# Big Data Analytics Final Report – Book Recommendation Engine

Yunfan Zhang yz2866, Huashu Li hl2919, Zhuangfei Yang zy2233
IEOR Department
Columbia University
yz2866@columbia.edu, hl2919@columbia.edu, zy2233@columbia.edu

*Abstract*—In our project, we build a recommendation system based on Mahout and Spark realizing three kinds of recommendation. The first one is recommending new books to customers. This includes recommending books to the existing customers based on their preference and recommending books to new customers based on books' popularity. The second one is recommending friends to customers if they have a similar taste for books. The third one is recommending potential useful reviews to customers. Finally, we use System G and build a Flask web application to visualize our findings.

***Keywords-Recommendation engine; Amazon Book Reviews***

## I. INTRODUCTION

The motivation of our project is to study aspects that constitute a good online sales website for books. Based on Amazon book review dataset, we want to recommend books for customers according to customer ratings and reviews for the books. For new customers or customers showing too little preference on its profile, we recommend a book list based on book popularity. Besides book recommendation, we also want to build a reader community which connects customers with similar tastes. This community recommends similar customers to each customer for their convenience of communication and reference.

Another aspect to enhance customer experience on the online book sales system is to recommend helpful reviews for each book. To attain this goal, we explored several features from the review texts including TF-IDF, positive word frequency, length of the review and average word length in the review. Then, we built classification models to classify the reviews into the helpful and the unhelpful groups. In this way, we are able to predict the helpfulness of one review and recommend helpful reviews for each book.

In the end, we visualized the reader community using System G and built the website using Flask and HTML.

## II. RELATED WORKS

Incentivized reviews from Amazon customers after receiving free or discounted items were studied and the data shows that average rating boosts in this case and introduces bias [1]. Thus it is worth studying what kind of reviews are helpful for customers and what are not.

Mike Seddon has written a blog on Natural Language Processing with Apache Spark ML and Amazon Reviews [2]. He built a Spark pipeline to extract features from the review text data and then classified review ratings. We refer to his idea of processing text data in Spark with modifications and use the features to classified helpfulness of the ratings. We believe that predication of helpfulness has more business value than predication of ratings. This is because almost half of the review data do not have helpfulness score while almost all of them have ratings.

## III. SYSTEM OVERVIEW

We use Amazon Book Review data from http://jmcauley.ucsd.edu/data/amazon/ [3].The original dataset contains product reviews and metadata from Amazon, including 142.8 million

reviews spanning May 1996 - July 2014. It includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). We use "K-cores" (i.e., dense subsets). These data have been reduced to extract the k-core, such that each of the remaining users and items have k reviews each. The size of the data is more than 9 gigabytes.

Our flow chart is shown in Figure 1. We first did some data exploration. Then we used rating data to build book recommender and friend recommender. Meanwhile, we used natural language processing to process review data and used it to predict the helpfulness of the reviews. Finally, we visualized our result and made it as a web application.

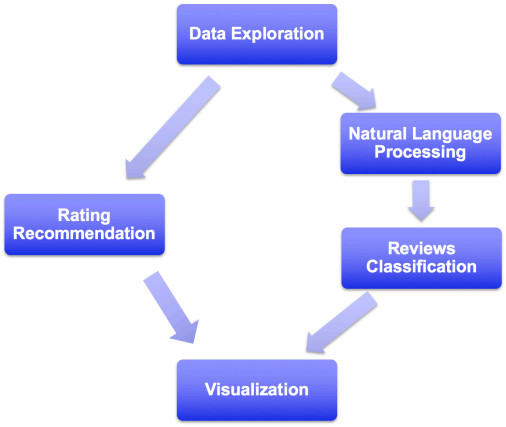


Figure 1. Flow chart of the system

## IV. ALGORITHM

To recommend books for each user, we used user-based recommender and item-based recommender in Mahout. For this dataset, the user-based recommender doesn't work well since the number of customers is far larger than the number of books. Also, many customers only show their preference for one book and it is hard to find neighbors for them with similar tastes. Thus, we mainly applied item-based recommender here. As one intermediate step in the algorithm, we defined the item similarity by Pearson correlation-based similarity matrix and Euclidean distance. In the evaluation stage to test how the different similarity matrices and the overall item-based recommender works, we divided the dataset into the training set which contains 90% of the data and test set which contains the other 10% data. Then we built the recommender on the training dataset and test the model on test set using average absolute difference between the predicted value and the response observed.

