AMAZON SPAM REVIEW DETECTION

CS 6350-OU1 Big Data Management and Analytics

8/7/2017

Kartheek Kopparapu [kxk060100]

Prachi Chauhan [pxc163630]

Sriraam Ramakrishnan [sxr163730]

Teja Krishna Talluri [tkt160230]

Abstract

This project takes the ‘Digital Music’ dataset from a Stanford University archive [1], ‘Amazon product data’- contains product reviews and metadata from Amazon including 142.8m reviews spanning May 1996- July 2014. The data was sampled and about 20,000 reviews were analysed to conclude the hypotheses – “A review can be classified better using just review text rather than using review text along with other features”. The legitimate reviews are identified from fake ones using text analytics and implementing machine learning classification algorithms to conclude that review text is the deterministic feature in classifying the reviews. Any other features that were considered either reduce the performance or have no significant impact on the model.

Introduction

Customer reviews have become an essential part of e-marketing in the past 2 decades. Efficient analysis of customer reviews is becoming a market strategy and hence the distinction of a real/customer-written review from a fake/deceptive review becomes important [2].

A fake review is often observed to possess certain characteristics which differentiates it from a real customer’s review by containing more than 150 words, the ratio of Capitalized letter to lower case letters per word is significant, a user posting more than 7 reviews per day is likely to be an impostor, a user posting a review having a full product name with a lengthy description and product ID is likely to be an impostor trying to defame the product, an incentivized review is most likely to contain one of the words- unbiased, discounted, honest, bias, etc. These characteristics were analysed to develop a feature set for our data analyses.

All code used to produce the result was written in Scala on spark using the Databricks cluster.

Data Description

We are using the ‘Digital Music’ dataset which consists of 64,706 reviews. Since the Databricks cluster provides us with a maximum cluster capacity of 6GB, we chose to sample the data using ‘Random Split’ to extract a subset having 25% data, i.e. 16,177 reviews, for our analyses.

The default feature set accompanying this data is:

- productid - pertaining to a product
- title - title of the product
- price - price of the product
- userId - Id of the user
- profileName - name of the user
- helpfulness - fraction who found the review helpful
- score - rating of the product
- time - time of review (unix time)
- summary - review summary
- text - text of the review

Since this dataset is unlabeled and Spark only has a single unsupervised classifier, I.e. k-means, we needed to label our data to perform supervised learning. We read through some related work done on spam detection/opinion mining and created a deception model to label the data. As for the ‘Deception Data Model’ we used data that was provided as part of a similar experiment done to detect deception in reviews. Using the model, we labelled the sampled data so that we can create our model and perform classification on Amazon review data [3][4].

Features

After some research, we came up with some custom features for our dataset which were used for the training and testing purposes.

- **Helpfulness** - an estimate of the helpfulness of the review by a comparison between the number of users who found the review useful and those who did not, giving an integral count.
- **Ratings** - a product is rated on a scale of 1 to 5.
- **Capital letter ratio** - ratio of the capital letters in the review to the total number of words in the review, taking the assumption that symbols/punctuations in the original text are considered as capital letters.
- **Review text word count** - Count of the number of total words in the review text.
- **Product-star ratio (for x-star)** - ratio of the number of x-star a product has received to the total number of reviews of the product.
- **User-star ratio (for x- star)** - ratio of the number of x-star a user has given to the total number of reviews by the user.

  Formula used to calculate Product/user-star ratio: $\frac{(number\ of\ x{-}stars)^2}{number\ of\ reviews \times weight\ of\ the\ star}$

  Weight for each of the rating: 20(1-star), 40(2-star), 60(3-star), 80(4-star), 100(5-star)
- **User review rate** - number of reviews recorded per user per day over the period that he was active on amazon.
- **Review Text** - text input by the user for an item that he has reviewed.

Data Cleaning

- **Tokenization** - the 'reviewText' is passed through a tokenizer to convert the text into tokens.
- **Stop-word removal** - using the Stanford core NLP library, the stop words (those with a very high frequency in the unprocessed review text) were removed.
- **Punctuation removal** - the punctuation symbols are removed from the review text.
- **Lemmatization** - the processed text is then normalized through lemmatization.
- **Lowercasing** - the text is finally converted to lowercase as the final step of data cleaning.
- **HashingTF and IDF** - Term frequency-inverse document frequency (TF-IDF) is a feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus. We set the feature to 5000. [source: wikipedia]

Classification

Spam detection was carried out in two phases on two datasets created using the same raw dataset.

