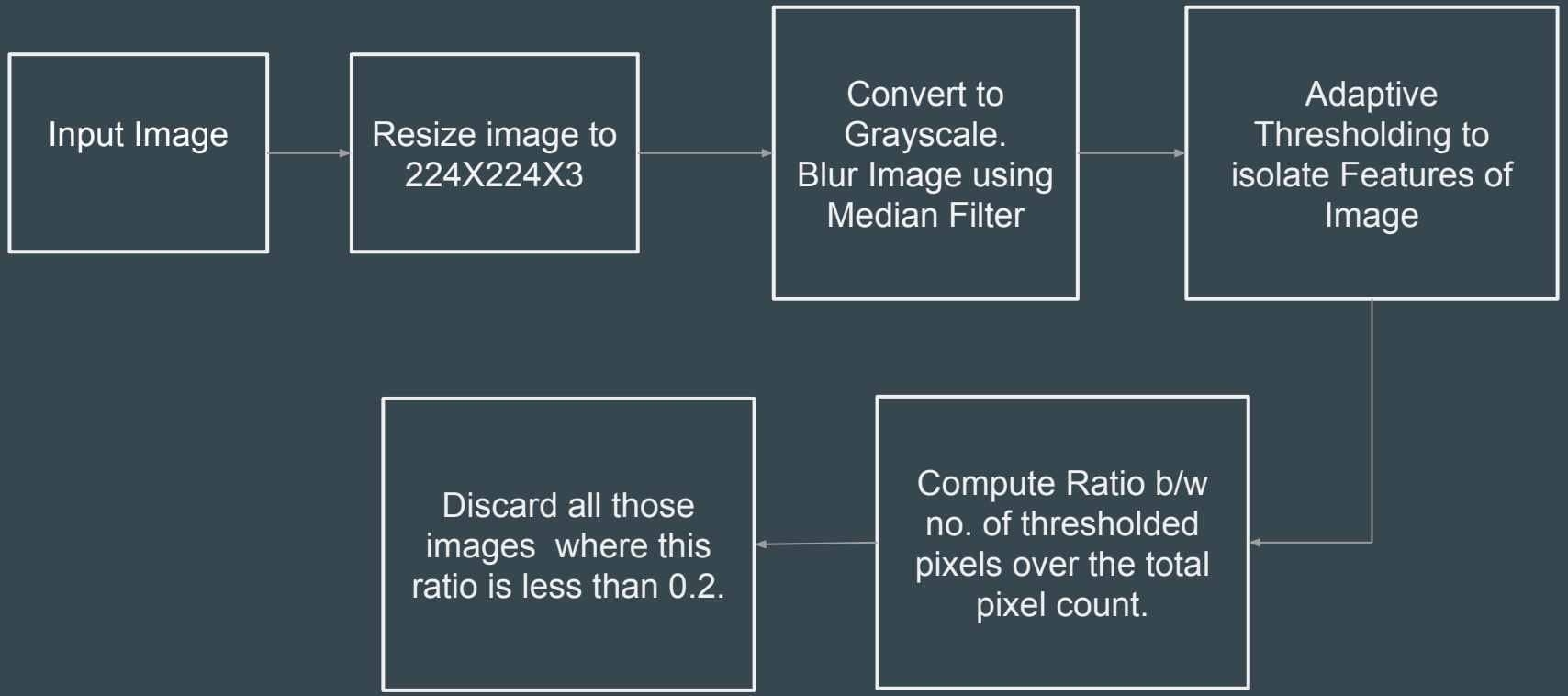
The dataset obtained from OpenLibrary.org consists of a total of 6,185 images from the five genres.

We used Selenium and BeautifulSoup library in python to scrape the book-cover images and titles to generate the dataset.

We have encoded the titles in UTF-8 to bring all the words in plain english.

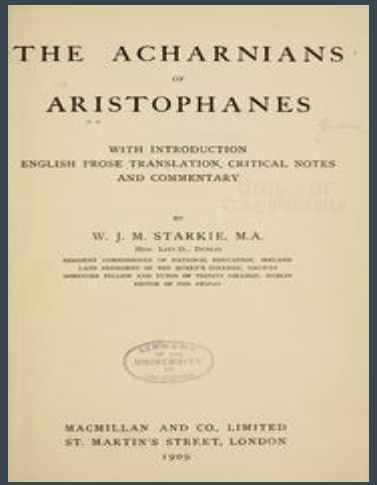
Testing: 17% of each genre data was incorporated in building the test dataset.