To recommend customers with similar tastes to each customer, we used the intermediate result from the user-based recommender and output the nearest 5 neighbors for each customer as the recommended friends.

We used tools provided in Python to capture features and classify the reviews. Nltk package was used to split the sentence into words, and WordCloud graph was used to give us a rough view of how frequently each word appears in the helpful review and unhelpful review.

Furthermore, we used the Spark pipeline for reviews classification (Naïve Bayes and logistic regression), which is shown in Figure 2.

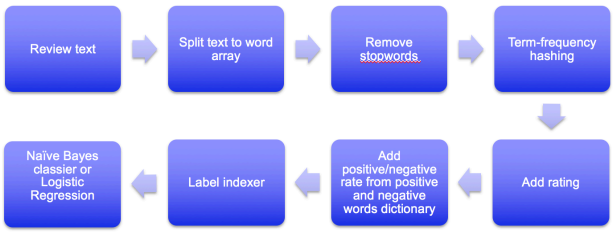


Figure 2. Spark machine learning pipeline for classification on reviews data

## V. SOFTWARE PACKAGE DESCRIPTION

We use HDFS, Mahout, Spark, System G, Python (Numpy, Pandas, Matplotlib, wordcloud, Flask) in our project.

- Apache Mahout collaborative filtering and recommendation

Mahout is an open source machine learning library from Apache. Recommendation is one of the three pillars of Apache Mahout's machine learning implementations (recommendation, clustering, classification). With these techniques, we can understand a person's tastes and find new, desirable content for them automatically [4]. Recommender can be implemented in simple java or combined with Apache Hadoop distributed computing framework.

We used Mahout in both Eclipse and command line combined with HDFS. Figure 3 showed a screen shot of Eclipse interface in our project.

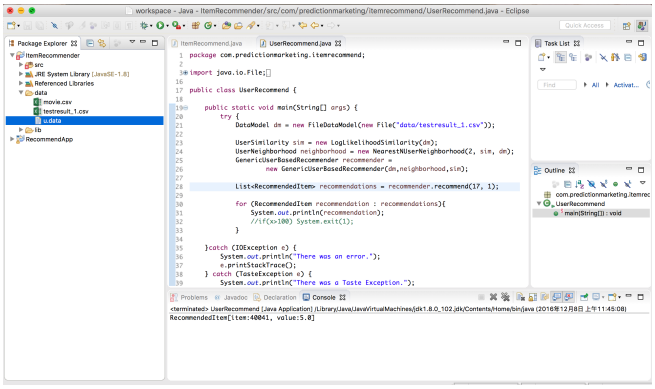

Figure 3. Eclipse interface using Mahout

- Python nltk package

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum [5].

In our project, we used nltk package to tokenize the reviews, and removed the stop words.

```
from nltk.corpus import stopwords
```

```
for val in pos["review"]:
    text = val.lower()
    text = text.translate(trantab)
    #find the words and punctuation in a string
    tokens = nltk.word_tokenize(text)
    #remove punctuations
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    for words in tokens:
        helpful_words = helpful_words + words + ' '
```

- Python wordcloud package:

An open source package in python to create word clouds in different shapes and various styles.


Figure 4. An example of alice mask word cloud created by python wordcloud pakage [6]

- PySpark for review classification.

The Spark Python API (PySpark) exposes the Spark programming model to Python.

Import findspark packages and packages from pyspark:

```
In [1]: import findspark
        findspark.init()
```

```
In [2]: #import pyspark
        from pyspark.sql import SparkSession
```

```
In [3]: spark = SparkSession.builder.appName('amazonRating').config("spark.some.config.option", "some-value").getOrCreate()
```

Load the data to PySpark Dataframe:

```
In [4]: #df = spark.read.csv('/Users/naauao/Downloads/amazonRating.csv')
        df = spark.read.json('/Users/naauao/Downloads/Books_5.json')
```

