Document Classification using Naive Bayes Classifier

Ramabhadra V [1]

Abstract

We aim at implementing Naive Bayes Classifier for Document Classification in Local machine and at Cluster.

1. Brief Method Sketch

The steps followed in predicting the class of a document are as follows.

- Count the Number of occurrences of each word in the training set. Output of this step will be the following dictionary.

$$\{word : \{class : number\}\}$$

- Count the Number of occurrences of each class and the number of words in each class. Output of this step will be the following dictionary.

$$\{class : [Occurrence, words]\}$$

- The test set requirement is built in this fashion. word ĩd

- Final dictionary will be constructed by combining word count and test set requirement. Output of this step will be the following dictionary.

$$\{id, id - class : \{word : \{class : number\}\}\}$$

  where, id-class represents the class to which id belongs to.

- The classification of the document will be based on the final, class count dictionary built, based on the algorithms discussed in next section.

2. Algorithms

I have implemented 2 Algorithms which give different accuracy on different architectures.
Algorithm discussed in class: This performs better when Naive Bayes is implemented on Hadoop MapReduce framework.

$$\begin{aligned} &logPr(y', x_1, x_2, .., x_d) \\ &= \sum_j log\frac{C(X = x_j \wedge Y = y') + mq_x}{C(X = ANY \wedge Y = y') + m} \\ &+ log\frac{C(Y = y') + mq_x}{C(Y = ANY) + m} \end{aligned} \quad (1)$$

-Return the best y'
Algorithm mentioned here[1]: This performs better on local machine.

$$\begin{aligned} &logPr(y', x_1, x_2, .., x_d) \\ &= \sum_j log\frac{C(X = x_j \wedge Y = y')}{C(X = ANY \wedge Y = y') + m/q_x} \\ &+ log\frac{C(Y = y') + mq_x}{C(Y = ANY)} \end{aligned} \quad (2)$$

where, q_x represents 1/(number of unique words in the training set)
-Return the best y'

3. Architectures

I implemented Naive Bayes in 3 different architectures.

1. **Local Implementation type 1:** Here I have used a simple python implementation of Naive Bayes classifier without using the above method sketch discussed.
2. **Local Implementation type 2:** In this, I used above discussed methodology and ran the programs by using pipes.
3. **Hadoop Implementation:** I used Hadoop MapReduce Framework to implement the the above discussed methodology.

I have summarized the outcomes of the above methods in the table shown.

Table 1. Classification accuracies, time(in minutes) for Naive Bayes on Local machine and cluster for various data sets.(A refers to Algo 1, B refers to Algo 2, C refers to type 2 implementation.)

TYPE	TEST SET	VALIDATION SET	TRAINING SET
LOCAL, A	25.09, 33	30.1, 64	47.40, 182
LOCAL, B	70.04, 33	69.92, 61	69.45, 182
LOCAL, C	13,3	13,4	10,5
CLUSTER, A	76,8	63,16	71,64
CLUSTER, B	54,8	52,16	-

4. Time taken for training vs Number of Reducers

The figure shows the plot of the time taken for training the model vs the number of reducers used. As we can observe, the time initially decreases as number of reducers increase. However, it again starts to grow as number of reducers reach a certain limit. This happens because of this reason. Initially, the increase in number of reducers makes data to be processed in parallel and hence decreasing the execution time. However, increase in number of reducers increases IO costs (splitting the data and sending it to the corresponding reducer). Hence, as we increase the number of reducers, IO cost will keep on increasing but speedup due to parallel execution will be bounded by Amdahl's law[2].

Therefore, the increase in number of reducers will help only till a certain point and after which it will affect adversely.

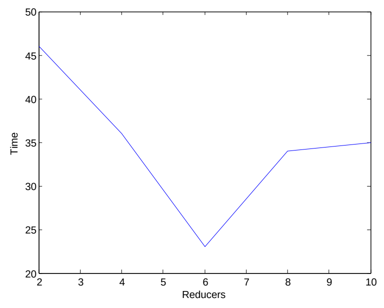

Figure 1. Number of reducers vs time (in sec).

5. Help Sought

1. Bhuthesh helped in resolving some errors in Hadoop.

6. References

1. https://www.3pillarglobal.com/insights/document-classification-using-multinomial-naive-bayes-classifier
2. https://en.wikipedia.org/wiki/Amdahl%27s_law