Classifying Movie Reviews: Crowd-powered Machine Learning

By: Harshith Shankar T R (19230323) and Shyam Kumar S (19230735)

1.0 Introduction

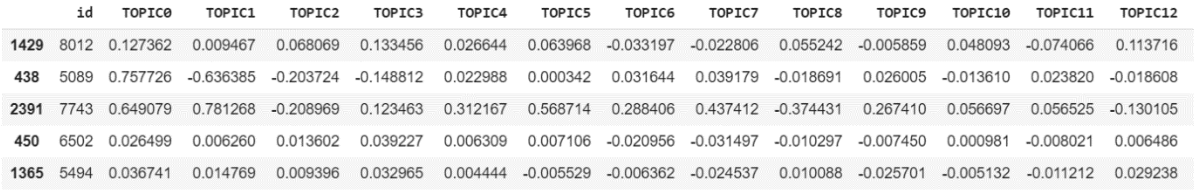
The aim of this assignment is to understand the importance of crowd sourcing in classifying movie reviews. Three different models are built for classifying the movie reviews as either positive or negative.

2.0 Model using gold dataset

A random sample of 1000 reviews are taken from the gold-standard dataset. This dataset is used as training data for building a classifier(model) based on the Decision Tree algorithm. The model thus created is tested using the test dataset and different scores for this like F-score, accuracy and prediction probabilities.

1000 reviews were sampled randomly from the gold-standard dataset using a random seed value. The python function sample is used for the random sampling. The sampling is done without repetition as this would bias the classification model with the review which has repeated many number of times.

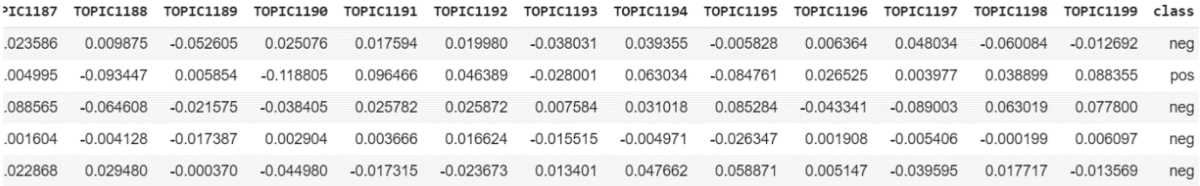
A view of the sample is as shown in Figure 1.1. The topic goes on till Topic 1199 and then the label(class). The gold dataset also looks similar.



	id	TOPIC0	TOPIC1	TOPIC2	TOPIC3	TOPIC4	TOPIC5	TOPIC6	TOPIC7	TOPIC8	TOPIC9	TOPIC10	TOPIC11	TOPIC12
1429	8012	0.127362	0.009467	0.068069	0.133456	0.026644	0.063968	-0.033197	-0.022806	0.055242	-0.005859	0.048093	-0.074066	0.113716
438	5089	0.757726	-0.636385	-0.203724	-0.148812	0.022988	0.000342	0.031644	0.039179	-0.018691	0.026005	-0.013610	0.023820	-0.018608
2391	7743	0.649079	0.781268	-0.208969	0.123463	0.312167	0.568714	0.288406	0.437412	-0.374431	0.267410	0.056697	0.056525	-0.130105
450	6502	0.026499	0.006260	0.013602	0.039227	0.006309	0.007106	-0.020956	-0.031497	-0.010297	-0.007450	0.000981	-0.008021	0.006486
1365	5494	0.036741	0.014769	0.009396	0.032965	0.004444	-0.005529	-0.006362	-0.024537	0.010088	-0.025701	-0.005132	-0.011212	0.029238

Figure 1.1 A view of the data sample from the gold standard dataset

The topic goes on till Topic 1199 and then the label(class) is present in the last column as can be seen in Figure 1.2