1. 'textOnly' dataset containing only the review text.
2. 'textFeatures' dataset containing the review text along with other custom features.

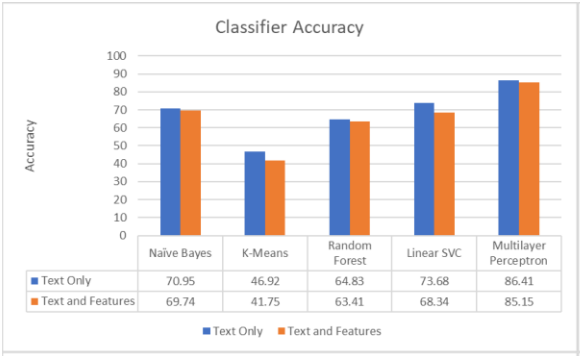
Each of the dataset was split into training and test data sets in 60-40 ratio using random split.

Phase I

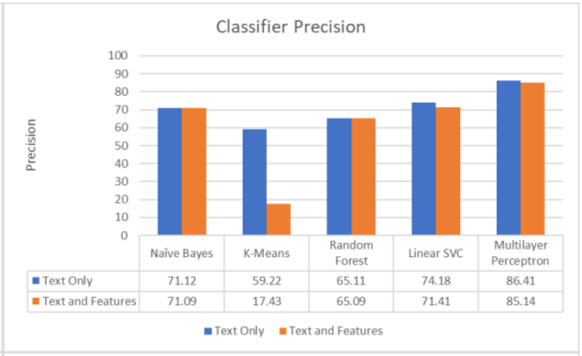
Five classifiers from spark.ml library and Multiclass Evaluators were used to evaluate each classification model.

1. **Naïve Bayes**: An object of the NaïveBayes class was created and 'multinomial' model was selected for the classifier. This object was then set as a stage in a pipeline. The pipeline was then used on bother 'textOnly' and 'textFeatures' training datasets. The resulting model was used on the test dataset of each respectively. The evaluator was used to derive the metrics: accuracy, precision, recall, and f-score.
2. **K-Means**: An object of KMeans class was created and 'k=2' and 'max-iterations=100'. The same process as Naïve Bayes model was applied to this model. Same evaluation metrics were used to evaluate the model.
3. **Linear SVC**: An object of LinearSVC class was created and 'max-iterations=10', 'Regparam=0.5'. We used Spark 3.1 cluster for this since LinearSVC is only available in Spark 2.2 and the only cluster on Databricks that had Spark 2.2 was Spark 3.0+ clusters. The same process as Naïve Bayes model was applied to this model. Same evaluation metrics were used to evaluate the model.
4. **Random Forest**: An object of RandomForestClassifier class was created and 'Impurity=gini', 'setFeatureSubsetStrategy=auto', 'Number of trees=3', 'max-depth=4', 'max-bins=32' were passed as parameters to it. The same process as Naïve Bayes model was applied to this model. Same evaluation metrics were used to evaluate the model.
5. **Multilayer Perceptron Model**: An object of MultiLayerPerceptronClassifier class was created and 'input layers=5000' (for 5000 features), 'output layers=2', 'Block size=128MB', 'seed=12345L', and 'max-iterations=400' were passed as parameters to it. The same process as Naïve Bayes model was applied to this model. Same evaluation metrics were used to evaluate the model. It was observed that adding a small number of hidden-layers had no significant effect on the accuracy and a large number of hidden-layers were not tested as the cluster resources were not sufficient.

