Ranking and Prediction of Amazon Fine Food based on Costumer's rating and review

Ng Wing Hin
Chinese University of Hong Kong
1155106860@link.cuhk.edu.hk

Lei Wen Fen
Chinese University of Hong Kong
1155002533@link.cuhk.edu.hk

Cheng Tin Chu
Chinese University of Hong Kong
1155106571@link.cuhk.edu.hk

Chan Hei Nam
Chinese University of Hong Kong
1155103957@link.cuhk.edu.hk

ABSTRACT

This project uitilize the Amazon Fine Food data set from Kaggle to implement a system that can predict rating based on user's comment. We will be using varies algorithms from big data to anaylsis the data set. In the end of this project, we are expected to observe the relationship between user's comments and rating. By comparing the result to test data we can do prediction of rating based on the similarity of the comments. Moreover, we will come up with the most and least popularity of products based on rating.

KEYWORDS

Big Data System, Ranking Algorithm, Document anaylsis, Prediction Model

1 INTRODUCTION

Currently, we are all living in a social system under capitalism. Every enterprise is competing with each other with their own products, services and even user experience. With the rapid growth of the internet in recent year, user experience can be further improved via processing huge data on costumer's review and rating. The Large internet-based retailer, like Amazon, have an enormous number of products but obviously, not all of them are popular. Therefore, processing costumer's review and rating of a product is vital for improving user experience thus increasing profit.

In order to determine the popularity of a product, an explicit way is to process costumer's review and rating. Nonetheless, the number of responses on a product would not always be sufficiently large enough for reference. In the Amazon Fine Food Reviews, there are user's scores and reviews on different products and our goal is to investigate the popularity of products by these two factors. This will benefit online retailer to promote further actions to enhance user experience and yield more revenue. Moreover, the system would be able to do predication of rating of the product based on the comment.

2 PROBLEM SETTING

k-singles on the comments in different rating. k can be changed, we will see the performance on different number of k. k is probably from 1 to 3
k-combination problem
So there will be 5 category and classify the comment to one of it.
Using Hadoop.
We will be writing a program that will filter out the stop words.

Ignoring the empty comment or defalt value.

(1) Work 0: Pre-process : remove stop words (python)
(2) work 1: k-singles (support = ?) mapReduce (Hadoop) cut top 100
(3) work 2: prediction : weight equation (

Firstly, Put every comments into that rating category and will only left with top 100 FIS. The system intake the data set we will be using mapReduce of k-singles on comment and it's count. Mapper will take in the data result will be <(shingle),<count,rating>,...> Reducer will be <(shingle),<count,rating>,...> When a new comment comes in we will compare its k-shingles to the result from Preprocessing result.

2.1 Basic Statistic of the dataset

Before we explain the structure of our system, we will be looking at basic statistic of the data set we will be using.

Table 1: Basic Statistic of the rating

	Score
Mean	4.18
Std	1.31
Min	1
First Quartile	4
Median	5
Third Quartile	5
Max	5

Table 2: Basic Statistic of the rating

	Rating	Word(Processed)
Mean	4.18	255
Minimum	1	7
Maximum	5	14425

3 SYSTEM STRUCTURE

In this project, we will be using python to code the system that perform processing on the data set. There are three stages in the whole system, we will be explaining the idea and coding in the

Table 3: Original Data

Id , ProductId, UserId, ProfileName, HelpfullnessNumerator, HelpfullnessDenominator, Score, Time, Summary, Text
1,B001E4KFG0,A3SGXH7AUHU8GW,delmartian,1,1,5,1303862400,Good Quality Dog Food,I have bought...
2,B00813GRG4,A1D87F6ZCVE5NK,dll pa,0,0,1,1346976000,Not as Advertised,"Product arrived labeled...
3,B000LQOCH0,ABXLMWJIXXAIN,"Natalia Corres ""Natalia Corres""",1,1,4,1219017600,"""Delight"" says it all","This is a confection...
4,B000UA0QIQ,A395BORC6FGVXV,Karl,3,3,2,1307923200,Cough Medicine,If you are looking...

Table 4: Part of Result Dataset

Score	Text
5	bought vitality canned dog food products good quality product looks like stew processed meat smells ...
1	product arrived labeled jumbo salted peanutsthe peanuts actually small sized unsalted not sure error ...
4	confection centuriesit light pillowy citrus gelatin nutsin case filberts cut tiny squares liberally coated powdered ...
2	looking secret ingredient robitussin believe iti got addition root beer extract ordered good cherry sodathe ...

following subsections. Based on the result generated by each stage we can do prediction of rating based on the give comment.