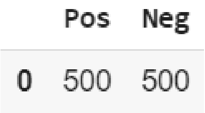


›IC1187	TOPIC1188	TOPIC1189	TOPIC1190	TOPIC1191	TOPIC1192	TOPIC1193	TOPIC1194	TOPIC1195	TOPIC1196	TOPIC1197	TOPIC1198	TOPIC1199	class
.023586	0.009875	-0.052605	0.025076	0.017594	0.019980	-0.038031	0.039355	-0.005828	0.006364	0.048034	-0.060084	-0.012692	neg
.004995	-0.093447	0.005854	-0.118805	0.096466	0.046389	-0.028001	0.063034	-0.084761	0.026525	0.003977	0.038899	0.088355	pos
.088565	-0.064608	-0.021575	-0.038405	0.025782	0.025872	0.007584	0.031018	0.085284	-0.043341	-0.089003	0.063019	0.077800	neg
.001604	-0.004128	-0.017387	0.002904	0.003666	0.016624	-0.015515	-0.004971	-0.026347	0.001908	-0.005406	-0.000199	0.006097	neg
.022868	0.029480	-0.000370	-0.044980	-0.017315	-0.023673	0.013401	0.047662	0.058871	0.005147	-0.039595	0.017717	-0.013569	neg

Figure 1.2 Samples data showing the number of features and the class columns

Sampling of data randomly is helps in getting the best estimation of the features as we are not sure of any ordering that the data follow. So, it will be useful in predicting the new data which also is random.

The sampled data obtained has 1000 sentences, their corresponding features and labels. The distribution of the labels is as shown in Figure 1.3



	Pos	Neg
0	500	500

Figure 1.3 Distribution of labels in the sampled dataset

From Figure 1.3 we can see that the labels are equally distributed. There are enough number of reviews with both type of labels which suggest that the data sampled from the gold dataset is proper for training the classification model.

2.1 Training the Decision Tree Classifier

The Model is trained using the Decision Tree algorithm. Python Machine Learning package “Sklearn” is used for this purpose.

The columns TOPIC0 to TOPIC1199 form the features for the sampled data. They are stored in a dataframe called features.

The class column forms the corresponding labels.

DecisionTreeClassifier is imported from sklearn.tree and an object of the DecisionTreeClassifier is created.

Using this object, the fit function is called which trains the model

The fit function takes the features and labels as input and trains the model.

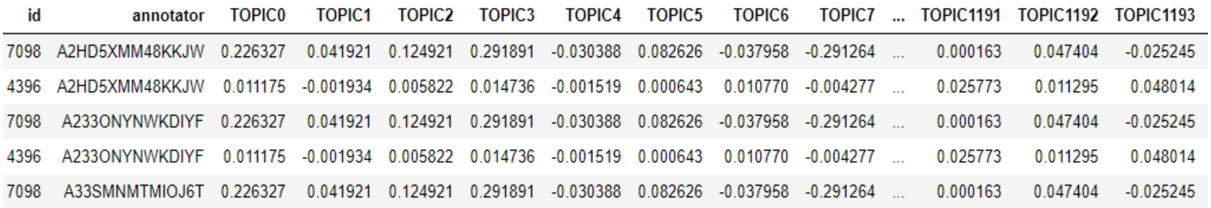
Then the model is tested using the predict function on the test data from the same object which returns the predicted labels as a list

The predicted labels are compared with the test labels to obtain the accuracy of the model

3.0 Model using majority vote

The sentences in the sampled data taken from the gold standard dataset are used to get the training dataset from the mturk dataset which is the crowdsourced dataset, where each of the reviews are crowdsourced, for training this model. The sentence id of the sampled dataset is taken and the reviews corresponding to these ids are taken from the mturk dataset to obtain an mturk sample dataset.

Sample_mturk dataset is created by extracting rows which has common ‘id’ between gold_sample and mturk dataset. Sample_mturk has 5548 number of rows.

