Shashank Agarwal
Agarwal.202@osu.edu

Anurag Kalra
Kalra.25@osu.edu

Report 6

Objective

★ Generate Association rules for Reuters data set. Build a classifier model using the Association rules.
★ Predict the labels/topics for Document test set.
★ Perform clustering on data set. On each cluster, again do the Classification.
★ Compare the classification done in both algorithms on efficiency and accuracy.

Input data

The input csv files were pre-processed and then converted into transactional model. In our input feature vector, each row represents data words for a document. Topics of a document are also part of the data set. Documents which did not have a topic were removed. In Algorithm 1, the feature vector was split in 80:20 ratio for training and test data. In Algorithm 2, each set of documents belonging in k clusters were divided in 80:20 ratio.

Generating Association Rules

We used the Apriori algorithm implementation by Christian Borgelt[1]. It takes the feature vector and list of all labels as input and generates the association rules. We can set the minimum threshold support and confidence values and minimum number of antecedents that should be in a rule. By manipulating the input given to the software, we can ensure that only the association rules of correct format are generated (W1..Wn → C). Rules of format like W1 → W2 are not generated.
We varied the support threshold from 2-10 and confidence threshold from 10-80 for the 2 algorithms. Beyond these limits we didn't get acceptable rules. For example, say for support of 20, we would get very few rules and support of <1 would generate far too many redundant rules.
The rules generated by the software has a single item in the consequent. Later we explain how we handle multiple Topic prediction for documents.
The generated rules are arranged to form the classification model. The rules are arranged in descending order of Confidence; for matching confidence values, arrange in descending order of support value; and for matching value of support, the order in which rules were generated is retained.

Algorithm 1

The feature vector is split in 80:20 ratio. 80% split is used as training data and the rest as test data. We use the training feature vector to generate association rules. The generated rules are ordered correctly to generate our classification model. The model is then used to predict labels of Test data.

Generating rules

We vary the threshold values for support from 2-6 and confidence from 10-80. These values were chosen by trial & error. We generated rules which had minimum of 2 terms in the antecedent.

Classification

The testing feature vector is run through the classification model and the labels/topics are predicted for each document. For documents which have multiple topics, we follow these steps: We find out the actual number of labels that the test document has. Then we predict the same number of labels for the document while ensuring that the different labels are generated each time.

Accuracy

We measure accuracy by comparing the predicted label with the actual label of the test document. For documents which have multiple labels/topics, we find out the ratio of correct labels. For example if a document has 5 actual labels. We'll predict 5 labels for that document. If 4 of the labels are correct, then we will store this information as: Add 5 (actual labels) to the counter containing total labels, and add 4 to the counter containing number of correctly predicted labels.

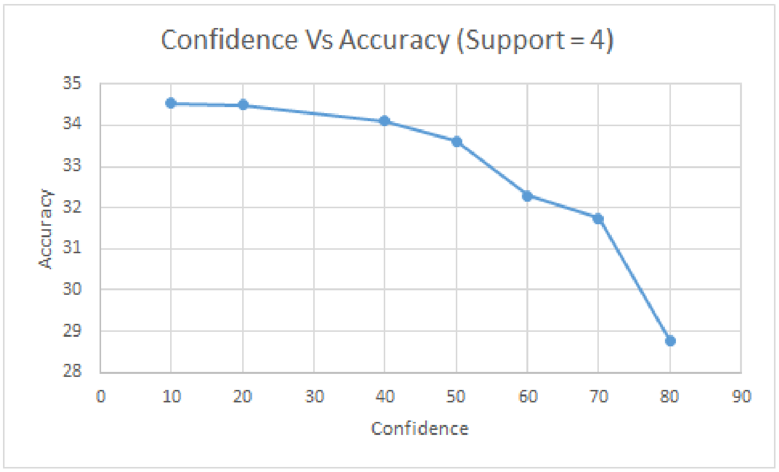
For Algorithm 1, we get:

Support	Confidence	No. of rules generated	Correct classification	Efficiency	Time to train (sec)	Time for Classification (sec)
2	10	7502	4830	33.70561282	16.23	12.17
2	20	6943	4807	33.66426624	8.4	12.22
2	40	6125	4762	33.29371461	2.31	11.71
2	50	5811	4695	32.82528141	1.47	11.45
2	60	3350	4523	31.62494756	0.95	11.16
2	70	2041	4456	31.15648161	0.7	9.96
2	80	1234	4243	29.66717942	0.52	9.04
3	10	3956	4633	32.39180591	5.34	8.23
3	20	3244	4626	32.34286513	2.6	7.95
3	40	2484	4562	31.89763669	0.92	8.04
3	50	1795	4495	31.42697336	0.59	7.78
3	60	1065	4334	30.30345406	0.42	7.4
3	70	661	4254	29.74201217	0.31	8.65
3	80	414	3913	27.35789694	0.24	5.83