3.1 Methodolgy

The prediction was based on k-shingle approach. The main steps were first to extract sample shingles from training set and then to predict the score by matching them. Shingle length of 1 to 5 were focused and this was considered according the average size of review comments. The further shingle length was not useful due to extremely low repeat rate.

Two approaches for the default score were compared: the middle score of score range and the average score of training set. Default score was required to assign the score if no shingle was matched from the sample shingle set.

Two prediction methods were implemented: k-shingle predicted score and Regression Method. The first method was calculated by matching shingles. However, each k-shingle prediction only focused on its own k-shingle information. We would like to explore an approach to gather all the information and test whether a more accurate result could be achieved.

A straight forward idea would be taking an average from the 5 k-shingle predicted scores; nevertheless, this deemed all the shingle scores were equally important. And if there were matched 5-shingles, the 5-shingle predicted score should be more representable and with higher weight.

Therefore, the Regression Method was introduced as an objective approach to investigate the linkage among the 5 k-shingle predicted scores.In this project, this method was executed by using linear regression.

3.2 Stage 1 - Data Preprocessing

In this subsection, we will be explaining the idea behind stage one, pre-processing. As only small portion of the given data are useful in this project, we need to do pre-processing before doing other applications.

During the pre-process, we will extract rating column, summary column and comment column of each review record. Moreover, we will remove all punctations and selected stopwords from comment part and left with only meaningful words. We need to preserve some stopwords because it effect the real meaning of the comment. For example, "Good" and "Not Good". If we remove stopword "Not", the meaning of the comment is completely reversed.

We will be looking at the algorithm for data preprocessing:

while *There exists next row inside Reviews.csv* **do**
  Extract **Rating** and **Text**;
  Lower **Text** and remove *some* stopwords and punctuation;
  Returning line with format $Rating, word_1, word_2, \ldots$;
end

Algorithm 1: Preprocessing

The original data file is in format of *.cvs*, each column is separated by a comma and each row is separated by a linebreak. Each row have the format : {Id , ProductId, UserId, ProfileName, HelpfullnessNumerator, HelpfullnessDenominator, Score, Time, Summary, Text}. As mentioned before, the only important parts are Score and Text. Thus, the resulting data have format : {Score, Text}. After the preprocessing is finished, we will split 70% of the dataset as training set and 30% as testing set.

Let us observe part of the original data and the result from preprocessing before we explain how we do pre-processing in Table 3 and Table 4.

3.3 Stage 2 - Construction of Shingle-Score Database

At first, the mass shingle-score database was constructed by MapReduce among the training set. The mapper generated 1-shingles to 5-shingles for each review. The key is {*Score*, *Shingle Length*, *Shingle*} and the value is "1". During the shuffle and sorting stage, same key value's tuples were gathered and the reducer summed all the 1's to count the frequency. In order to confine the data, shingles with count of 1 were excluded. This not only reduced the size of output, but also filtered out unnecessary shingles. For example, a shingle that appeared only once was not meaningful for prediction and it required a considerable size of storage to store all of them. In general, a frequency threshold could be set for this step.

Here is the algorithm for mapper and reducer:

```
while There exists next row inside Preprocessed.csv do
    for k ← 1 to 5 do
        Find all k-shingles;
        for Each shingle found do
            Return << rating, k, shingle >, 1 >;
        end
    end
end
```

Algorithm 2: Mapper

```
forall the tuples with key < rating, k, shingle > do
    frequency = sum of all tuple values;
    Return < rating, k, frequency, shingle >;
end
```

Algorithm 3: Reducer

Here is part of the result of the Map-Reduce procedure:

Score	k	frequency	shingle
3	4	23	food freshly openedpi likes
5	2	1712	very nice
2	1	1601	say
3	4	27	coffeetea love organic coffee

After the mass database was yielded from the training set, it was then be refined to a smaller set for prediction use. The refined database should be restrained to be sufficiently small in order to facilitate the speed of reading it. In this project, top n frequency for each shingle-score combination was chosen to build the prediction shingle set. Top frequency was considered as the higher the shingle frequency, the more representable of score of it. For example, a "top-100 database" would be 2500 rows long, as it extracted top 100 record from each 5 shingles and 5 different scores. Futhermore, "top-100 database" and "top-300 database" were used in this project.Other refinement method could also be considered, such as, top 10% of each shingle-score combinations.