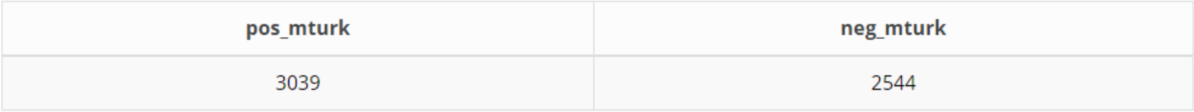


id	annotator	TOPIC0	TOPIC1	TOPIC2	TOPIC3	TOPIC4	TOPIC5	TOPIC6	TOPIC7	...	TOPIC1191	TOPIC1192	TOPIC1193
7098	A2HD5XMM48KKJW	0.226327	0.041921	0.124921	0.291891	-0.030388	0.082626	-0.037958	-0.291264	...	0.000163	0.047404	-0.025245
4396	A2HD5XMM48KKJW	0.011175	-0.001934	0.005822	0.014736	-0.001519	0.000643	0.010770	-0.004277	...	0.025773	0.011295	0.048014
7098	A233ONYNWKDIYF	0.226327	0.041921	0.124921	0.291891	-0.030388	0.082626	-0.037958	-0.291264	...	0.000163	0.047404	-0.025245
4396	A233ONYNWKDIYF	0.011175	-0.001934	0.005822	0.014736	-0.001519	0.000643	0.010770	-0.004277	...	0.025773	0.011295	0.048014
7098	A33SMNMTMIOJ6T	0.226327	0.041921	0.124921	0.291891	-0.030388	0.082626	-0.037958	-0.291264	...	0.000163	0.047404	-0.025245

Figure 2.1: The same review annotated by different annotators.

The dataset consists of different sentences which are labelled by different annotators. So, the same review will be annotated by different annotators as shown in Figure 2.1. Only a single instance of each review is required. So, the labels are assigned to the different sentences based on the majority vote. If the vote is *tie*, then value *negative* is considered as majority vote. It is done so as the distribution of labels is more skewed towards the positive label. Majority voting is calculated by first grouping the sentences with same ‘id’ and get the maximum count of the unique values on each grouped sentence.

The distribution of labels is shown in Figure 2.2

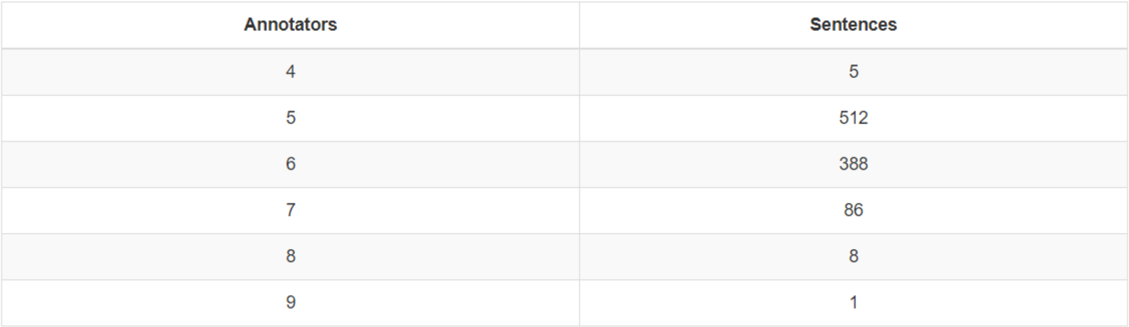


pos_mturk	neg_mturk
3039	2544

Figure 2.2 Distribution of labels in the crowdsourced sampled data

From Figure 2.2 we can see that the labels are not equally distributed. But there are enough number of reviews with both type of labels which suggest that the data sampled from the crowdsourced dataset is good for training the classification model.

The distribution of annotators for different sentences is as shown in Figure 2.3



Annotators	Sentences
4	5
5	512
6	388
7	86
8	8
9	1

Figure 2.3 Number of Annotators and corresponding Number of Sentences for the sampled crowdsourced data

The average number of annotators maybe between 5 and 6 as we can observe from the Figure 2.3 which is good as we can get different perspectives which helps is the classifying the label properly. Minimum of 4 annotators and a maximum of 9 annotators classify each of the sampled reviews. Half of the training data is annotated by 5 annotators followed by 6 and 7 annotators in decreasing order of the sentences.

3.1 Training the Decision Tree classifier:

The Model is trained using the Decision Tree algorithm. Python Machine Learning package “Sklearn” is used for this purpose.

Sample_mturk has repeated sentences which is redundant to train the model. Hence redundant rows are removed and stored in data frame called “*mt_sample*”.

The columns TOPIC0 to TOPIC1199 from *mt_sample* data frame is used as feature to train the model.

Label is calculated by using majority voting by grouping sentence id. Majority vote is stored in variable “*mt_class*”. “*mt_class*” is used as label to train the decision tree model. Below code is used to get majority vote of each sentence.

mt_class = mt_sample.groupby(['id'])['class'].agg(lambda x:x.value_counts().index[0])

DecisionTreeClassifier is imported from sklearn.tree and an object of the dtc is created.