Shashank Agarwal
Agarwal.202@osu.edu

Anurag Kalra
Kalra.25@osu.edu

4	10	1853	4938	34.52422569	2.4	5.53
4	20	1482	4933	34.48926799	1.27	5.4
4	40	1068	4879	34.11172481	0.48	5.62
4	50	762	4807	33.60833392	0.34	5.28
4	60	455	4617	32.27994127	0.25	4.93
4	70	304	4540	31.74159267	0.18	4.47
4	80	178	4117	28.78417115	0.14	3.92
5	10	1042	4852	33.92295323	1.42	4.35
5	20	853	4829	33.7621478	0.78	4.29
5	40	582	4820	33.69922394	0.33	4.37
5	50	417	4755	33.24477382	0.22	4.12
5	60	243	4673	32.67146752	0.15	3.84
5	70	164	4576	31.99328812	0.11	3.52
5	80	90	4306	30.10557226	0.08	3.11



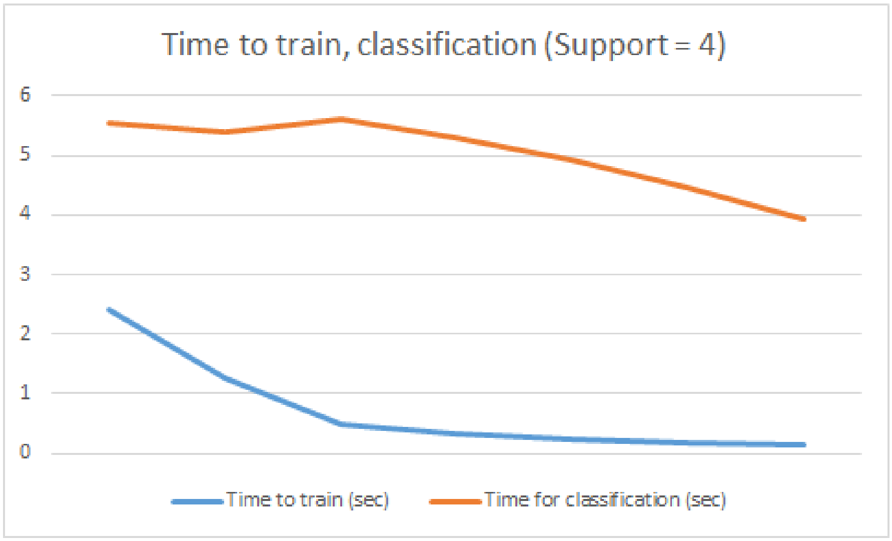
From the above table and graph we can see that the accuracy of the Classification model is not varying significantly with support and confidence values. However we can see that for higher values of confidence (here 80%), accuracy decreases.

Efficiency

We note the Time to train (Time to generate association rules + time to create classification model) and Time to classify (Time to predict labels for test data). We see that Time to train is very high for lower values of confidence and support - because of the high number of rules that are generated. Both the time measures decrease for higher values of support and confidence.

Shashank Agarwal
Agarwal.202@osu.edu

Anurag Kalra
Kalra.25@osu.edu



Algorithm 2

Procedure:

- Create feature_vector for the documents
- Use the feature vector to cluster (k=8,k=16) using kmeans [2]
- Create files based on clusters and also generate true labels file for each cluster
- Split and run 'apriori' on each file. Using training data from each cluster, generate classification model to be used for that cluster.
- For each cluster, perform classification to predict labels. Generate "accuracy" data for each model.

