Bigdata Analytics Final Project submission

Aftab Hassan Swathika Balasundaram

June 9 2015

1 Introduction

Internet networks now-a-days increasingly support the ability for users to express opinions and publicly evaluate website content. With the increasing popularity of on-line shopping, Amazon supports the ability for users to both rate and review products that they purchase. The goal of the project is to predict how helpful a review is, based on certain parameters.

2 Problem Definition

Reviews given by users on a shopping website play an important role in the decision making process of buying a product. Though there are increasing number of online reviews, it is difficult for users to find a helpful review for purchasing the product they would like. Therefore most online review websites allow other users to rate the helpfulness of a review. So, in this project, based on the score given to a review and it's text length, we try and predict the review helpfulness. The review helpfulness rating is fundamentally different from the item rating. The former indicates "how does a user X rate a review from another user Y ?" while the latter denotes "how does a user X rate an item?". These differences not only are useful to differ our studied problem from item rating prediction problem, but also present unique opportunities for us to investigate the review helpfulness rating prediction problem.

3 Dataset

The dataset we will be using is the Amazon fine foods dataset. It contains approximately 500,000 reviews. Reviews include information such as review text,rating given by the reviewer for the product, review helpfulness score given by other users for that review, in addition to identification parameters such as product ID, reviewer ID etc.

A typical row in the data set looks as follows

productId	userId	profileName	helpfulness	score	time
B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1/1	5.0	1303862400

summary	text
Good Quality Dog Food	I've bought Vitality products and found them to be of good quality.

4 Data preprocessing

The given data set is converted into a comma separated file with eight columns and around five hundred thousand rows where each row corresponds to a product review.

We remove all rows from the data which have not been reviewed for helpfulness yet, that is, rows which have a review score of 0/0, as models cannot be trained on data whose ground truth is not known.

5 Algorithm

5.1 Baseline Algorithm - Average of a particular author's review

Here, we group all the reviews belonging to a particular author and predict the review helpfulness as the average of the review helpfulness scores of all of that author's reviews so far. In doing so, we assume that the quality of an author's review will remain consistent across any review and any product. This is a reasonable assumption since, we generally like to follow reputed critics, be it in any field.

5.2 Multiple Linear Regression

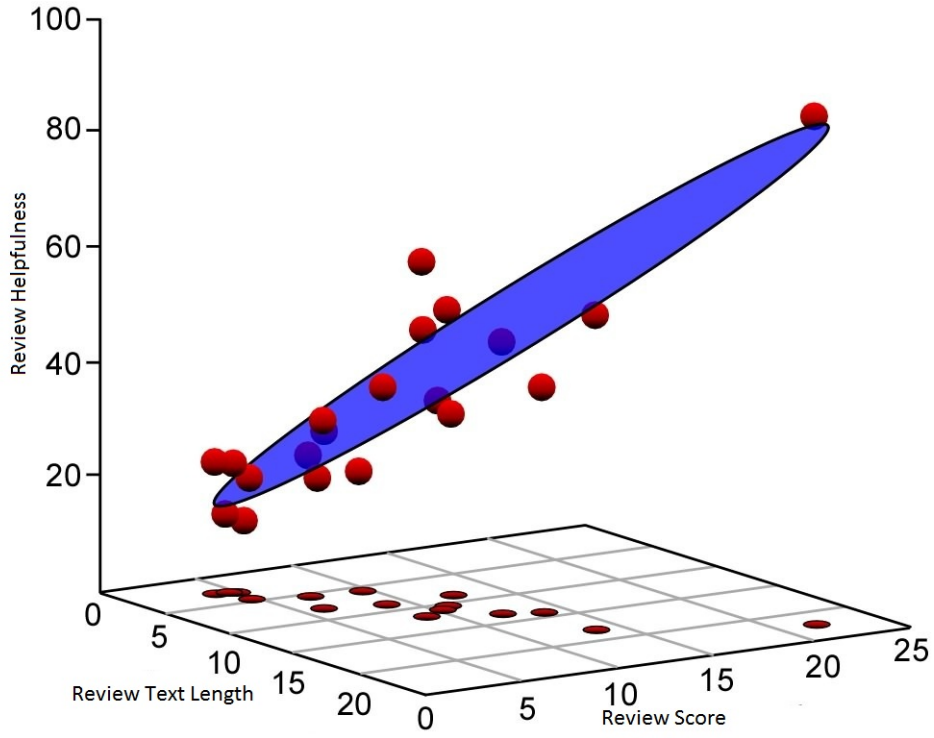
Since our problem statement requires us to predict a number conveying the review helpfulness, we would be using Multiple linear regression.

Multiple linear regression is an extension of simple linear regression. It is used when we want to predict the value of a response variable based on the value of two or more predictor variables. In our project review helpfulness is the response variable and review score and review text length are the predictor variables.

Formula for Multiple Linear Regression :

$$y = c_1p_1 + c_2p_2 + c_3p_3 + c_4p_4 + c_5p_5 + ...$$

where, y is the response variable(review helpfulness), p1, p2, p3.. are the predictor variables, and c1, c2, c3 .. are the co-efficients of these predictor variables



6 Implementation

We will be implementing our solution on three different frameworks and compare performance.