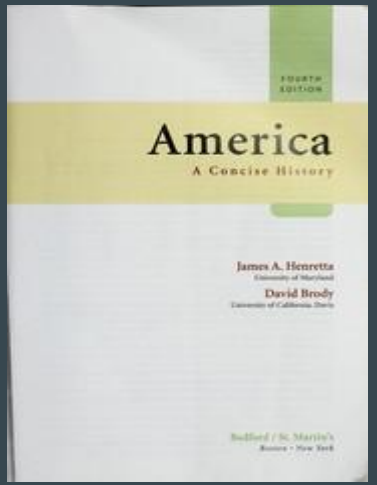
Pre-processing Images Before Training



Results from pre-processing step

‘History’ ,’Music’ and ‘Medicine’ genres were dropped out and we came up with new genre ‘textbooks’ as the book-covers were mostly plain color in texture and text on it also didn’t provide much of the information. So, most of the books in this genre were discarded in the pre-processing step where the ratio was less than 0.2.





Feature Extraction from Images

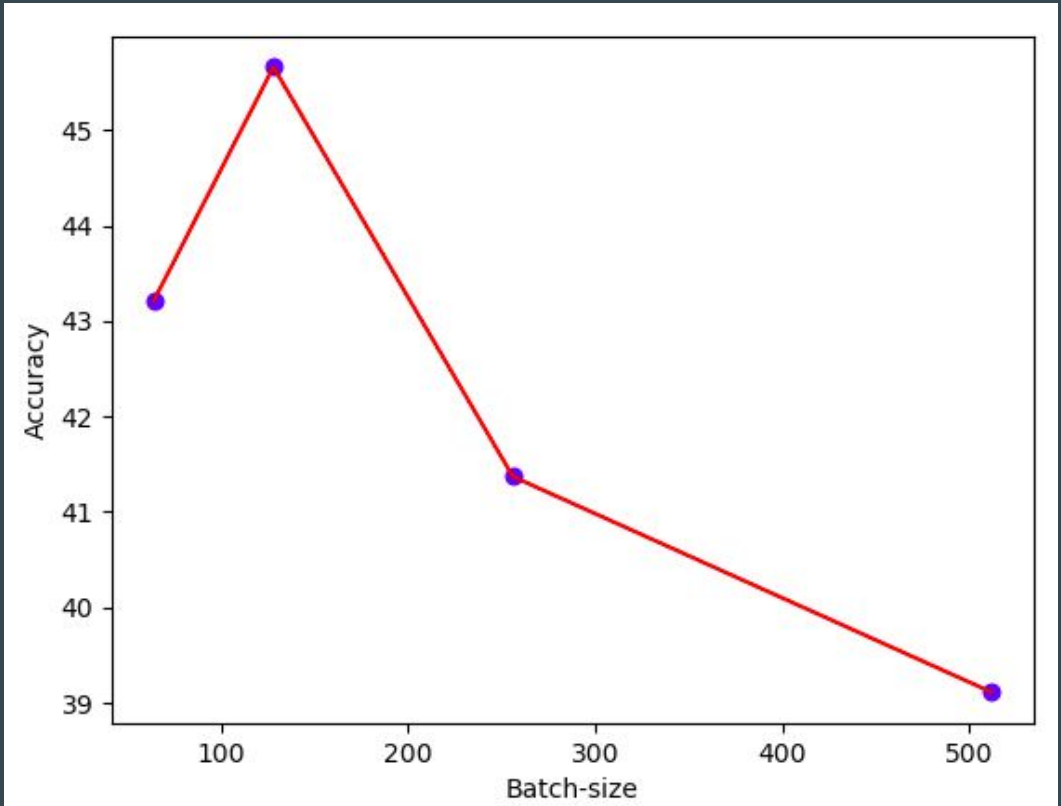
No. of image features used are 4096. Various CNN architecture were considered to extract features from the images:

1. AlexNet
2. VGGNet-16
3. VGGNet-19

We chose AlexNet because it gave the maximum accuracy on our dataset.