Build machine learning pipeline:

```
In [17]: # Configure an ML pipeline,
         regexTokenizer = RegexTokenizer(inputCol="text", outputCol="words")
         remover = StopWordsRemover(inputCol=regexTokenizer.getOutputCol(), outputCol="filtered")
         #ngram = NGram(inputCol=remover.getOutputCol(), outputCol="ngrams")
         hashingTF = HashingTF(inputCol=remover.getOutputCol(), outputCol="features")
         stringIndexer = StringIndexer(inputCol="label", outputCol="indexedLabel")
         nb = NaiveBayes(smoothing=1.0, modelType="multinomial", labelCol=stringIndexer.getOutputCol())
         #labelConverter = IndexToString(inputCol='prediction', outputCol='predictedLabel')
         pipeline = Pipeline(stages=[regexTokenizer, remover, hashingTF, stringIndexer, nb])
```

Train model and compute accuracy on the test set:

```
In [18]: # train the model
         model = pipeline.fit(training)
```

```
In [19]: # compute accuracy on the test set
         result = model.transform(test)
         predictionAndLabels = result.select("prediction", "indexedLabel")
         evaluator = MulticlassClassificationEvaluator(metricName="accuracy", labelCol="indexedLabel")
         print("Accuracy: " + str(evaluator.evaluate(predictionAndLabels)))
```

- System G

IBM System G is a comprehensive set of graph computing software system for Big Data portfolio. IBM System G is the first complete software stack for all aspects of Graph Computing. IBM System G also includes several derived Network Science Analytics tools based on the state-of-the-art researches.

We visualized customer clusters using System G, where we uploaded files with customers as nodes and neighborhood relationships as edges. Part of the undirected graph is shown in Figure 5.

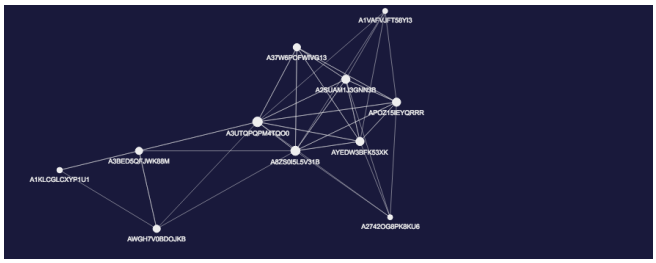


Figure 5. Cluster of depth 2 for customer APOZ15IEYQRRR using System G

According to degrees shown in System G, we can get the popularity of customers. Size of nodes in Figure 6 shows the degrees of different nodes, which means that nodes with higher degree represent customers with higher popularity. Thus we recommended these most popular customers for new customers or customers with no neighbors because these popular customers are more likely to be others' neighbors and have most common taste for books.

Figure 6. Part of a graph showing customers with different popularity in System G

- Flask

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frameworks related tools.

We created a web GUI based on Python Flask framework and free HTML template [7]. After running the python program and opening website in local host, customers can enter their IDs in the book recommendation page and get special recommendations after few seconds.

