Naive Bayes Implementation using MapReduce

Suraj Panwar, Mtech DESE, 14644

1. Introduction

The project undertakes the task of implementation of Naive Bayes on Hadoop based MapReduce platform and compares the performance on the basis of accuracies obtained and the time taken for the implementation of the task. The dataset used for this project is extracted from DBPedia which is a 50 class classification problem with multiple labels per example.

The random guessing accuracy for this dataset considering only 1 class per example is 2% and if we consider the possibility of multiple labels, this further decreases. This is taken as the baseline accuracy for this project. Due to having multiple labels the algorithm classification becomes a difficult task, however, during the classification we consider the most probable label provided by the Naive Bayes algorithm and if it is one of the true classes it is considered as the correct classification.

This project also explores the application of MapReduce framework for the task of the Naive Bayes Implementation by assigning the task of constructing the dictionary to the framework and build upon the results obtained thereafter. Along with that the effect of increasing the number of reducers is also recorded upon what is essentially a counting task.

All the mentioned results have been recorded and mentioned in this report along with possible explanations of the cause. The algorithms are however compared mostly on the grounds of test accuracy and execution time.

2. Naive Bayes Implementation

2.1. Local Naive Bayes

The local Naive Bayes has been implemented on a personal laptop with no distributed implementation. The main training corpus has been the DBPedia from which the test and dev sets have also been extracted. The model has been implemented using smoothing method to increase the model performance. The model is updated using performance on the development set and finally the accuracy is evaluated on the test dataset provided.

The probability of a class for a given sample have been modeled using multiplication of the conditional probabilities of the words in the sample. The class with the highest probability is assigned as the class label. If the class label is one of the given labels, the sample is considered as correctly classified.

2.2. Naive Bayes on Hadoop MapReduce

The Naive Bayes is implemented in the Turing cluster with variable reducer workers, the results of which are plotted in the following sections. The data present is converted to a (label, word, count) database using the mapper and reducer functions, programs of which are uploaded in the git hub repository. The original data is converted to the (label, word, 1) using the reducer, where the label and corresponding word are the class and word of the sample, in case of a multiple class case the sample words have been considered in creating the vocabulary for both the classes.

- **Mapper Output:** (label, word, 1)
- **Reducer Output:** (label, word, count)

The streaming input provided from the mapper function is processed using the reducer which collapses the obtained counts of the streaming input to provide the total number of occurrences of the words which have then been used as the base dictionary of the vocabulary of the different classes, where every sample of the reducer output is finally processed as (label, word, count) where the count is the total count of word occurrence in the label. The data is then used a dictionary and the Naive Bayes program from the previous case is run on the dataset to obtain the class. The program is then implemented using Laplacian smoothing like in the previous case. The dictionary which was constructed by iterating over all the words in the corpus as in the previous case is instead loaded from the MapReduce output in this case which as we will see is the most time time consuming part of the algorithms and hence helps in reducing the implementation time of the algorithm. The classification accuracy is calculated as in the previous case by using the class with highest joint sample probability using the Naive Bayes.

Table 1. Classification accuracies for Local Naive Bayes and MapReduce based Naive Bayes.

	LOCAL	MAPREDUCE
TRAIN SET	93.26 %	86.33 %
DEVEL SET	71.01 %	67.16 %
TEST SET	73.41 %	66.10 %

3. Performance Comparison

The performance of the Local Naive Bayes and the MapReduce Naive Bayes is compared on basis of various metrics and the results have been recorded as follows:

3.1. Accuracy

The accuracy is calculated on the training, test as well as the development partition, the sample is considered to be correctly classified if the sample predicted class is amongst one of the actual classes of the sample. The accuracy observed in both of the applications is recorded in the given table. The accuracy is computed using the train corpus in the local machine and using MapReduce to obtain the dictionary of the DBPedia corpus provided on the hadoop files system.

The accuracy obtained from the local Naive Bayes implementation is slightly higher than the MapReduce counted Naive Bayes especially in the training data partition. The high training accuracy of the training dataset compared to the development and test partition suggest over fitting in the model and can be explored further for improving the model performance by using better smoothing methods or better language model such as Kneser Ney models, Back-off models, N gram model, etc.

3.2. Model Parameters

The model parameters in case of Naive Bayes are the probability estimates that we calculate for carrying out the classification, as in this case we are using smoothing thus all the words for all the classes have a non zero probability of occurring, hence we are left with a probability distribution for each word in the vocabulary of the training corpus for all the classes.

Using this idea and taking into consideration calculation of the prior probability for all the classes we have:

- **Vocabulary size:** 374680 words
- **Parameters:** 374680 x 50 + (50 - 1) = 18734049 parameters

Where the 49 parameters are due to the prior class probabilities that we have to calculate for the given 50 classes as the last class prior can be calculated as the difference of the rest of the priors from one, hence one less parameter. We can see that for such a large vocabulary the parameters are considerably less compared to exponential scaling otherwise.

3.3. Timing

The timing performance of the Naive algorithm is compared in two ways, first from the perspective of the effect of the number of reducers on the algorithm performance which is recorded in the next section and the wall time for the training for the Naive Bayes model in both the local as well as cluster implementations. The results are recorded as follows, however it should be noted that the wall time for the MapReduce implementation is for a single reducer only and as we increase the number of the reducers, the timing will further decrease, thus further improving from the Local implementation.

- **Local NB :** 170 sec (approx)
- **MapReduce NB (1 reducer) :** 120 sec (approx)

Compared to local implementation the MapReduce provides significant time improvement which is further enhanced as we increase the number of reducers, which can be seen in the next section showcasing the suitability for this application.

4. Effect of Reducers

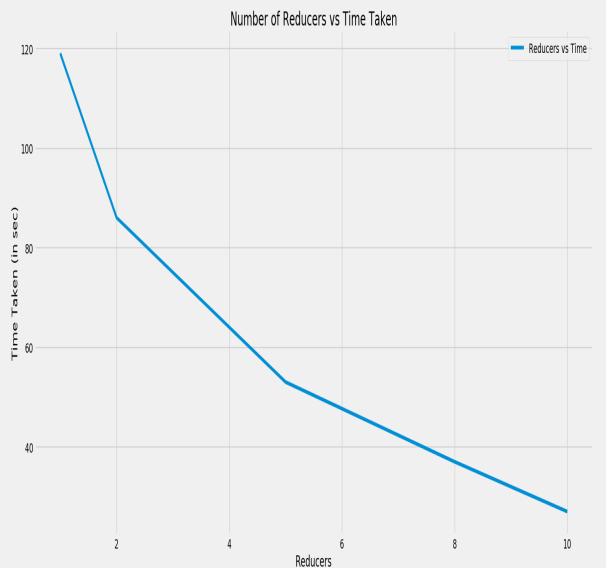

Figure 1. Computation time vs Number of Reducers

For exploring the effect of the number of reducers on the computation time the number of reducers is varied from one

to ten and the timing for the MapReduce operation is plotted as shown in the graph. The important inference that we can examine from the plot is that with increase in the number of the reducers the computation time decreases almost linearly due to parallelization across different reducer workers.

However a slight tailing effect can be seen in the plot which suggests that continuous addition of reducers does not provides equal decrease in the computation time for the operation, which could mean a optimal number of reducer numbers compromising between allocated resources and the implementation time.

5. Github Resource

Project program files along with the required resources have been uploaded at github at the following link:

https://github.com/Suraj-Panwar/Naive-Bayes-using-MapReduce.git

Readme provides the required insight on the project documents uploaded.