Architecture	Accuracy (Using only CNN for feature extraction)
AlexNet	45.66%
VGGNet-19	39.17%
VGGNet-16	36.52%

Batch-Size and Epochs



No. of epochs = 200, which takes around 13 mins, ie , 4s for each epoch on GeforceGTX 1080 Ti at 1.582Ghz.

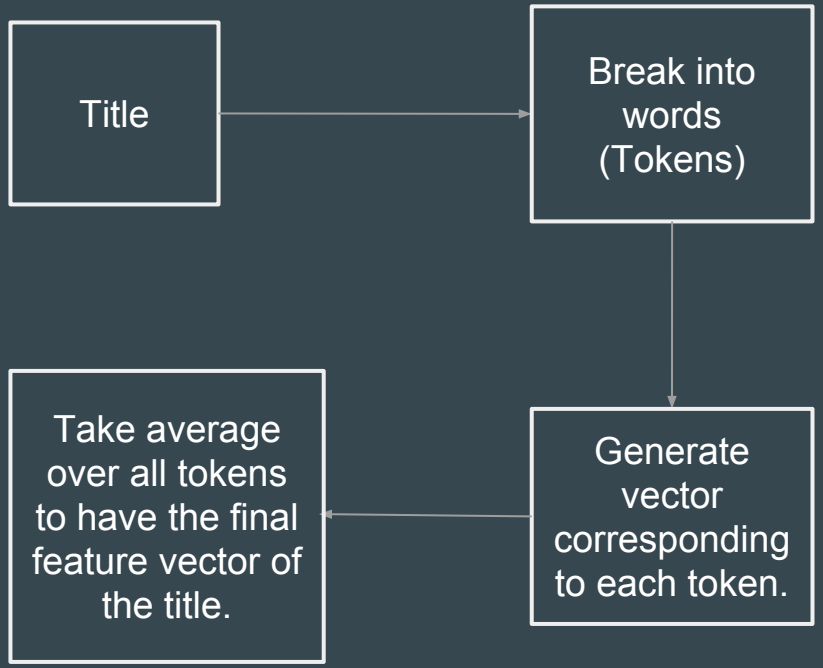
Maximum accuracy of 44.56% obtained at 128 batch-size.

Model Summary : AlexNet

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 56, 56, 3)	1092
batch_normalization_1 (Batch	(None, 56, 56, 3)	12
max_pooling2d_1 (MaxPooling2	(None, 27, 27, 3)	0
conv2d_2 (Conv2D)	(None, 27, 27, 96)	7296
batch_normalization_2 (Batch	(None, 27, 27, 96)	384
max_pooling2d_2 (MaxPooling2	(None, 13, 13, 96)	0
conv2d_3 (Conv2D)	(None, 13, 13, 256)	221440
conv2d_4 (Conv2D)	(None, 13, 13, 384)	885120
conv2d_5 (Conv2D)	(None, 13, 13, 384)	1327488
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 384)	0
flatten_1 (Flatten)	(None, 13824)	0
dense_1 (Dense)	(None, 4096)	56627200
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 5)	20485

Total params: 75,871,829
Trainable params: 75,871,631
Non-trainable params: 198

Feature Extraction from Title



Word2vec is a shallow, two-layer neural networks that is trained on GoogleNews dataset.

It maps each word in the corpus to a 300 dimensional vector in the vector space of real numbers.

Choosing Averaging over Appending

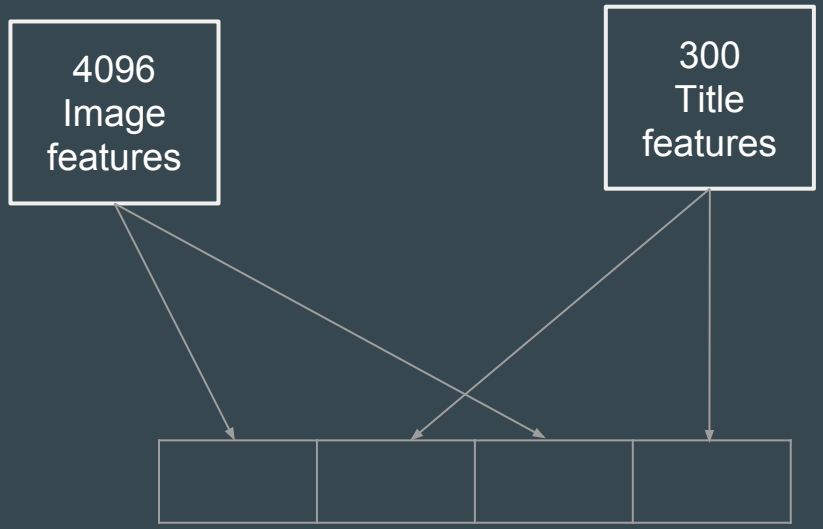
Appending vector of each token in the title will produce each features of varying length and padding gave worse results than those obtained in the case of averaging.