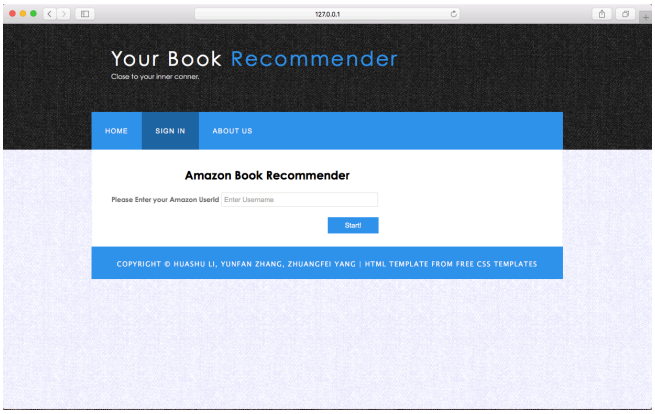


Figure 7. Web GUI recommendation sign-in page

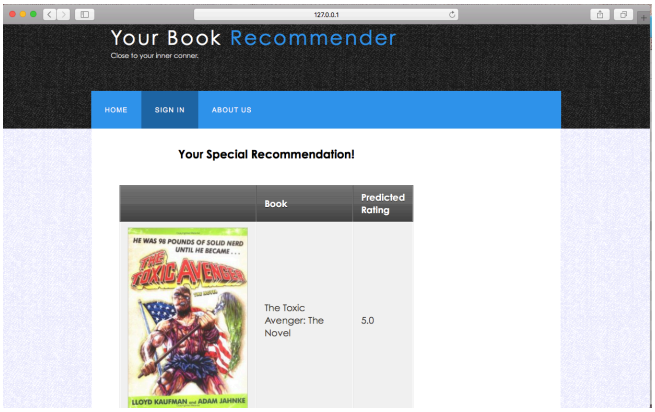


Figure 8. Web GUI result page: recommending books for the given user

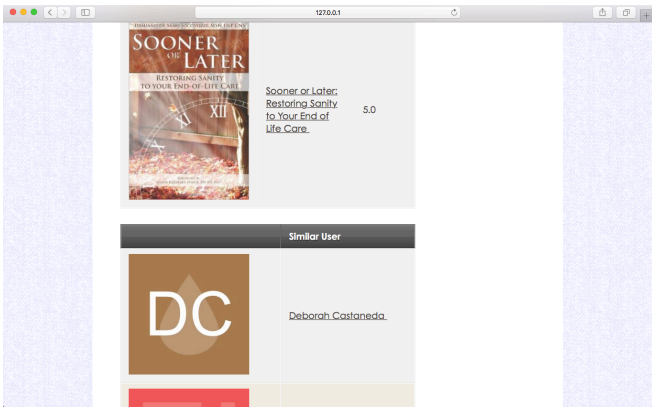


Figure 9. Web GUI result page: recommending users for the given user

## VI. Experiment Results

- Data Exploratory Analysis

The rating distribution shows that most customers give a rating of 5.0 out of 5.0, and most books receive an average rating higher than 4.0. Thus we can't recommend books to users simply based on the ratings.

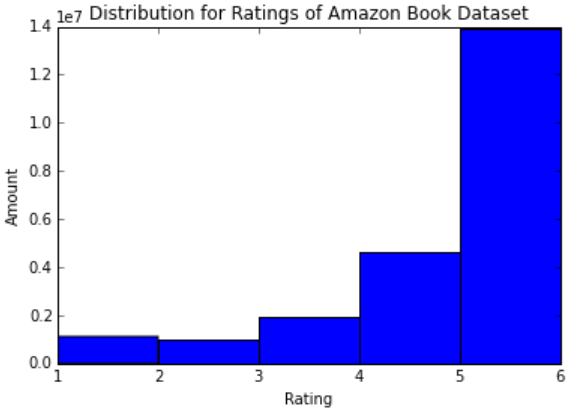


Figure 10. Rating distribution of Amazon book review dataset

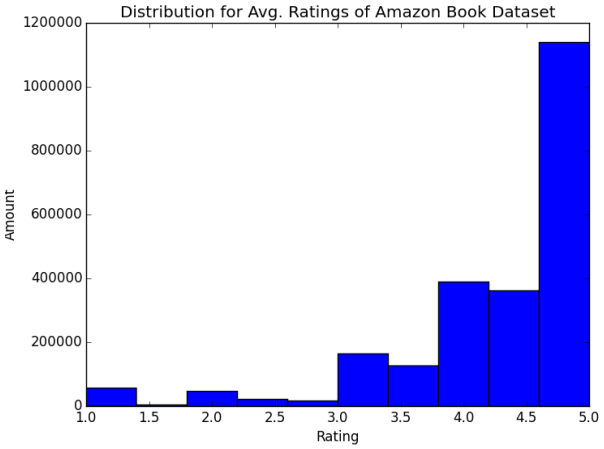


Figure 11. Distribution for average ratings for each book

- Recommendation in Apache Mahout

We used item-based recommender and compared the Pearson correlation-based similarity matrix and similarity based on Euclidean distance, the result showed that the average absolute difference calculated from the test set is smaller for Euclidean distance method (approximately 0.63) than that for Pearson correlation-based similarity (approximately 0.74). Thus we chose the Euclidean distance method in item-based recommender to generate a whole list of recommended books to customers, which recommends top 5 books for each customer. The results are shown in Figure 12-14.