6.1 R implementation

The function called 'lm' provides support for multiple linear regression. We will be using this function keeping review helpfulness as the response variable and review text length and review score as the predictor variables

In the below example, we try to use the linear regression function (lm) provided by R to perform multiple linear regression and predict review helpfulness given review text length and review score.

```
> df
  reviewhelpfulness reviewscore reviewtextlength
1                 5           5                20
2                 5           6                33
3                 8           9                78
4                 8           7                78
5                 8           8                98
> lm(reviewhelpfulness ~ reviewscore +  reviewtextlength)

Call:
lm(formula = reviewhelpfulness ~ reviewscore + reviewtextlength)

Coefficients:
     (Intercept)       reviewscore  reviewtextlength
         3.17505           0.15852           0.04097
```

As seen above, this function returns the co-efficients of the two predictor variables and the y intercepts. So the equation of the regression line would be

$$ReviewHelpfulness = 0.15852 \times ReviewScore + 0.04097 \times ReviewTextLength + 3.17505$$

Once we have this equation, we will be able to predict the review helpfulness for any new data, by substituting the corresponding values of review text length and review score.

6.1.1 R pseudocode

Algorithm 1 R implementation

```
Load Dataset
Folds = 10
for all folds do
  Separate data into train and test
  formula ←ReviewHelpfulness ~ReviewScore+ReviewTextLength
  lm(ReviewHelpfulness ~ReviewScore + ReviewTextLength)
  for all test data do
    Substitute ReviewScore,ReviewTextLength to predict ReviewHelpfulness
    Compute RMSE
  end for
  folds++
end for
```

6.2 Python implementation

Similar to the R implementation above, we use the python package sklearn to perform Multiple Linear Regression using Python.

6.2.1 Python pseudocode

Algorithm 2 Python implementation

```
from scipy import stats
Load Dataset
Folds = 10
for all folds do
  Separate data into train and test
  stats.linregress(ReviewHelpfulness,ReviewScore + ReviewTextLength)
  for all test data do
    Substitute ReviewScore,ReviewTextLength to predict ReviewHelpfulness
    Compute RMSE
  end for
  folds++
end for
Report Average RMSE across all 10 folds
```

6.3 Hadoop implementation

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. In this project, we will use a single node Hadoop cluster.

Since there are no packages yet to perform multiple linear regression in Hadoop, we will be implementing it as follows

6.3.1 Calculating co-efficients

$$
\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \ldots + \beta_k x_{1k} + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \ldots + \beta_k x_{2k} + \epsilon_2 \\
&\ldots \\
y_i &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik} + \epsilon_i \\
&\ldots \\
y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \ldots + \beta_k x_{nk} + \epsilon_n
\end{aligned}
$$

Each line in the above equation corresponds to a row in the data set.

The system of n equations can be represented in matrix notation as follows:

$$y = X\beta + \epsilon$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & . & . & . & x_{1n} \\ 1 & x_{21} & x_{22} & . & . & . & x_{2n} \\ . & . & . & & & & . \\ . & . & . & & & & . \\ . & . & . & & & & . \\ 1 & x_{n1} & x_{n2} & . & . & . & x_{nn} \end{bmatrix}$$

The matrix X is referred to as the design matrix. It contains information about the levels of the predictor variables at which the observations are obtained. The vector B contains all the regression coefficients. To obtain the regression model, B should be known and is estimated using the following equation: $\hat{\beta} = (X'X)^{-1}X'y$

6.3.2 Building linear regression with MapReduce on Hadoop

We save the co-efficients received in the above step in a file called coefficients.csv. Every node needs this file to calculate the predicted review helpfulness value, so this file needs to be distributed to every node. This can be done in Hadoop with the distributed cache functionality which assures that every node has the file before it starts the calculation.

Once this is done, the Mapper multiplies the predictor variables, review text and review score with the corresponding coefficients received in the above step to compute the predicted review helpfulness.

For each record, the mapper also calculates the square of the error between predicted and actual values(called 'costs') by subtracting the predicted review helpfulness from the true value and squaring this difference.

The reducer then adds all these costs from each mapper and finds the average to compute the average root mean square error(RMSE).

7 Evaluation and Results

We use two evaluation measures in our project.

7.1 Evaluation Metrics

1) Root mean Square Error

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

To construct the r.m.s. error, you first need to determine the residuals. Residuals are the difference between the actual values and the predicted values. I denoted them by $\hat{y}_i - y_i$, where y_i is the observed value for the ith observation and $\hat{y}_i$ is the predicted value.

2) Mean Absolute Error :

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i|.$$

The mean absolute error is an average of the absolute errors $|e_i| = |f_i - y_i|$, where f_i is the prediction and y_i the true value. Note that alternative formulations may include relative frequencies as weight factors.

The results of the various implementations based on the above metrics are as follows. As can be seem, the errors using the R and python packages are smaller as compared to our linear regression implementation on Hadoop framework. Also, both R and python beat the Average Baseline.