Hit and Miss cases

If token not found in the text corpus, then it is ignored.

If entire title not found, then the book is classified using the features obtained from AlexNet itself.

Combining the features



Features from both the attributes are appended alternatively and title features are re-used when they are exhausted.

Total features thus obtained =

4096 + floor(4096/300)*300 = 7996

This ensures uniformity and equality of both the features since title features are very less than image features.

Classification into genres...

We have used various classifiers into predict the genres like:

1. Multi-class SVM
2. AdaBoost using Random Forest
3. Softmax

Multi-class SVM(C=1)

```
Epoch 180/200
6183/6183 [==============================] - 4s - loss: 1.3541 - acc: 0.4292
Epoch 181/200
6183/6183 [==============================] - 4s - loss: 1.3538 - acc: 0.4301
Epoch 182/200
6183/6183 [==============================] - 4s - loss: 1.3558 - acc: 0.4380
Epoch 183/200
6183/6183 [==============================] - 4s - loss: 1.3513 - acc: 0.4299
Epoch 184/200
6183/6183 [==============================] - 4s - loss: 1.3522 - acc: 0.4331
Epoch 185/200
6183/6183 [==============================] - 4s - loss: 1.3515 - acc: 0.4296
Epoch 186/200
6183/6183 [==============================] - 4s - loss: 1.3486 - acc: 0.4372
Epoch 187/200
6183/6183 [==============================] - 4s - loss: 1.3516 - acc: 0.4288
Epoch 188/200
6183/6183 [==============================] - 4s - loss: 1.3469 - acc: 0.4351
Epoch 189/200
6183/6183 [==============================] - 4s - loss: 1.3497 - acc: 0.4346
Epoch 190/200
6183/6183 [==============================] - 4s - loss: 1.3457 - acc: 0.4398
Epoch 191/200
6183/6183 [==============================] - 4s - loss: 1.3494 - acc: 0.4320
Epoch 192/200
6183/6183 [==============================] - 4s - loss: 1.3456 - acc: 0.4339
Epoch 193/200
6183/6183 [==============================] - 4s - loss: 1.3470 - acc: 0.4318
Epoch 194/200
6183/6183 [==============================] - 4s - loss: 1.3498 - acc: 0.4344
Epoch 195/200
6183/6183 [==============================] - 4s - loss: 1.3465 - acc: 0.4301
Epoch 196/200
6183/6183 [==============================] - 4s - loss: 1.3439 - acc: 0.4315
Epoch 197/200
6183/6183 [==============================] - 4s - loss: 1.3457 - acc: 0.4354
Epoch 198/200
6183/6183 [==============================] - 4s - loss: 1.3421 - acc: 0.4352
Epoch 199/200
6183/6183 [==============================] - 4s - loss: 1.3429 - acc: 0.4344
Epoch 200/200
6183/6183 [==============================] - 4s - loss: 1.3437 - acc: 0.4343
(6183, 1, 300)
(1284, 1, 300)
Test accuracy: 0.53738317757
```

Accuracy obtained is 53. 73%

Parameters used :

C = 1 (slack parameter)