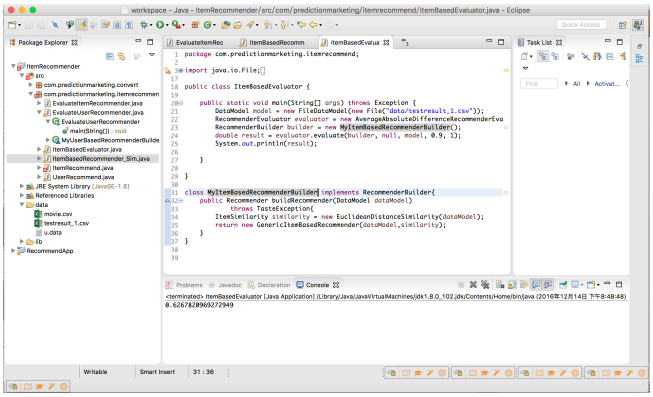

Figure 12. Evaluation of item-based recommender using Euclidean distance similarity matrix

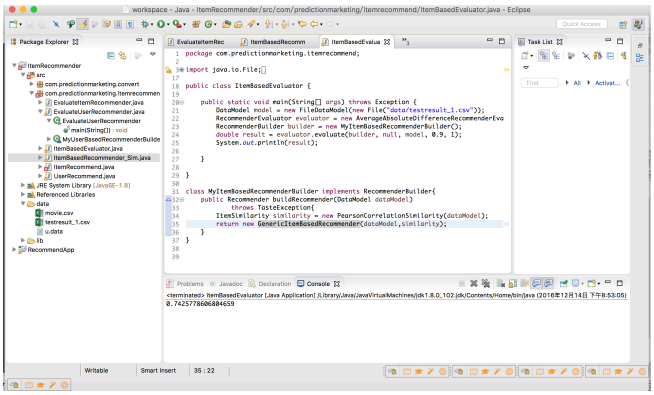

Figure 13. Evaluation of item-based recommender using Pearson correlation-based similarity matrix

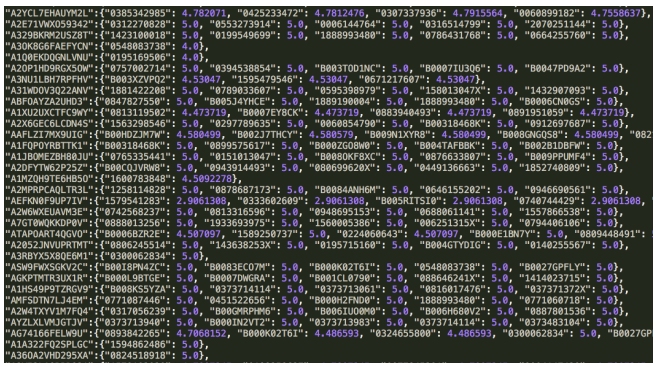

Figure 14. Top 5 book recommended for each customer

We also output the nearest 5 neighbors for each customer in Mahout.

Figure 15. Top 5 neighbors recommended for each customer

- Word Cloud for helpful and unhelpful reviews

After classifying the reviews into helpful category and unhelpful category, we drew the word cloud for each category after removing words that are frequent appeared in both helpful reviews and unhelpful reviews and are not critical in deciding the helpfulness of one review like ‘book’, ‘author’, ‘story’ and ‘people’.

The results are shown in Figure 16 and Figure 17.

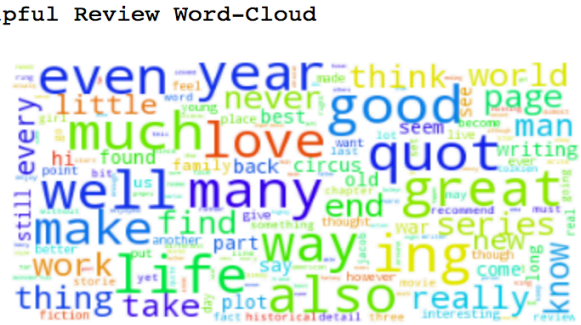


Figure 16. Helpful review word-cloud


Figure 17. Unhelpful review word-cloud

We discovered that helpful reviews tend to have more quotations from the book and other readers.

- Reviews Classification

Accuracy is 63.0% for Naïve Bayes and 65.0% for logistic regression. We also compute the corresponding confusion matrixes shown in Figure 18 and Figure 19.

