Document Classification using Naive Bayes Classifier

Sahil Deep [*1]

Abstract

In this assignment, the training and testing of a Multinomial Naive Bayes Classifier for classification of DBPedia dataset was implemented with in-memory and streaming based Map Reduce approach.

1. Preprocessing on Dataset

- Each document was converted into words using $wrds = re.findall('w+', document)$ as tokenizer.
- Words were converted to lower case.
- No stopword removal or stemming was done.
- As a result, the vocabulary consisted of 301404 words(for full training data).

2. Basic Implementation

- On the given Train and Test dataset initailize an event counter(hash table) C.
- For each example $id, y, x_1, ...x_n$ in train:
  $C = ("Y = ANY") ++; C("Y = y") ++$ (b) for j in $1, ...d_{id}$:
  $C("Y = y" \wedge x = ANY) ++ c("Y = y \wedge X_j = x_j") ++$
- For each example id ,$y, x_1, ..., x_d$ in test
  (a) For each y' in domain(y)
  (b) Compute $Pr(y', x_1, ...x_d)$ as $a + b$ where a and b are as follows:

$$a = \frac{\sum_{j=1}^{d} log(C(X = x_j \wedge Y = y_j) + mq_j}{C(X = ANY \wedge y = y') + m} \quad (1)$$

$$b = \frac{logC(Y = y') + mq_y}{C(Y = ANY) + m} \quad (2)$$

- Return best y'

3. In Memory Implementation

In Local Naive Bayes' implementation the algorithm is implemented same as above with in memory hash table.

4. Parameters

Total number of parameters are 1195617. Out of these 1195517 are of type $C(X = x_j \cap Y = y')$, 50 are for $C(X = ANY \cap Y = y')$ and 50 are for the class frequency.

5. Hadoop Implementation

- Reorganize C to make C'.
  - Stream through C and for each event count (e, n_e), let W by the word such that $e = Y = y$ and $W = w$. Print the message (w, e, n_e) to temporary file 1.
  - Sort temporary file 1 by word w.
  - Stream through the sorted temporary file 1 and convert it to pairs $(w, wordCounts_w)$, exploiting the fact that all the event counts for w are now stored sequentially. Store the output in file C'.
- Produce word-count requests: For each document $i, (w_1, ..., w_n)$:
  - For each word position j in document i, write a message (w_j, i) to temporary file 2.
- Produce word-count replies:
  - Concatenate C' and temporary file 2, sorting them by word, so that the word-count records from C' precedes the request, from temporary file 2.
  - Stream through the resulting file as follows:
    - If a line contains a tuple $(w, wordCounts_w)$, save that record in memory. (It will be small, containing one count for each possible label y.)
    - If a line contains a tuple (w, i), then print a message $(i, w, wordcounts_w)$ to temporary file 3.
  - Sort temporary file 3 by document id i.

Table 1. Training Time

LOCAL/REDUCERS	TRAINING TIME
LOCAL	10M34.34S
MAPREDUCE(R=1)	4M7.399S
MAPREDUCE(R=2)	3M34.635S
MAPREDUCE(R=5)	3M17.987S
MAPREDUCE(R=8)	3M4.691S
MAPREDUCE(R=10)	3M1.233S

Table 2. Testing Time

TEST TIMING	FULL TEST	FULL TRAIN
LOCAL	0M51.09S	
MAPREDUCE(R=1)	3M19.297S	16M33.3075S
MAPREDUCE(R=2)	3M12.428S	22M48.576S
MAPREDUCE(R=5)	3M22.053S	15M42.297S
MAPREDUCE(R=8)	3M19.258S	16M40.745S
MAPREDUCE(R=10)	3M34.095.961	15M33.320S

– Stream through temporary file 3, accumulating together all the sequentially contiguous tuples $(i, w, wordCounts_w)$ for a single document i, and saving them in memory. When this sequentially contiguous block is finished, output a message $(i,)$ where is the predicted class for i.

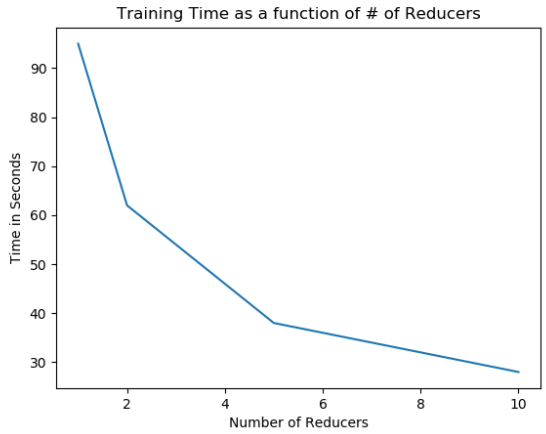

Figure 1.

6. Experimental Analysis

The figure 1 above plots the training time with respect to number of reducers. The relation is not linear and we can observe that increasing the number of reducers decreases the training time to some extent but later the the decrease in training time is not that prominent(**may be due to more IO operations**).

Since the algorithm implemented in both the cases is the same thus both of them have the same accuracy over training,development and testing datasets.

The Reducer vs Time in the table 4 is measure for stage 1 of the training by measuring the time between mapper 100% reducer 0% to mapper 100% reducer 100% on training dataset. This measurements been used for plotting the figure 1

Table 3. Classification Accuracy

IMPLEMENTATION	TRAINING	DEVELOP	TESTING
LOCAL	95.62%	75.87%	77.89%
MAPREDUCE	95.62%	75.87%	77.89%

Acknowledgements

Table 4. Reducer vs Time

REDUCER COUNT	TIME(S)
1	95
2	62
5	38
8	32
10	28