Kernel = ‘ rbf‘

Multi-class SVM(C = 0.0001)

```
6183/6183 [==============================] - 4s - loss: 1.3407 - acc: 0.4
Epoch 181/200
6183/6183 [==============================] - 4s - loss: 1.3387 - acc: 0.4443
Epoch 182/200
6183/6183 [==============================] - 4s - loss: 1.3378 - acc: 0.4368
Epoch 183/200
6183/6183 [==============================] - 4s - loss: 1.3387 - acc: 0.4388
Epoch 184/200
6183/6183 [==============================] - 4s - loss: 1.3408 - acc: 0.4338
Epoch 185/200
6183/6183 [==============================] - 4s - loss: 1.3377 - acc: 0.4453
Epoch 186/200
6183/6183 [==============================] - 4s - loss: 1.3412 - acc: 0.4381
Epoch 187/200
6183/6183 [==============================] - 4s - loss: 1.3380 - acc: 0.4414
Epoch 188/200
6183/6183 [==============================] - 4s - loss: 1.3372 - acc: 0.4412
Epoch 189/200
6183/6183 [==============================] - 4s - loss: 1.3387 - acc: 0.4362
Epoch 190/200
6183/6183 [==============================] - 4s - loss: 1.3369 - acc: 0.4377
Epoch 191/200
6183/6183 [==============================] - 4s - loss: 1.3405 - acc: 0.4389
Epoch 192/200
6183/6183 [==============================] - 4s - loss: 1.3369 - acc: 0.4346
Epoch 193/200
6183/6183 [==============================] - 4s - loss: 1.3357 - acc: 0.4425
Epoch 194/200
6183/6183 [==============================] - 4s - loss: 1.3359 - acc: 0.4398
Epoch 195/200
6183/6183 [==============================] - 4s - loss: 1.3389 - acc: 0.4372
Epoch 196/200
6183/6183 [==============================] - 4s - loss: 1.3354 - acc: 0.4432
Epoch 197/200
6183/6183 [==============================] - 4s - loss: 1.3369 - acc: 0.4398
Epoch 198/200
6183/6183 [==============================] - 4s - loss: 1.3368 - acc: 0.4407
Epoch 199/200
6183/6183 [==============================] - 4s - loss: 1.3357 - acc: 0.4377
Epoch 200/200
6183/6183 [==============================] - 4s - loss: 1.3348 - acc: 0.4423
(6183, 1, 300)
(1284, 1, 300)
Train accuracy: 0.250687368591
Test accuracy: 0.241433021807
```

Accuracy obtained is 24. 73%

Parameters used :

C = 0.0001 (slack parameter)

Kernel = ‘ rbf‘.

The above isn’t convincing.

Multi-class SVM (C = 10)

```
6183/6183 [==============================] - 4s - loss: 1.3143 - acc: 0.4541
Epoch 181/200
6183/6183 [==============================] - 4s - loss: 1.3187 - acc: 0.4503
Epoch 182/200
6183/6183 [==============================] - 4s - loss: 1.3195 - acc: 0.4498
Epoch 183/200
6183/6183 [==============================] - 4s - loss: 1.3211 - acc: 0.4462
Epoch 184/200
6183/6183 [==============================] - 4s - loss: 1.3208 - acc: 0.4548
Epoch 185/200
6183/6183 [==============================] - 4s - loss: 1.3180 - acc: 0.4546
Epoch 186/200
6183/6183 [==============================] - 4s - loss: 1.3180 - acc: 0.4498
Epoch 187/200
6183/6183 [==============================] - 4s - loss: 1.3200 - acc: 0.4503
Epoch 188/200
6183/6183 [==============================] - 4s - loss: 1.3153 - acc: 0.4530
Epoch 189/200
6183/6183 [==============================] - 4s - loss: 1.3141 - acc: 0.4527
Epoch 190/200
6183/6183 [==============================] - 4s - loss: 1.3092 - acc: 0.4571
Epoch 191/200
6183/6183 [==============================] - 4s - loss: 1.3138 - acc: 0.4504
Epoch 192/200
6183/6183 [==============================] - 4s - loss: 1.3190 - acc: 0.4461
Epoch 193/200
6183/6183 [==============================] - 4s - loss: 1.3143 - acc: 0.4453
Epoch 194/200
6183/6183 [==============================] - 4s - loss: 1.3200 - acc: 0.4514
Epoch 195/200
6183/6183 [==============================] - 4s - loss: 1.3163 - acc: 0.4514
Epoch 196/200
6183/6183 [==============================] - 4s - loss: 1.3126 - acc: 0.4524
Epoch 197/200
6183/6183 [==============================] - 4s - loss: 1.3125 - acc: 0.4520
Epoch 198/200
6183/6183 [==============================] - 4s - loss: 1.3162 - acc: 0.4514
Epoch 199/200
6183/6183 [==============================] - 4s - loss: 1.3120 - acc: 0.4585
Epoch 200/200
6183/6183 [==============================] - 4s - loss: 1.3177 - acc: 0.4469
(6183, 1, 300)
(1284, 1, 300)
Train accuracy: 0.727478570273
Test accuracy: 0.678348909657
```

