AMAZON PRODUCT REVIEWS PROJECT REPORT

Ryan Erwin, Curtis Hughes, Joe King, and Nathan Ige

NOVEMBER 25, 2015
FDAC15
UTK

Summary

The objective of this project is to use Amazon product review information to find what causes a positive review. The project entails web scrapping, text mining, and utilizing predictive models to reach the objective of receiving a high rated review. By accomplishing this goal we can provide sellers the words or product features that boost the success and rating of their product.

Our methodology is to scrape the review data of the top 1,000 reviewers via python, AWS machines, and R. After this we clean and import the review data into R for analysis. In R, a word cluster analysis is created using the customer summary review, product description, and item title. This analysis pinpoints which words co-occur with other words and fall in a category of high frequency. Finally, from these words a random forest model is created to form a descriptive analysis to determine what words lead to a high Amazon review rating of 5 stars.

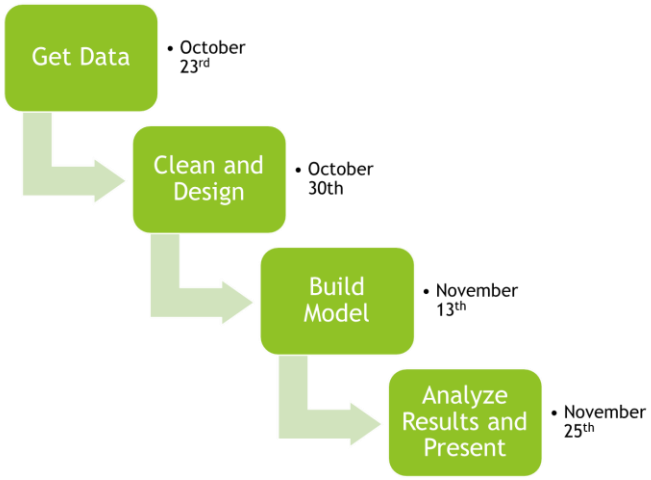
Our random forest model has a performance of 0.75 and a top decile lift of 1.6. Meaning that the model predicts 1.6 times better than randomly selecting words that impact a higher rating success. Therefore, our model is successful in creating a bag of words that conclude to a high rating of 5 stars. For future models and analysis, the model can be changed to also create what words or review feature lead to a bad review. This opens the door to determine what characteristics make a bad product and which make a great product.

Team Introduction

The team for this projected consisted of four members. Three provided help in the area of data retrieval, while one created the final analysis. The table below shows the description of tasks for each of the team members.

Team Member	Responsibilities
Ryan	• Project Manager tasks • Utilize R to pull in Data • AWS • Support Role
Curtis & Nate	• Utilize Python to pull data • Tasks that require python • Support Role
Joe	• Model Developer • Interpret Results • Support Role

To ensure every step of the project was completing in a timely fashion, the following time deadlines were outlined.



Methodology

This section will describe the methods used for review data retrieval and the usage of machine learning algorithms.

Data Retrieval

Python, R, and AWS machines were used to scrape and retrieve the amazon product review information. A list of 1,000 top reviewers was used as a starting point for selecting what reviews to choose from. From these reviewers, we took their first 10 pages of reviews.

Python

The Python portion of the data retrieval relied solely on html parsing. This was mainly due to the specific user information that we wanted to inspect. The actual data parsing utilized a Python package called *BeautifulSoup* that was used to grab html tags and their values. Using this package and Python's native tool-set we were able to retrieve a multitude of review information from the provided list of users.

AWS Machine

Here we will briefly discuss the role of several technologies outside of R that made this project possible, namely Amazon Web Services (AWS). To retrieve millions of reviews from Amazon's servers required hundreds of thousands of GET requests. If all of those GET requests came from one PC with the same IP address, then Amazon might think I have malicious intentions and temporarily ban my IP from retrieving additional data. To overcome this hurdle, I employed AWS and ran my web scraping script across more than twenty machines. This approach delivers on two fronts: it parallelizes the data collection, thus reducing human waiting time, and it mitigates the problem of sending too many GET requests from one IP.

Model Building

A cluster analysis was along with a random forest model to create a descriptive analysis for the data. To reduce data dimensions a product category (e.g. Toys and Games, Books, Appliances, etc) is chosen. From this a subset of reviews are chosen that are within this category. The reviews are then used in a cluster analysis. To create a cluster analysis we need to do the following.