Using this object, the fit function is called which trains the model

The fit function takes the features and labels as input and trains the model.

Then the model is tested using the predict function on the test data from the same object which returns the predicted labels as a list

The predicted labels are compared with the test labels to obtain the accuracy of the model

3.0 Model using Dawid & Skene

Dataset used here is from *Sample_mturk* and uses dawid & skene method.

3.1 Training the Decision Tree classifier:

The model is trained using the Decision Tree algorithm. Python Machine Learning package "Sklearn" is used for this purpose.

Sample_mturk dataset is created by extracting rows which has common 'id' between gold_sample and mturk dataset.

Sample_mturk has repeated sentences which is redundant to train the model. Hence redundant rows are removed and stored in a data frame called "*mt_sample*".

The columns TOPIC0 to TOPIC1199 from *mt_sample* data frame is used as a ***feature*** to train the model.

Labels are calculated by aggregating the labels provided by workers using David and Skene method that uses a maximum likelihood estimation approach.

David and skene method is as follows,

This method requires three data structure

a) Input data frame:

Create input data frame using column 'id', 'annotator', and 'class' from the sample_mturk data frame. Input data frame has sentence id as rows and annotator as columns. Values of input data frame is from the column 'class'. Input data frame is sparse matrix having class values at the corresponding sentence id and annotator.

b) Estimated polarity data frame:
This data frame is initialised with majority voting. Previously used method is also used for majority voting.

c) Reliability matrix of each worker node:
Each worker node is a confusion matrix in the list. Each index of the list represents each worker node. Confusion matrix of each worker node is initialised with a true positive and false negative which indicates the annotators has 100% accuracy.

Process:

a) Weight workers votes according to workers reliability.
For each worker, compare worker value with the value (i.e. maximum value) in the estimated polarity data frame.
If the value given by the worker is positive for the particular sentence and it matches with estimated polarity then, add value from positive column of the estimated polarity data frame to the true positive in the confusion matrix of that worker. if the values don't match then value is added to false positive in the confusion matrix.
If the value given by the worker is negative for the particular sentence and it matches with estimated polarity then, add value from negative column of the estimated polarity data frame to the true negative in the confusion matrix of this worker. if the values don't match then the value is added to false negative in the confusion matrix.
Normalise the values on actual values in the confusion matrix of each worker.

b) Update estimated polarity data frame with respect to worker nodes.

For each sentence check value given by worker (annotator).
If the value given by the worker is positive then take predicted positive values from confusion matrix and add true positive to the positive column of the estimated polarity data frame and false positive to the negative column of the estimated polarity data frame.
If the value given by the worker is negative then take predicted negative values from confusion matrix and add true negative to the negative column of the estimated polarity data frame and false negative to the positive column of the estimated polarity data frame.
Normalise the values for each sentence.

Repeat this process until convergence or maximum iteration reached.

Labels created by Dawid and skene is used to train decision tree classifier.

DecisionTreeClassifier is imported from sklearn.tree and an object of the dtc_ds is created.

Using this object, the fit function is called which trains the model

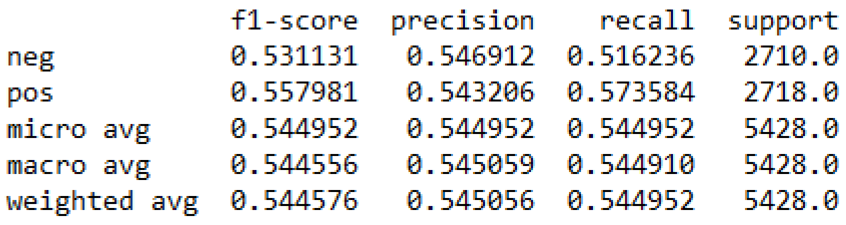
The fit function takes the features and labels as input and trains the model.

Then the model is tested using the predict function on the test data from the same object which returns the predicted labels as a list

The predicted labels are compared with the test labels to obtain the accuracy of the model.