Test Accuracy obtained is 67.83%

Parameters used :

C = 10 (slack parameter)

Kernel = ‘ rbf‘

Adaboost : Random Forest

```
Epoch 180/200
6183/6183 [==============================] - 4s - loss: 1.3473 - acc: 0.4318
Epoch 181/200
6183/6183 [==============================] - 4s - loss: 1.3408 - acc: 0.4262
Epoch 182/200
6183/6183 [==============================] - 4s - loss: 1.3446 - acc: 0.4352
Epoch 183/200
6183/6183 [==============================] - 4s - loss: 1.3367 - acc: 0.4377
Epoch 184/200
6183/6183 [==============================] - 4s - loss: 1.3416 - acc: 0.4330
Epoch 185/200
6183/6183 [==============================] - 4s - loss: 1.3412 - acc: 0.4351
Epoch 186/200
6183/6183 [==============================] - 4s - loss: 1.3370 - acc: 0.4385
Epoch 187/200
6183/6183 [==============================] - 4s - loss: 1.3373 - acc: 0.4357
Epoch 188/200
6183/6183 [==============================] - 4s - loss: 1.3397 - acc: 0.4323
Epoch 189/200
6183/6183 [==============================] - 4s - loss: 1.3350 - acc: 0.4398
Epoch 190/200
6183/6183 [==============================] - 4s - loss: 1.3339 - acc: 0.4330
Epoch 191/200
6183/6183 [==============================] - 4s - loss: 1.3364 - acc: 0.4364
Epoch 192/200
6183/6183 [==============================] - 4s - loss: 1.3325 - acc: 0.4302
Epoch 193/200
6183/6183 [==============================] - 4s - loss: 1.3362 - acc: 0.4389
Epoch 194/200
6183/6183 [==============================] - 4s - loss: 1.3328 - acc: 0.4334
Epoch 195/200
6183/6183 [==============================] - 4s - loss: 1.3350 - acc: 0.4393
Epoch 196/200
6183/6183 [==============================] - 4s - loss: 1.3340 - acc: 0.4383
Epoch 197/200
6183/6183 [==============================] - 4s - loss: 1.3337 - acc: 0.4367
Epoch 198/200
6183/6183 [==============================] - 4s - loss: 1.3294 - acc: 0.4396
Epoch 199/200
6183/6183 [==============================] - 4s - loss: 1.3304 - acc: 0.4344
Epoch 200/200
6183/6183 [==============================] - 4s - loss: 1.3296 - acc: 0.4393
(6183, 1, 300)
(1284, 1, 300)
Test accuracy: 0.519470404984
```

