CSE 5243 Lab Assignment 5 - Report

Team Members and Contributions

(1) Pravar D Mahajan
(2) Anu Yadav

We both have worked equally on the project

Problem Description and Assumptions

In this assignment we explore the efficacy and efficiency of Classification by Association (CBA) classifier. We evaluate the performance metrics on the Reuters dataset, which consists of 20,000 documents with zero or many labels. The task is to use CBA classifier classifier which can accurately predict as many labels as possible. We use some of the key assumptions are as follows:

1) Stop words, punctuations and non-words tokens like numbers and dates have been considered noise and thus filtered from our dataset.
2) Documents with missing labels have been discarded.
3) Titles and places labels have not been considered for evaluation of efficacy or efficiency.

An Overview of the proposed solution

We propose the following steps to solve the problem. We will describe how we performed these steps in the later sections.

(1) Preprocessing - The dataset is in raw XML format, it needs to be cleaned, parsed and noisy words and tokens will be filtered out.
(2) Tokenization - Documents in raw text will be split into tokens and then converted into features in this step.
(3) Conversion to Transaction Format - In this step, we will convert the document tokens and their labels into the standard transaction format.
(4) Association Rule Mining - The documents and their associated labels in transaction format will be mined for association rules. These association rules should be of the form `<tokens> → class` so that they can be used for getting class labels.
(5) Training and Testing - The generated rules will be split into training and testing data. The time taken to train and test are computed.
(6) Benchmarking - This method of classification via association mining will be compared against the existing methods of classification used in Lab Assignment 2: K Nearest Neighbours and Decision Tree.

Details of our Approach to Solve the Problem

The following are the steps involved in analysing the efficiency and efficacy of CBA approach to classification of documents.

1. Tokenization and Features Generation: In the previous assignment on classification, we were using unigram/bigram models to to build features for our classifiers. For this assignment, we have used the same code for parsing, cleaning and tokenizing. However, instead of generating the document-terms matrix and class labels matrix, for each document we generate a set of tokens consisting of both document body tokens and class labels. To distinguish class labels from the the body tokens we prefix the class labels with a colon. For example, ‘cocoa’ is prefixed and converted into ‘:cocoa’.
2. Association Rule Mining: A document, which is a set of body tokens and class labels, can be treated as a transaction. Mining of association rules over the entire transactions dataset (ie, all

the documents and associated labels) will give us rules which can be used to determine class labels given tokens. However, care must be taken that we mine only for those rules which have document body tokens as antecedents and class labels as consequents. For mining rules, we used Christian Broglet's apriori software. The software allows us to distinguish between tokens which are to considered as antecedents from those which are to be considered as consequents only. We specify all the document tokens as antecedents and all the class labels as consequents and feed our transaction data into the apriori software.

3. Classification: We sort the rules in the order of decreasing confidence. If two or more rules have same confidence, we sort in decreasing order of support. For a new test case, starting from the first rule, we see which rule(s) is/are applicable. Since we are required to produce multiple labels per documents, we do not stop at first matched rule, instead we perform rule matching till we have upto *n* labels. We experimented with different values of *n* ranging from 1 to 15 and found that *n=10* gave the best results on the validation set. This seems reasonable since it is quite close to the average number of labels per document in our training dataset (~11.3)
   If no rules are applicable to our test case, we do not assign any class label to it.

Experimentation and Results: We split our dataset into training (60%), validation (20%) and testing (20%) sets. We experimented with different values of support and confidence, and recorded the training time (offline), testing time (online), number of rules generated and F1 scored observed. We chose F1 score as the evaluation metric because we observed that the model would otherwise learn to label all documents as belonging to all the classes, thereby scoring high on accuracy and recall, and low on precision. The observed results have been documented in Table 1, Table 2 and Table 3. The figure below shows a heatmap of F1 scores with respect to different values of support and confidence on the test dataset. The general definition of F1 score for evaluating binary classifiers was extended to multilabel setting by calculating F1 score for each class individually and taking an average across all the classes. The best F1 score observed on the test dataset was about 0.40, for support=2 and confidence=30

Figure 1: Heatmap of F1 score for different values of support(y-axis) and confidence (x-axis)