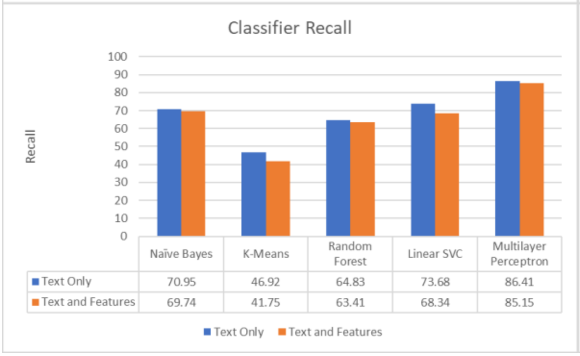
Results



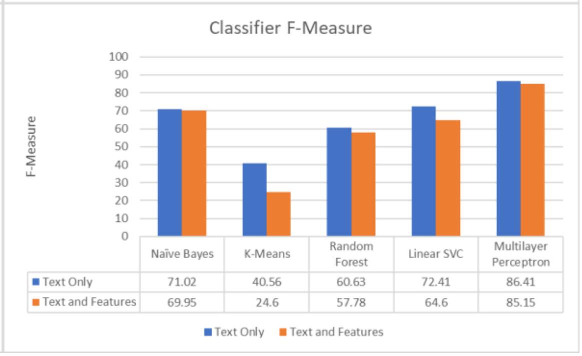
	Naïve Bayes	K-Means	Random Forest	Linear SVC	Multilayer Perceptron
🟦 Text Only	70.95	46.92	64.83	73.68	86.41
🟧 Text and Features	69.74	41.75	63.41	68.34	85.15



	Naïve Bayes	K-Means	Random Forest	Linear SVC	Multilayer Perceptron
🟦 Text Only	71.12	59.22	65.11	74.18	86.41
🟧 Text and Features	71.09	17.43	65.09	71.41	85.14



	Naïve Bayes	K-Means	Random Forest	Linear SVC	Multilayer Perceptron
🟦 Text Only	70.95	46.92	64.83	73.68	86.41
🟧 Text and Features	69.74	41.75	63.41	68.34	85.15



	Naïve Bayes	K-Means	Random Forest	Linear SVC	Multilayer Perceptron
🟦 Text Only	71.02	40.56	60.63	72.41	86.41
🟧 Text and Features	69.95	24.6	57.78	64.6	85.15

Based on the results Multilayer Perceptron Classification Model can be considered the best model taking all the four metrics collected for each model. We can also make some deductions on why some of the classifiers performed poorly as well. For example, K-Means model has the lowest accuracy because the initial data was not labelled and curated for the specific purpose of spam review detection. The low quality of the data caused the K-Means model to perform very poorly. As for the Random Forest Model, the quality of the data and the lack of unique features set caused the precision to be very poor. Naïve Bayes LinearSVC, and Multilayer Perceptron Classification Model usually perform much better in text classification and since review text provides most of the feature of the dataset, it is acceptable that they both performed well. Based on these observations we have decided to remove LinearSVC and Multilayer Perceptron Classification from phase II as improving their performance would require higher quality data and a larger cluster to get the results.

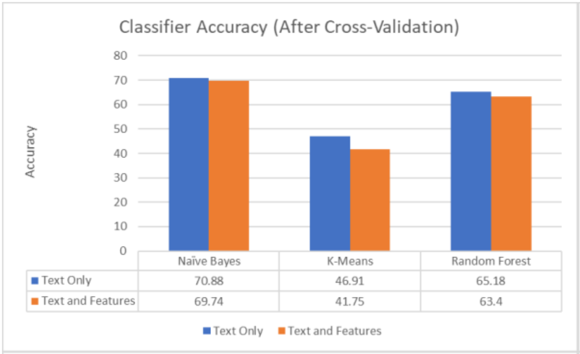
Phase II

To add a higher level of tuning, we incorporated Cross-Validation and Parameter Grid to our classification models. We decided to tune the low performers of phase I - KMeans, Naïve Bayes and Random Forest. A 7-fold cross validation was performed to sample the data.

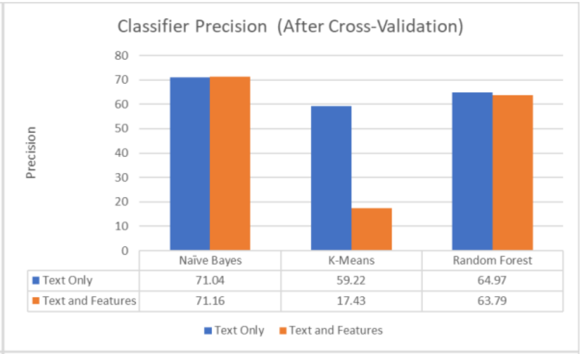
Parameter Grid values for each classifier:

1. Naïve Bayes: Smoothing Factors [0.0, 0.5, 1.0]
2. K-Means: Max Iterations [100, 400, 500]
3. Random Forest: Max Depth and Max Tress [10, 12, 15]