4.0 Results comparison of models:

The classification report for the model with gold sample is as shown in Figure 4.1



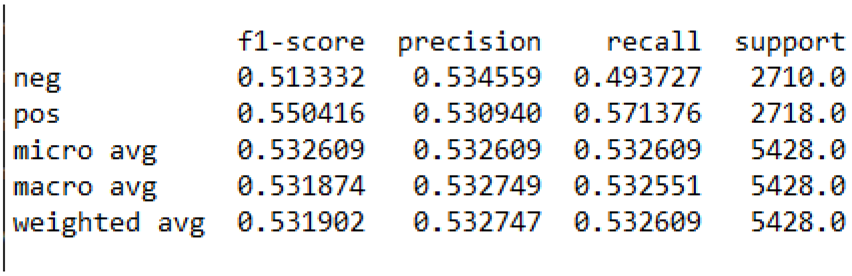
```
              f1-score  precision    recall  support
neg           0.531131   0.546912  0.516236   2710.0
pos           0.557981   0.543206  0.573584   2718.0
micro avg     0.544952   0.544952  0.544952   5428.0
macro avg     0.544556   0.545059  0.544910   5428.0
weighted avg  0.544576   0.545056  0.544952   5428.0
```

Figure 4.1: Classification report of gold sample

We can see that the model has an accuracy of 54.4%.

In addition, we also get the prediction probability for the classifier and store it in a file train_gold.txt

Classification report for model with majority vote is as shown in the Figure 4.2



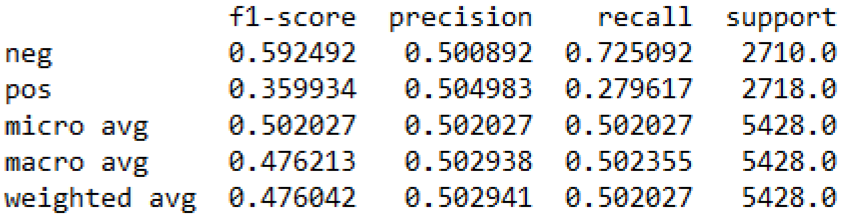
```
              f1-score  precision    recall  support
neg           0.513332   0.534559  0.493727   2710.0
pos           0.550416   0.530940  0.571376   2718.0
micro avg     0.532609   0.532609  0.532609   5428.0
macro avg     0.531874   0.532749  0.532551   5428.0
weighted avg  0.531902   0.532747  0.532609   5428.0
```

Figure 4.2: Classification report for Majority vote.

We can see that the model has an accuracy of 53.2%.

In addition, we also get the prediction probability for the classifier and store it in a file train_mv.txt

Classification report for model with Dawid & Skene is as shown in the Figure 4.3



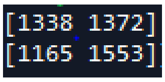
```
              f1-score  precision    recall  support
neg           0.592492   0.500892  0.725092   2710.0
pos           0.359934   0.504983  0.279617   2718.0
micro avg     0.502027   0.502027  0.502027   5428.0
macro avg     0.476213   0.502938  0.502355   5428.0
weighted avg  0.476042   0.502941  0.502027   5428.0
```

Figure 4.3: Classification report for Dawid & Skene method

We can see that the model has an accuracy of 50.2%.

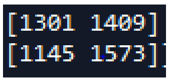
In addition, we also get the prediction probability for the classifier and store it in a file train_ds.txt

4.1 Confusion matrix:



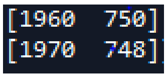
```
[1338 1372]
[1165 1553]
```

Figure 4.4: Confusion matrix of Gold sample



```
[1301 1409]
[1145 1573]
```

4.6: confusion matrix of Majority vote



```
[1960  750]
[1970  748]
```

4.5: confusion matrix of Dawid & skene

From 4.4, we can see that the confusion matrix of the gold sample which has true positive and true negative around 1338 and 1553 respectively. False positive and false negative has 1165 and 1372 respectively.

From 4.5, we can see that the confusion matrix of Dawid & skene which has true positive and true negative around 1960 and 748 respectively. False positive and false negative has 1970 and 750 respectively.

From 4.6, we can see that the confusion matrix of majority vote which has true positive and true negative around 1301 and 1573 respectively. False positive and false negative has 1145 and 1409 respectively.