1. Convert all text to lowercase, remove stop words and punctuation
2. Create a review by term matrix
3. Create an adjacency matrix or in other words a term by term matrix
4. Create a cluster network diagram

Steps 1-3 can be found on the repository located at https://github.com/fdac15/AmazonProductReviews under the file name "functions_for_Text_and_Networking_Mining.R". Below shows the code that produces a cluster network diagram. For this example the item category used is "Toys and Games".

```
data <- read.csv('alldata1.csv',header=TRUE,sep=",",stringsAsFactors=FALSE)
source("functions_for_Text_and_Network_Mining(1).R")

t <- table(data$itemCategory); t <- t[order(-t)];

IC <- "Toys & Games"

cat <- data[which(data$itemCategory == IC),]


cat$Summary <- sapply(cat$Summary,function(x) tolower(x))
cat$itemTitle <- sapply(cat$itemTitle,function(x) tolower(x))
cat$itemBrand <- sapply(cat$itemBrand,function(x) tolower(x))
cat$Description <- sapply(cat$Description, function(x) tolower(x))
cat$Description <- sapply(cat$Description, function(x) substr(x,1,200))


#STEP 1: create document by term matrix
sdoc_term_mat <- create_document_term_matrix(cat$Summary)
ddoc_term_mat <- create_document_term_matrix(cat$Description)
itemTitledoc_term_mat <- create_document_term_matrix(cat$itemTitle)
branddoc_term_mat <- create_document_term_matrix(cat$itemBrand)
#investigate output
#doc_term_mat
#inspect(doc_term_mat)

#STEP 2: create adjacency matrix
sadj_mat <- create_adjacency_matrix(sdoc_term_mat)
titleadj_mat <- create_adjacency_matrix(itemTitledoc_term_mat)
brand_adj_mat <- create_adjacency_matrix(branddoc_term_mat)
dadj_mat <- create_adjacency_matrix(ddoc_term_mat)
```

```
#object: output from the create_adjacency_matrix function
plot_network <- function(object){

      #create graph from adjacency matrix
      if (!require("igraph")) install.packages("igraph") ; library(igraph)
      g <- graph.adjacency(object$z, weighted=TRUE, mode ='undirected')
      g <- simplify(g)
      # set labels and degrees of vertices
      V(g)$label <- V(g)$name
      V(g)$degree <- igraph::degree(g)

      layout <- layout.auto(g)
      opar <- par()$mar; par(mar=rep(0, 4)) #Give the graph lots of room
      #adjust the widths of the edges and add distance measure labels
      #use 1 - binary (?dist) a proportion distance of two vectors
      #1 is perfect and 0 is no overlap (using 1 - binary)
      edge.weight <- 7  #a maximizing thickness constant
      z1 <- edge.weight*(1-dist(t(object$termbydocmat)[object$ind,], method="binary"))
      E(g)$width <- c(z1)[c(z1) != 0] #remove 0s: these won't have an edge
      clusters <- spinglass.community(g)
      cat("Clusters found: ", length(clusters$csize),"\n")
      cat("Modularity: ", clusters$modularity,"\n")
      plot(g, layout=layout, vertex.color=rainbow(4)[clusters$membership],
        vertex.frame.color=rainbow(4)[clusters$membership] )
}
```

It should be noted that the plot_network function was developed by David Hunter, the author of the network package. After this is created, a random forest model is built from the most frequent words and the words that co-occur with other words the most. Our response variable was a binary variable that represented if the review had a five star rating or not. The following table shows the predictors and the response.

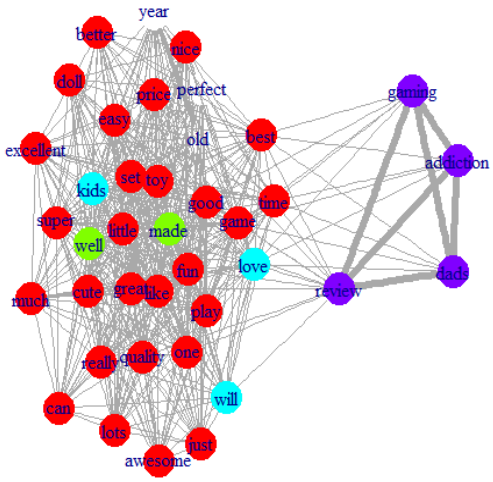
Predictors	Response
Recency of Review Frequency of Top Words from Review Summary Frequency of Top Words from Product Description Frequency of Top Words from Item Name Frequency of Top Words from Brand Name	0 – Not a 5 Star Review 1 – A 5 Star Review

Half of the data, by random selection was used to build the model. The second half is used as a test set to evaluate the model's performance. This is a way we can use a form of cross-validation to ensure our model is performing successfully.