Metrics	R	Python	Hadoop	Average Baseline
MAE	0.24	0.23	0.38	0.28
RMSE	0.32	0.31	0.40	0.34

8 Graphs

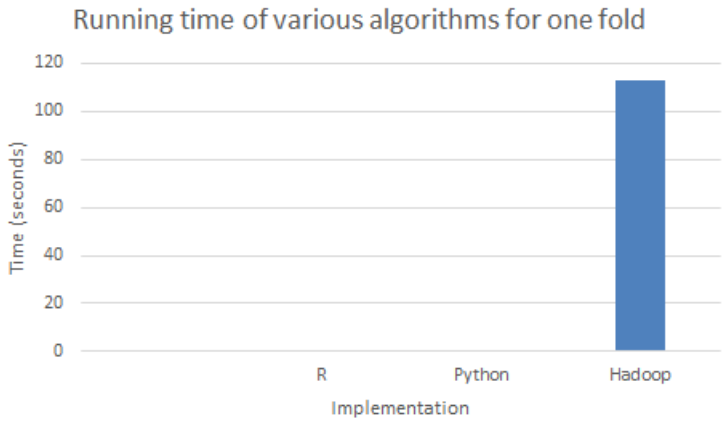
We compare the performance, in terms of MAE and RMSE as well as running time of the three implementations.

8.1 Time comparison

Below graph shows the running time comparison between R,Python and Hadoop. The y axis shows the time in seconds and the x axis shows the three different implementations. The running time of Hadoop is larger since the implementation is done on a single node cluster. Please note that these are running times of a single fold, since the Hadoop implementation was done for one fold taking ninety percent of the data for training and the remaining ten for testing.

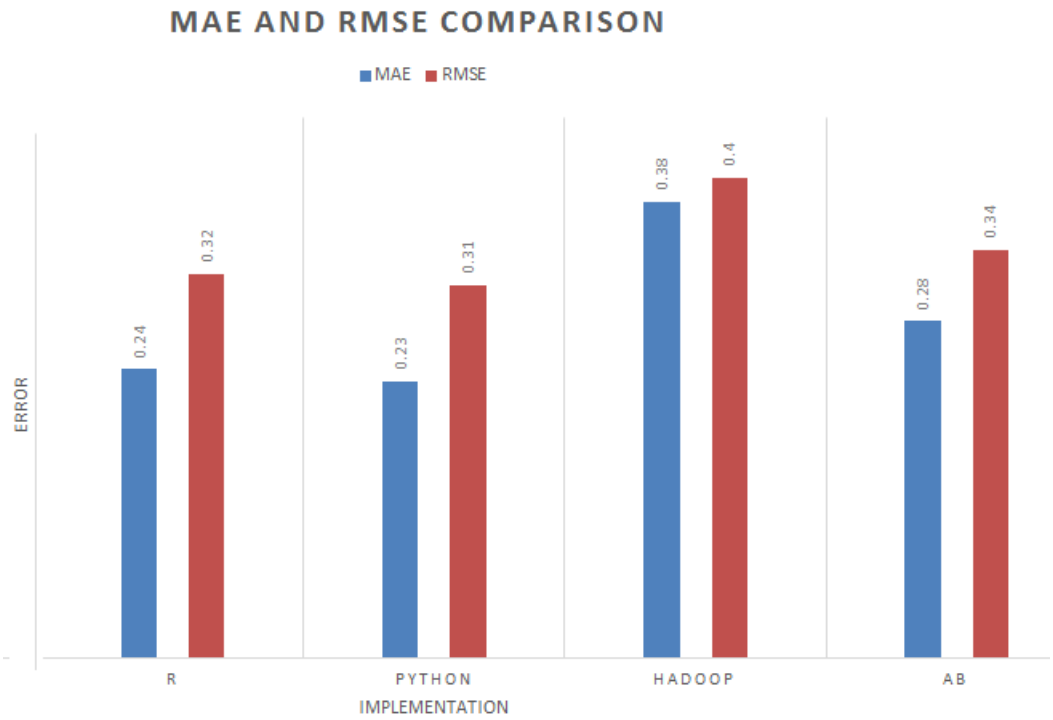
The running times (in seconds) are as follows

R	Python	Hadoop
0.57224	0.129	88



8.2 MAE and RMSE comparison

Below graph shows the mean absolute error(MAE) and root mean square error(RMSE) comparison between our R,Python and Hadoop implementations. The y axis shows the MAE and RMSE values and the x axis just shows the three different implementations. The errors using the R and python packages are smaller as compared to our linear regression implementation on Hadoop framework. Also, the errors using R and python packages are smaller than the Average Baseline.



References

[1] C. Yeung and T. Iwata. *Strength of social influence in trust networks in product review sites. WSDM, 2011..*

[2] J. Liu, Y. Cao, C. Lin, Y. Huang, and M. Zhou. *Low-quality product review detection in opinion summarization. EMNLP-CoNLL,2007.*

[3] Jiliang Tang, Huiji Gao, Xia Hu and Huan Liu. *Context-Aware Review Helpfulness Rating Prediction.*

[4] Yang Liu,Xiangji,Huang,Aijun An,Xiaohui Yu *Modeling and Predicting the Helpfulness of Online Reviews*