4.2 ROC Curve:

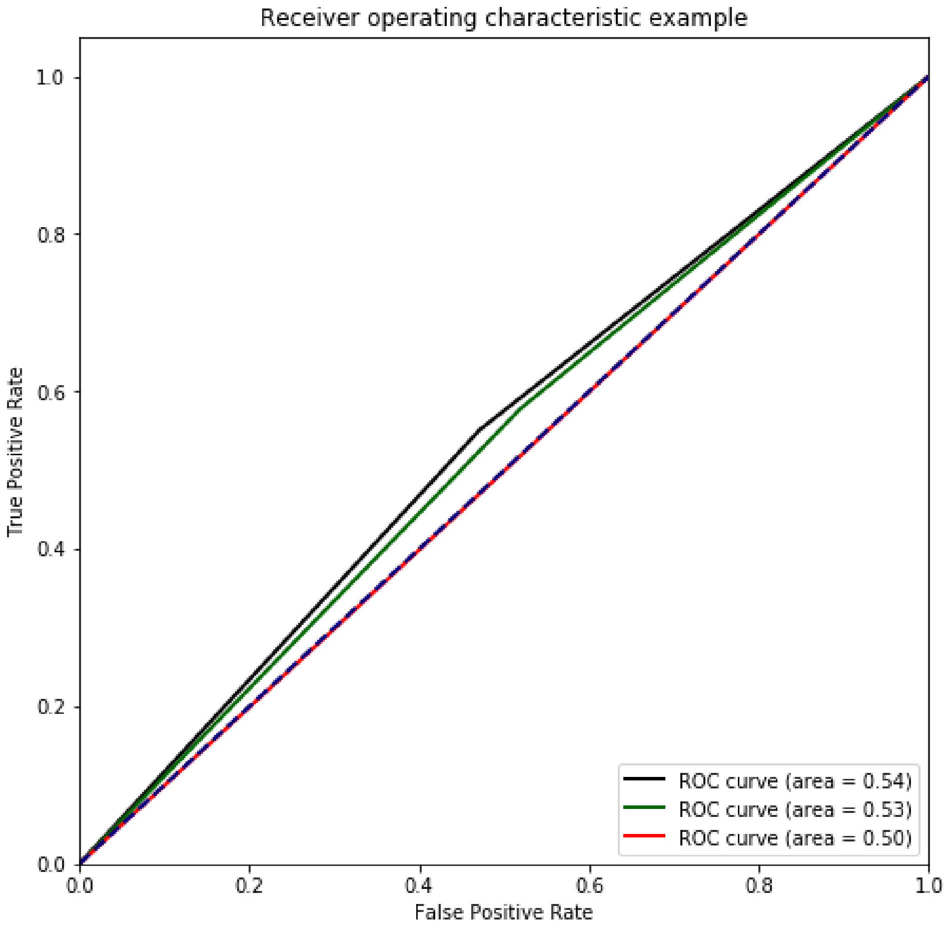


Figure 4.7: ROC curve of all models

Figure 4.7 represents ROC curve of models using gold sample dataset, majority vote, and Dawid & skene. Black colour ROC curve is for gold sample dataset. Green colour ROC curve is for majority vote. Red colour ROC curve is for Dawid & skene method. Area under ROC curve of gold sample and majority vote is same in value. Area under ROC curve of Dawid & skene method is lesser than other models.

5.0 Conclusion:

The gold dataset is the benchmark dataset. So, the model built by it is expected to give a good accuracy. We are observing an accuracy of around 54%. The low accuracy for all the datasets is due to fact that the number of training samples is 1000 which is very low as the number of testing samples is more than 5000 which is pretty high.

The aim of this assignment is to look at how crowd sourcing helps in building a good model for classification. We build Majority Voting and Dawid Skene models to aggregate the movie reviews and test the accuracy of these models. Majority voting involves taking the majority vote of the multiple annotators to give the aggregated labels. Dawid Skene algorithm also considers the votes of all the annotators. In addition, it also considers the worker(annotators) accuracies. It is like the votes of the high accuracy workers are given more weight as they are more likely to make an accurate review. This suggests that the model created using Dawid and Skene method generally should give a higher accuracy than that created using Majority Vote. But from the results as seen from Section 4 we can see that Majority vote model gives a better accuracy than that of the Dawid and Skene method. This may be due to the underlying assumption that higher accuracy workers give proper predictions not being true for this particular dataset. The small size of the training dataset maybe also be one of the reasons for this observation. As we have observed the accuracies of all the 3 models are almost equal to each other which suggests they are almost equally good. This suggests that crowdsourcing is a good way of labelling needed for the process of classification. In this assignment we have seen it for movie reviews but it can be done for other applications as well.