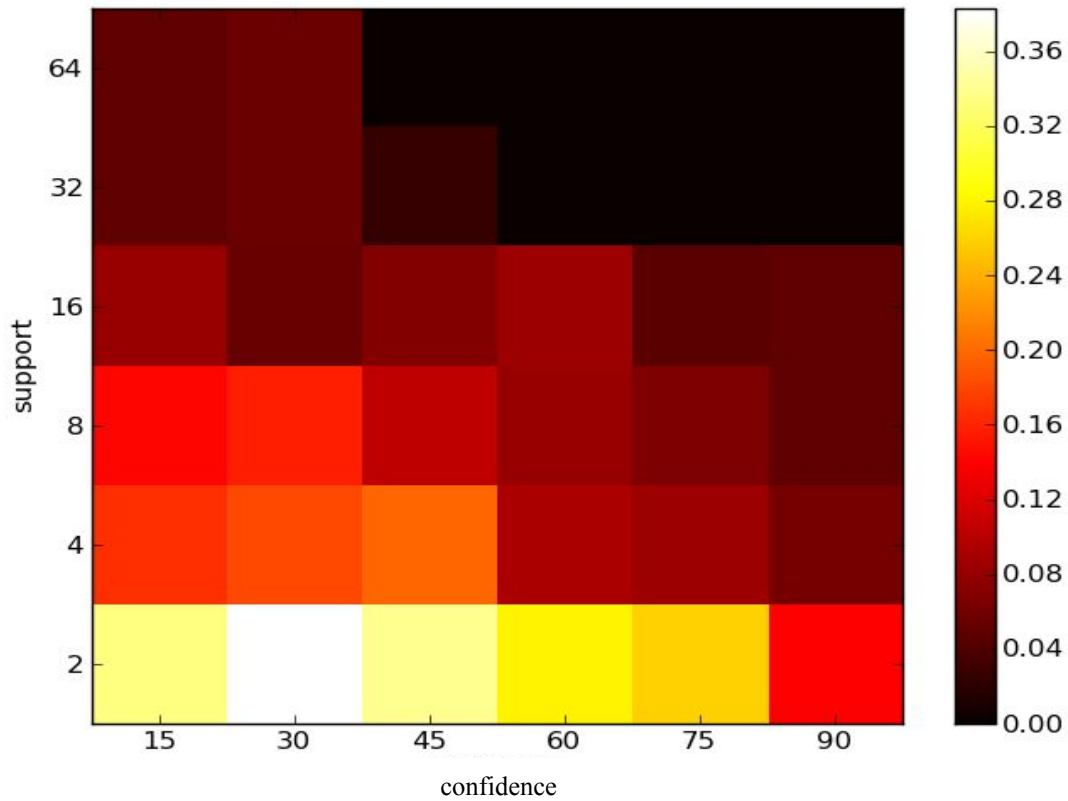

Table 1: Offline time with different supports (columns) and confidences (rows) in sec

	2	**5**	**10**	**20**	**40**	**70**
15	13.64	2.12	0.49	0.18	0.1	0.08
30	5.05	1.08	0.3	0.14	0.08	0.09
45	2.49	0.52	0.21	0.12	0.08	0.07
60	1.47	0.36	0.21	0.1	0.07	0.09
75	1.62	0.3	0.13	0.09	0.07	0.07
90	1.05	0.24	0.12	0.09	0.07	0.07

Table 2: Online time with different supports (columns) and confidences (rows) in sec

	2	**5**	**10**	**20**	**40**	**70**
15	44.92	4.48	0.62	0.13	0.07	0.07
30	18.44	2.23	0.35	0.1	0.07	0.07
45	11.96	1.67	0.28	0.09	0.06	0.06
60	9.2	1.36	0.23	0.08	0.06	0.06
75	7.61	1.2	0.2	0.08	0.06	0.06
90	6.38	1.04	0.2	0.08	0.06	0.06

Table 3: Number of rules generated with different supports (columns) and confidences (rows)

	2	**5**	**10**	**20**	**40**	**70**
15	53150	5965	762	101	8	3
30	20883	2748	383	56	5	2
45	13489	1961	272	41	1	0
60	10129	1537	211	28	0	0
75	8071	1279	180	25	0	0
90	6514	1138	165	24	0	0

Discussion on Observed Results:

(1) In general, we observe that the training (offline) and testing (online) times increase as we decrease support and confidence. Low thresholds imply that more association rules (as observed in Table 3) need to be generated and tested against, which requires more time.
(2) The heatmap suggests that as we decrease the support, better F1 scores are obtained. This is because decreasing the support generates more rules, which leading to better predictions. We also found that support values need to be as low as 2% to yield the best results, which can be explained by the fact that some of the class labels are very rare, occurring in 2% or less number of documents
(3) The trends for F1 scores are not very straightforward with respect to confidence values. We see that F1 score initially increases as we increase the confidence, but later decreases on very high values. This is because high values of confidence return very few rules, because of which we are not able to capture the rare class labels. However, for very low values of confidence, the model also captures noise in the training set, causing overfitting.
(4) Though the method of association based rule mining is able to perform the task of classification to some extent, it's performance is not as good as the KNN or Decision Trees. This is because the standard classification techniques are designed to perform well on these techniques, whereas association based mining is not. For example, KNN tries to find optimal centers which minimize the misclassification rate, thereby increasing the accuracy. The best result we obtained for Lab 2 was an F1 score of 0.65, compared to a score the best F1 score of 0.40 with CAB classifiers.
(5) The efficiency in terms of offline is better for CBA classifiers. This is probably because the conventional classification techniques require several passes over the entire dataset to find optimal parameters, whereas CBA classifiers are based on set intersection and union techniques which are generally very fast. The offline training for KNN and decision tree were in the range of 30 secs to few minutes, compared to 0-15 secs for CAB classifiers.
(6) However, once the optimal parameters have been obtained, conventional classifying algorithms work very fast. Both KNN and decision tree algorithms take fraction of a second to obtain the label for all the test instances, whereas the CAB classifiers take a lot of time, specially for low support/threshold values. This is because the testing step requires going through all (or most) of the rules, which is a slow process.

Code Details

The script *cba_classifier.py* is the backbone of our code. Its main function calls *convert_to_transaction* function to convert the input dataset into transaction format (<tokens> → class). Then it calls another method to execute apriori algorithm. After that rules are tested in t*est* function which takes sorted rules. The rules generated are sorted by *get_sorted_rules* method. Inside the main function, we are also computing the offline cost and online cost and also generating a heatmap graph between support and confidence.

Libraries Used

Scipy, Numpy, NLTK and MatplotLib