3.4 Stage 3 - Prediction

3.4.1 k-Shingle Prediction Score. First, a review would be pre-processed with the same procedure on training data set, namely, remove stop-words. Next, 1-shingles to 5-shingles of it were constructed and then were searched through the refined database for matchup. All matched shingles were collected with the score and frequency of each. The k-shingle predicted score was based on this collection and was calculated by the weighted average.
For each k and t_i = k-shingles:

$$\text{Weighted Score} = \frac{\sum_i \sum_j score_j(t_i) \times freq_j(t_i)}{\sum_i \sum_j freq_j(t_i)}$$

where $i = i$-th k-shingles and j = Score range. The following was a simple example of 1-shingle score:

Table 5: Part of Result Dataset

Score	k	frequency	shingle
5	1	68	good
4	1	35	good
5	1	57	really

Input: *this is really good*
Score:

$$\frac{5 \times 68 + 4 \times 35 + 5 \times 57}{68 + 35 + 57} = 4.78125$$

Note that the same shingle, "good" in the example, could appear multiple times in the database but in different scores. The frequency respresented the weight of it in each score. And in the above, "good" was baised to score of 5. Repeated words were also considered. For example, "this is really really good" would be a higher score that the above example, as "really" would be considered twice. The reason for not focusing on distinct words was that repeated words or phrases from the reviewer represented the sentiment even more clearly.

3.4.2 Regression Method. Regression Method was applying linear regression on the five k-shingle scores. There were five regression coefficients and one intercept to be estimated from the training data. The k-shingle predicted scores of training data were first calculated and the system of equations was obtained as follows:

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \cdots + \beta_5 x_{15} \\ y_2 &= \beta_0 + \beta_1 x_{21} + \cdots + \beta_5 x_{25} \\ &\vdots \qquad\qquad \vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \cdots + \beta_5 x_{n5} \end{aligned}$$

In matrix form :

$$\mathbf{Y} = \boldsymbol{\beta_0} + \mathbf{X} \cdot \boldsymbol{\beta}$$

where $\mathbf{Y}$ is a $n \times 1$ matrix, $\boldsymbol{\beta_0}$ is a $n \times 1$ matrix, $\boldsymbol{\beta}$ is a 5×1 matrix, $\mathbf{X}$ is a $n \times 5$ matrix.
The coefficients were estimated by the following formula:
[beta = blah blah blah formula]
The computational complexity were examined as: $\mathbf{X}^T\mathbf{X}$ was in $O(n \times k^2)$; $(\mathbf{X}^T\mathbf{X})^{-1}$ was in $O(k^3)$; $\mathbf{X}^T\mathbf{Y}$ was in $O(n \times k)$; the final multiplication was in $O(k^2)$. Note that n was the number of rows of training data set and k was the number of coefficients. Moreover, n was much larger than k and hence it was the dominant term. As a result, the overall computation complexity was approximately in $O(n)$. As the time was linear with n, this computation was also scalable for large dataset. Finally, the formula of Regression Method is as following :

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_5 x_5$$

where x_i is the i-th shingle score, for $i = 1, \ldots, 5$. For prediction, the 5 k-shingle scores were calculated and then were input to the above formula.

4 PREDICTION RESULT

4.1 Shingle Percentage Performance

Table 6: Shingle Match Performance

Shingle Length	Match %
1	99.87%
2	67.88%
3	16.47%
4	2.19%
5	0.30%

4.2 Default Score Comparison

Table 7: Mean Squared Error Comparison

Prediction Method	Default Score	
	3	Train Mean = 4.18
Train Data Mean	1.717	1.717
1-shingle	1.596	1.596
2-shingle	2.160	1.595
3-shingle	2.918	1.667
4-shingle	3.089	1.704
5-shingle	3.112	1.710
Regression	1.449	1.369

4.3 Prediction Method Comparison

5 TESTING

The system will be taking 70% of the data set to be based set and the rest as testing and improving the prediction.
Testing result is as following:

6 CONCLUSION

7 FUTURE WORK

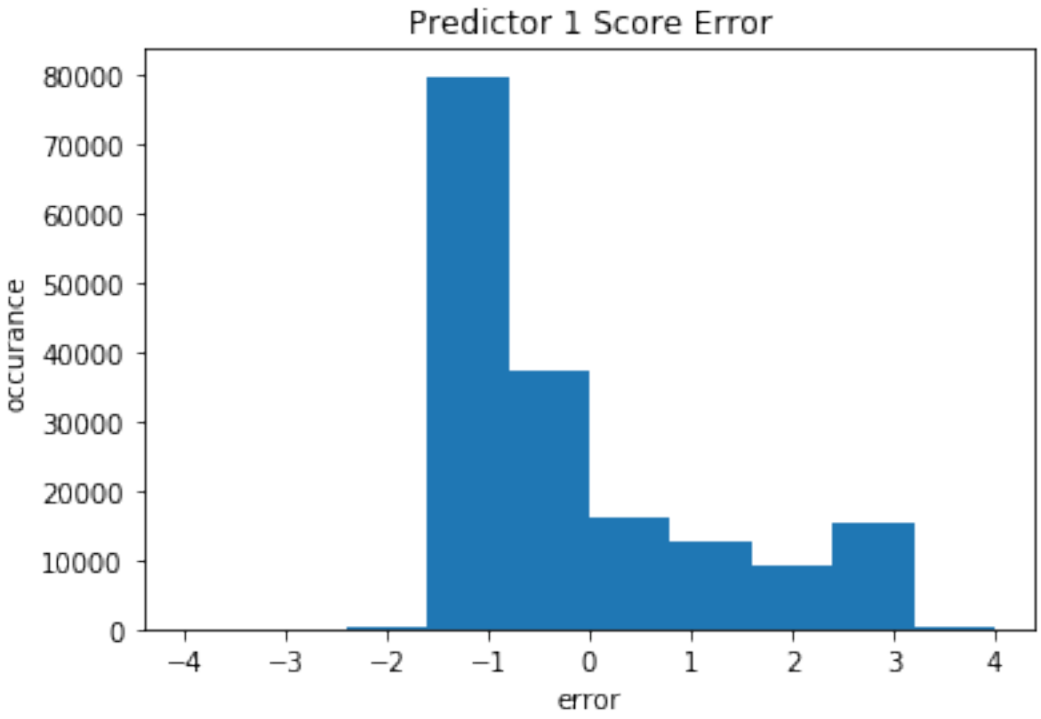

Figure 1: Predictor 1 with Default Score set to 3

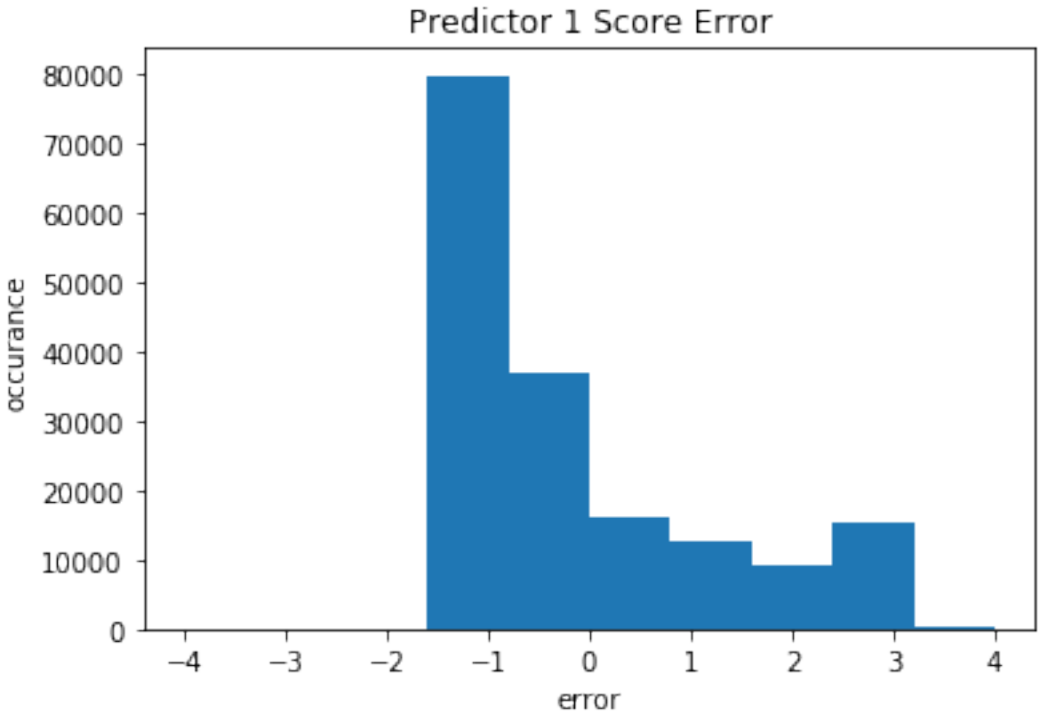

Figure 2: Predictor 1 with Default Score set to 4.18