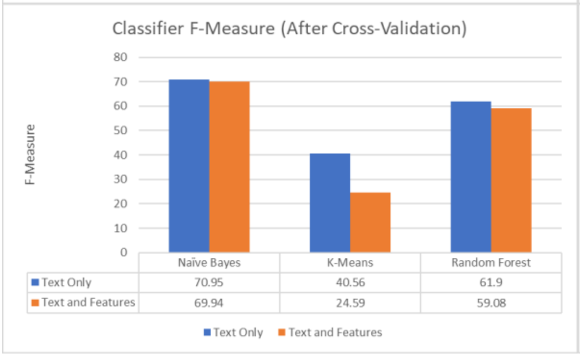
Results



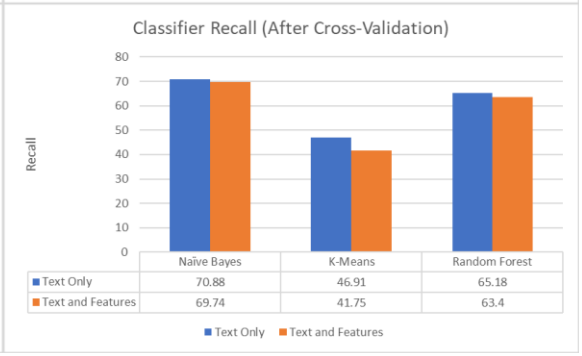
	Naïve Bayes	K-Means	Random Forest
■ Text Only	70.88	46.91	65.18
■ Text and Features	69.74	41.75	63.4



	Naïve Bayes	K-Means	Random Forest
■ Text Only	71.04	59.22	64.97
■ Text and Features	71.16	17.43	63.79



	Naïve Bayes	K-Means	Random Forest
■ Text Only	70.95	40.56	61.9
■ Text and Features	69.94	24.59	59.08



	Naïve Bayes	K-Means	Random Forest
■ Text Only	70.88	46.91	65.18
■ Text and Features	69.74	41.75	63.4

Conclusion

Based on the results of Phase I and Phase II we can make several observations. One is that Multilayer Perceptron model performed the best of all classification models used in the project. Naïve Bayes and LinearSVC were the next best performing models. And Random Forest and K-Means were the worst performing models. It was expected that Multilayer Perceptron, Naïve Bayes, and Linear SVC would perform well, since they work well with text based classification like reviews. Random Forest performed better than expected but the quality of data and lack of unique/strong features significantly affected the performance. K-Means classification model similarly performed poorly, especially on ‘textFeature’ dataset – it had low precision, due to similar factors that affected the performance of Random Forest.

The cause of the poor performance after cross validation and parameter grid could be due to overfitting or underfitting by over tuning data or not having selected the best parameters and high enough folds. But the cluster restrictions and the data quality forced us to make compromises on these variables.

We observed that the accuracy metrics show lower values for ‘textFeatures’ dataset than those for ‘textOnly’ dataset. So, we choose not to reject the hypotheses and conclude that “A review can be classified better using just review text rather than using review text along with other features”. Based on the collected metrics, the Multilayer Perceptron Model was the best classifier for both our datasets with an accuracy of 86.41 for ‘textOnly’ dataset and 85.15 for ‘textFeatures’ dataset. Not only is the accuracy high but all the other metrics are also within 83-86 range. This shows that the accuracy is dependable and the model is very consistent in properly predicting the dataset labels based on the given train and test features.

Further improvement can be made in all the classification models with better tuning, a larger dataset, and higher quality dataset/features.

References

The links below helped us understand the concepts and strategies involved in spam review detection and how to go about an unsupervised/un-labeled dataset:

1. McAuley, Julian. "Web Data: Amazon Reviews." SNAP: Web Data: Amazon Reviews. Stanford University, Fall 2016. Web. 06 Aug. 2017: The Dataset Archive
2. J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.
3. M. Ott, Y. Choi, C. Cardie, and J.T. Hancock. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.
4. M. Ott, C. Cardie, and J.T. Hancock. 2013. Negative Deceptive Opinion Spam. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.
5. Ma, Yingying, and Fengjun Li. "Detecting Review Spam: Challenges and Opportunities."Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (2012): n. pag. Web.
6. Liu, Bing. "Opinion Spam Detection: Detecting Fake Reviews and Reviewers." Opinion Spam Detection: Detect Fake Reviews and Reviewers. Bing Liu, 2008. Web. 06 Aug. 2017.
7. WikiHow. "How to Spot a Fake Review on Amazon." WikiHow. WikiHow, 06 Aug. 2017. Web. 06 Aug. 2017.