Naive Bayes Classifier implementation on Hadoop

Dola Ram, DESE IISc, 15197

1. Introduction

The Naive Bayes classifier greatly simplify the learning of probability distribution by assuming that features are independent given a class. In this assignment study observed the effect of preprocessing such as lemmatization and removing stop words on the classification accuracy of naive Bayes classifier on large data set. Naive Bayes implemented on local machine as well as on Hadoop based on mapreduce. The dataset used has been taken from DBPedia which is a 50 class classification problem with multiple labels per example. 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.

2. Naive Bayes Classifier

2.1. Implementation on local Machine

In this the naive bayes classifier was implemented on single local machine. Naive Bayes has been implemented using Laplacian smoothing to increase the accuracy of classifier.

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.

(a) For smoothing, let $m = 1$ and let $p_0 = \frac{1}{V}$.

(b) For each possible class $y \in Y$:

- Estimate the log-probability $\log P(y|w_1, \ldots, w_{n_i})$ as

$$S_y = \log \frac{C[\text{“Y=}y\text{”}]}{C[\text{“Y=ANY”}]} + \sum_{j=1}^{n_i} \log \frac{C[\text{“Y=}y \text{ and W=}w_j\text{”}] + mp_0}{C[\text{“Y=}y \text{ and W=ANY”}] + m}$$

- predict $\hat{y} = \text{argmax}_y \, S_y$

Figure 1. Naive Bayes classifier.

Here V is the size the vocabulary. m = 1 is taken as Laplacian smoothing was used for implementation.

4 different preprocessing methods was used to increase the performance of the classifier are: -

1. **Lemmatization: -**

   Lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For example Cats is converted to Cat.

2. **Removing StopWords: -**

   Some words carry more meaning than other words and some words are just plain useless, and are Stop words. Stop words does not contribute that much in the classification so has been removed during preprocessing step.

3. **Removing punctuation and web link: -**

   Removing the unnecessary punctuations and web link increased the accuracy of classifier.

4. **Removing Digits: -**

   The numbers in the unnecessary increases the vocabulary and computation so numerical digits in the database has been during preprocessing.

2.2. Naive Bayes Implementation on Haddop

Training corpus taken from the DBPedia have the the formate **(class-labels, text)**. Firstly the dataset was converted to **(class-label, word, 1)** using the mapper function and the to **(class-label, word, count)** using reduce. Here class-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. Count here is the number of times a word appear in the particular class.

Data from dataset is taken by the mapper through Standard input. Mapper reprocesses the input text coming.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 classification accuracy is calculated as in the previous case by using the class with highest joint sample probability using the Naive Bayes.

3. Performance Of Naive Bayes

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

Accuracy of the Naive Bayes implementation on the local machine is found to be greater than accuracy of implementation on MapReduce. 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.

Table 1. Classification accuracies of two implementations

DATA SET	MAPREDUCE	LOCAL MACHINE
TRAIN	90.71%	96.01%
DEVEL	69.05%	73.53%
TEST	67.01%	77.44%

The difference in between the accuracy on training set and test set is about 10% that shows model is over fitting to training set. Train set accuracy without preprocessing step found to be 58% and after using preprocessing methods such as removing stop words, punctuation and numerical digits and lemmatization increases the train set accuracy to 96%.

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. Size of vocabulary without preprocessing was 997153 and after doing the preprocessing vocabulary size reduce to 374680. Number of parameters in in the Naive Bayes classifier implementation as follows: -

- **Vocabulary size without preprocessing:** 997153 words
- **Parameters:** 997153 x 50 + (50 - 1) = 49857699 parameters
- **Vocabulary size with preprocessing :** 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.

3.3. Timings

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. Total computation time for two implementation is given below:

- **Local Naive Bayes :** 243 sec
- **MapReduce Naive Bayes(10 Reducer):** 91 sec

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.

4. Effect of Numbre of Reducers

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.

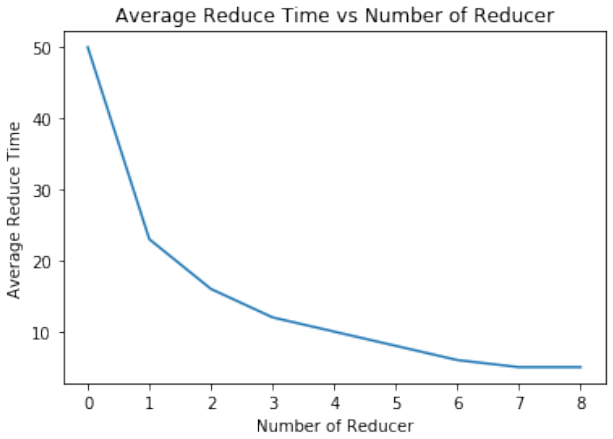

Figure 2. Average Reduce Time(sec) vs Number of Reducer.

5. Github Resource

Project program files have been uploaded at github at the following link: https://github.com/dolaram/Naive-Bayes-using-MapReduce.git