Accuracy obtained is 51. 94%

Classifier: Decision-trees

Max-Depth = 100 levels

No . of estimators = 10

Softmax : Highest Accuracy

```
Epoch 180/200
6183/6183 [==============================] - 4s - loss: 1.3536 - acc: 0.4333
Epoch 181/200
6183/6183 [==============================] - 4s - loss: 1.3525 - acc: 0.4398
Epoch 182/200
6183/6183 [==============================] - 4s - loss: 1.3562 - acc: 0.4341
Epoch 183/200
6183/6183 [==============================] - 4s - loss: 1.3524 - acc: 0.4275
Epoch 184/200
6183/6183 [==============================] - 4s - loss: 1.3529 - acc: 0.4318
Epoch 185/200
6183/6183 [==============================] - 4s - loss: 1.3521 - acc: 0.4391
Epoch 186/200
6183/6183 [==============================] - 4s - loss: 1.3508 - acc: 0.4347
Epoch 187/200
6183/6183 [==============================] - 4s - loss: 1.3543 - acc: 0.4281
Epoch 188/200
6183/6183 [==============================] - 4s - loss: 1.3515 - acc: 0.4322
Epoch 189/200
6183/6183 [==============================] - 4s - loss: 1.3540 - acc: 0.4291
Epoch 190/200
6183/6183 [==============================] - 4s - loss: 1.3503 - acc: 0.4343
Epoch 191/200
6183/6183 [==============================] - 4s - loss: 1.3514 - acc: 0.4368
Epoch 192/200
6183/6183 [==============================] - 4s - loss: 1.3502 - acc: 0.4344
Epoch 193/200
6183/6183 [==============================] - 4s - loss: 1.3509 - acc: 0.4406
Epoch 194/200
6183/6183 [==============================] - 4s - loss: 1.3509 - acc: 0.4391
Epoch 195/200
6183/6183 [==============================] - 4s - loss: 1.3463 - acc: 0.4328
Epoch 196/200
6183/6183 [==============================] - 4s - loss: 1.3471 - acc: 0.4370
Epoch 197/200
6183/6183 [==============================] - 4s - loss: 1.3498 - acc: 0.4325
Epoch 198/200
6183/6183 [==============================] - 4s - loss: 1.3489 - acc: 0.4372
Epoch 199/200
6183/6183 [==============================] - 4s - loss: 1.3489 - acc: 0.4406
Epoch 200/200
6183/6183 [==============================] - 4s - loss: 1.3479 - acc: 0.4377
(6183, 1, 300)
(1284, 1, 300)
Test accuracy: 0.703271028037
```

Test Accuracy obtained is 70.32%

penalty = ‘l2’

C = 1.0

Softmax : Confusion Matrix

```
Train accuracy: 0.839721817888
Test accuracy: 0.710280373832
Confusion matrix for Testing
[[204   2  14  12  21]
 [  4 158   4  31  72]
 [ 16   2 149   2   7]
 [  9  32   1 248  20]
 [ 17  76   5  25 153]]
Confusion matrix for Training
[[1126   18   28   21   41]
 [  20  923   10  116  169]
 [  39   12  813   13   35]
 [  10   88    6 1391   55]
 [  34  180   22   74  939]]
```

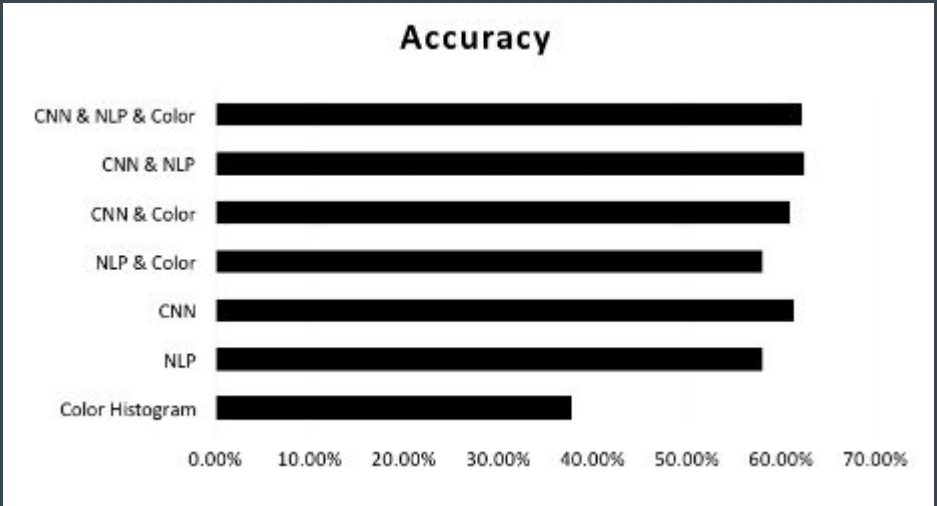
Precision and Recall in Testing for all classes

	Precision	Recall	F1 score
Business	0.806	0.816	0.81
Fantasy	0.587	0.581	0.583
Textbooks	0.846	0.861	0.85
Romance	0.8	0.779	0.78
Science-fiction	0.55	0.560	0.55

Final Accuracy = 0.718

Extension from paper

The paper registers a maximum accuracy of around 63% but while experimenting with different classifiers we were able to achieve an accuracy of 70.32%.



Accuracies by different methods given in paper