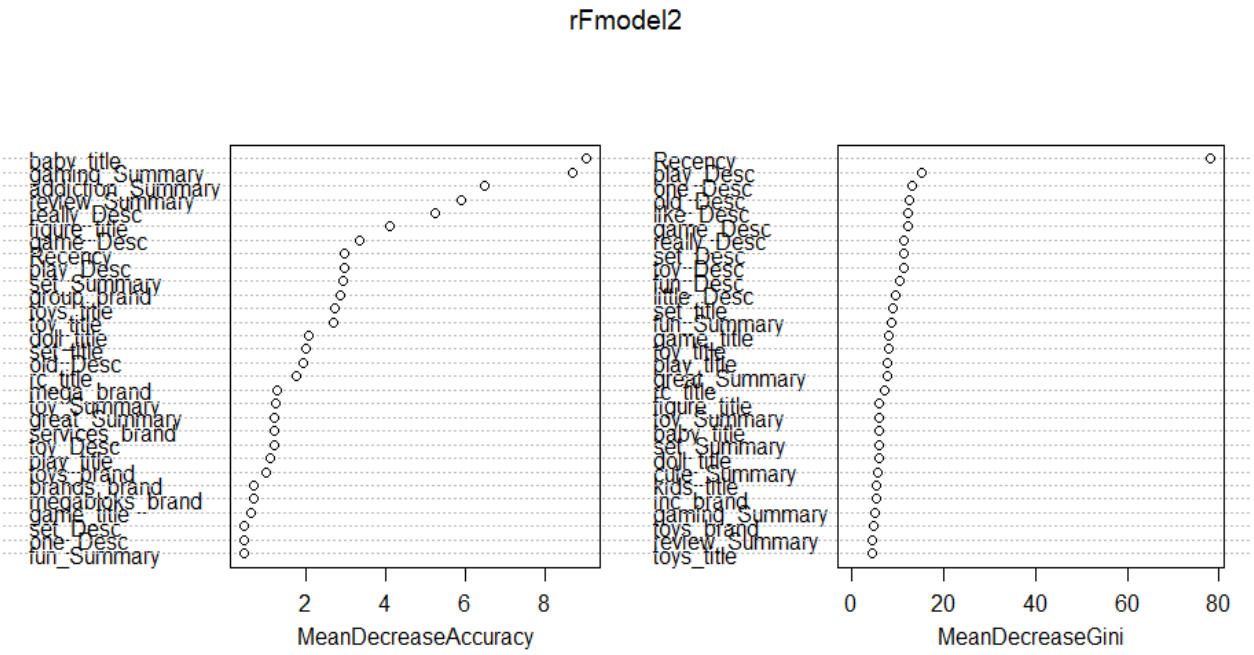
Analysis and Model Performance

We used the item category "Toys & Games" as an example for an analysis and model of the data.

As stated in our methodology, we create a cluster analysis to find the most frequent words. The following shows a cluster diagram of summary review words that are of high frequency and co-occur together. From the graph below, we can see that there are four clusters. The thicker the rod means the more time the two words/nodes occur together in the frequency of reviews.

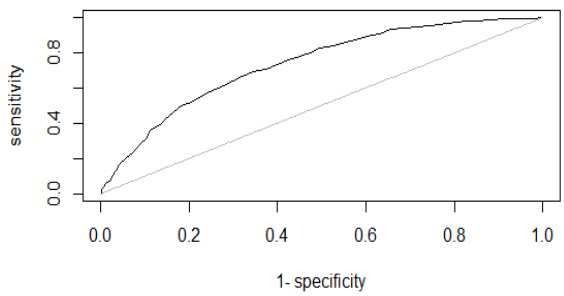


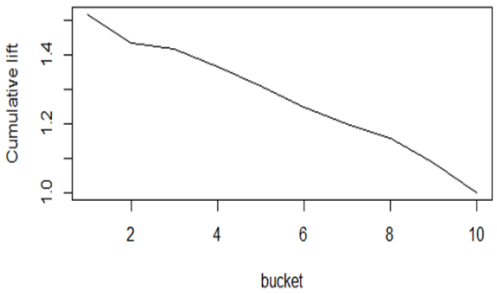
As we can see one cluster contains words gaming, addiction, review, and dads. These words are then counted in each review and made as a predictor in the random forest model. The following graphs depict which predictors have the largest impact on the likelihood of getting a 5 star rating review. From the plots we can see that the word baby in the title description has the highest impact on our response variable. These plots can show a seller what words are a game changer in the success of their project.



Model Performance

The following graphs shows AUC (area under the curve) and the cumulative lift curve of the model. From the AUC graph (left plot), we see that our model had a performance of about 0.75. This is a decent performance value. From this we can say it isn't bias nor is it overfitting the data. Our top decile lift is approximately 1.6. This means that our model is 1.6 times better than randomly choosing what words make a 5 star review.





Conclusion

By our model performance and results we conclude that we have successfully established a method to determine what bag of words occur in a positive rated review. This could help a seller pinpoint what words may be impacting the success of their product.