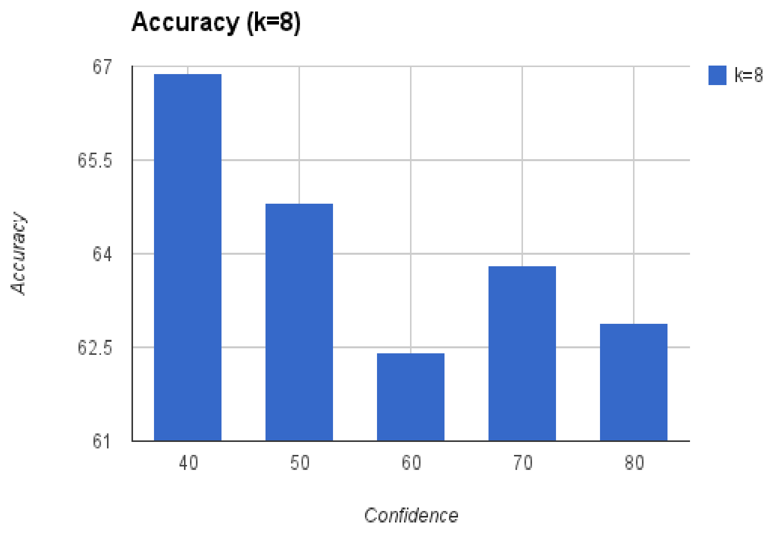
We used KMeans algorithm implementation from scikit-learn library[2] for clustering our documents. Documents present in a cluster were saved in separate files. For example, for k=8, we created 8 files to save documents in each cluster.
We process each file/cluster one-by-one. Same process as Algorithm 1 is followed: We split the documents in 80-20 ratio for training and testing. We run apriori software[1] on training data to generate association rules. The rules are ordered based on Confidence, support and order of generation to create the classification model. We run the classification model on testing data to predict labels for test documents. We compare the predicted labels with original labels to measure accuracy.

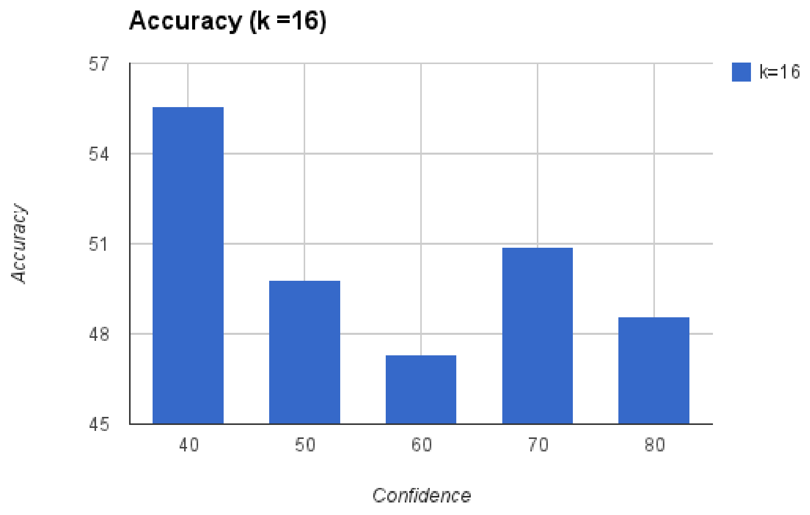
Accuracy

The below table shows comparison of average accuracy in k=8 & k=16 cluster classification. We kept the support value constant as 2. We see that even in Algorithm 2, changing the minimum threshold value of confidence does not have a significant impact on Accuracy.

(However, there is a huge difference between average accuracy value of Algorithm 2 and 1. It is discussed in later section)

confidence	Avg. accuracy	
	k=8	k=16
40	66.89882698	55.57534247
50	64.82699795	49.79908676
60	62.42222222	47.30672926
70	63.82699795	50.89827856
80	62.8937964	48.57534247





Comparison of accuracy for 8 vs 16 clusters

We observe that accuracy decreases when we use k=16 for clustering (although the accuracy for k=16 clusters is still better than Algorithm 1). The explanation for this observation is that for k=16, the number of documents per cluster is low. Thus the training data corpus is not big enough for good association rules to be generated.

Comparison between Algo 1 and Algo 2 (8 and 16 clusters)

We can observe that the Classification based on association rules is more accurate for Algorithm 2 compared to Algorithm 1. We can explain this observation by noting that a clustering algorithm clusters similar documents together. Thus when we generate association rules using the training data, we get good association rules with higher values of confidence. Also the probability of Testing documents matching the high confidence rules increases. Hence we see much better accuracy in text classification after clustering. However for higher values of K, accuracy decreases (though still better than without clustering).
On the other hand, the time taken for Algorithm 1 is much lower than Algorithm 2. In Algorithm 2, the clustering algorithm adds an extra overhead. Also, after clustering is completed we run the Apriori model generation and classification on each cluster serially (as we had a single machine). Thus Algorithm 2 takes more time for completion.

Contributions:

Shashank Agarwal & Anurag Kalra both implemented the "apriori" classification rules generation and testing scripts. The work overlaps and is hard to be segregated.

References

[1] Apriori implementation software: http://www.borgelt.net/apriori.html
[2] KMeans implementation library:
http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html