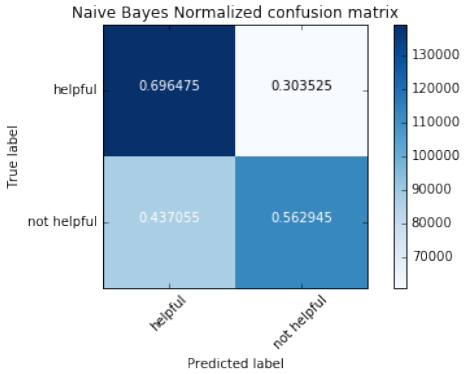


Figure 18. Normalized confusion matrix for Naïve Bayes

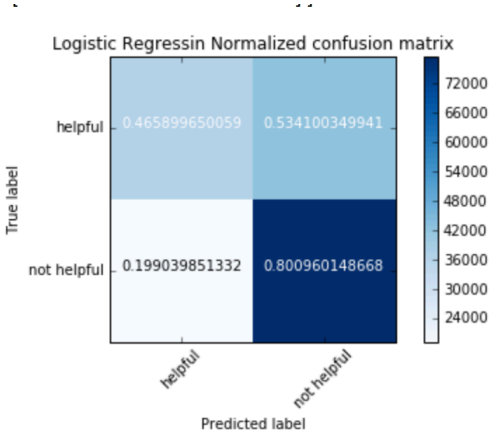


Figure 19. Normalized confusion matrix for logistic regression

- Web Application

We built a web GUI in local which can recommend books and similar customers according to customers' tastes for books. Also, it recommends most popular books and customers for new customers in the community. After clicking name of books or customers, the browser will redirect to Amazon product/profile page.

## VII. Conclusion

According our exploratory results, we find it necessary to recommend books catering to customers' tastes for books. Also, we hope to build a community for customers to communicate or follow each other. We conclude our project into 3 parts:

- Recommend books for customers

Because the number of customers is far larger than number of books, we recommend books via Mahout item-based recommender. If the customer is new or there is no recommendation in the recommender for him/her, the engine will randomly recommend 5 most popular books.

- Recommend customers for customers

We recommend customers that may have similar tastes for books according to neighbors in Mahout user-based recommender in Eclipse. If the customer is new or there is no neighbor in recommender for him/her, the engine will randomly recommend 5 most popular customers as what is shown in the graph of System G.

- Recommend reviews

Through Word Cloud we find that useful reviews tend to quote more from original books and other readers. Also we classified reviews in Spark using Naïve Bayes and Logistic Regression. The accuracy can be improved in the future because we can improve it by feature engineering and hyper-parameter optimization.

For the next step, we hope we can add reviews for each book what we thought might be helpful to our web GUI according to our classification methods. In addition, we hope to improve classification's accuracy by feature engineering. Last but not least, we can enhance our engine's speed by building a real time database to store books, reviews and customer information so that we do not need to scrape it online.

The contribution of each team member is 33%.

## Acknowledgment

We want to thank our media topic CA Chuwen Xu who gave us a hint on how to explain user-based recommender's bad performance in our case.

We also would like to thank professor and all TAs for the amazing course this semester. Without their help, we could not build a solid understanding of Big Data and handle complicated coding.

## REFERENCES

[1] Analysis of 7 million Amazon reviews: customers who receive free or discounted item much more likely to write positive review. Retrieved from https://reviewmeta.com/blog/analysis-of-7-million-amazon-reviews-customers-who-receive-free-or-discounted-item-much-more-likely-to-write-positive-review/.

[2] M. Seddon, (2015, Dec 5). Natural Language Processing with Apache Spark Ml and Amazon Reviews (Part-1) [Web log post]. Retrieved from http://mike.seddon.ca/natural-language-processing-with-apache-spark-ml-and-amazon-reviews-part-1.

[3] J. McAuley, (2016) Amazon Product Data [Dataset post]. Retrieved from http://jmcauley.ucsd.edu/data/amazon/.

[4] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Shelter Island, NY: Manning Publications, 2012. Print, pp. 11.

[5] Nltk 3.0 documentation. Retrived from http://www.nltk.org/.

[6] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[7] Free HTML template downloaded from http://www.html5webtemplates.co.uk/.