**Random Forest Text Classification Model for Ecommerce products**

*Biva Sherchan*

*MKTG 6640 - Text Analytics*

*Project*

*7/26/2023*

**Introduction**

Natural language processing (NLP) plays a significant role in e-commerce tasks, including product search, text classification, recommender systems, product question answering, machine translation, sentiment analysis, product description and review summarization, and customer review processing. NLP techniques have been applied extensively to solve modern e- commerce challenges (Malmasi et al., 2020; Zhao et al., 2020). One major NLP challenge in e-commerce is text classification which refers to classifying a product based on textual information, typically the product title or description, into one of numerous categories in the product category taxonomy tree of online stores. Although significant progress has been made in the area of text classification, many standard open-source data sets have limited numbers of classes which are not representative of data in industry where there can be hundreds or even thousands of classes (Li and Roth, 2002; Pang and Lee, 2004; Socher et al., 2013). Hand-coded rules for product classification can be applied but such rules are expensive to build and maintain. To cope with the large number of products and the complexity of the category taxonomy, an automated text classification system is needed, and its prediction quality needs to be high enough to provide positive shopping experiences for customers and consequently drive sales.

In this project my goal is to create a machine learning algorithm, in particular a text classification model that will be able to accurately classify a product into one the existing labels based on the product description. I will test out two models in this project - Naïve Bayes and Random Forest Classifier and will select the one with the highest accuracy rate. The problem is considered for an e-commerce domain and the dataset I used to train our models contains the descriptions of the products and their labeled categories. Using the text classification model, the companies can reduce the time and effort they spend on the manual tagging of product categories along with the manual errors.

**Data Source**

The dataset I used for this project is a classification-based E-commerce text dataset for 4 categories - "Electronics", "Household", "Books" and "Clothing & Accessories", which almost cover 80% of any E-commerce website. The dataset is in ".csv" format with two columns. The first column is the class name i.e., the label of the product description. The second one is the datapoint of that class i.e., the product and description from the e-commerce website.

The dataset has the following features:

Data Set Characteristics: Multivariate

Number of Instances: 50425

Number of labels: 4

Attribute Characteristics: Real

Number of Attributes: 1

Associated Tasks: Classification

**Methodology**

In this e-commerce dataset our target variable is the “Label” that has 4 categories - "Electronics", "Household", "Books" and "Clothing & Accessories". We only have one independent variable which is the “Description” of the product. I started with analyzing the dataset to check if any data cleanup was required. The dataset had one record of description text missing so that record was removed. To check if we have an imbalanced data, I calculated the proportions of the 4 product labels (Fig 1). Imbalanced data can cause issues in the model training and evaluation process. Since most machine learning algorithms are designed to optimize accuracy, they will often predict the majority class for most of the observations, resulting in poor performance for the minority class. This is especially problematic when the minority class is the one, we are interested in predicting accurately. From the label proportions (Fig 1), we see that the label “Household” has the most data (38%) making the dataset somewhat imbalanced.

I trained two different models Naïve Bayes and Random Forest Classifier for this classification task. The models were trained based on the product description text, along with the product labels to build a predictive model. The first step to build the models was to create a corpus from the description texts. A corpus is a large and structured set of texts. The next step was to preprocess the corpus using NLP concepts like tokenization, stemming, converting to lower case, removing stop words, punctuations, numbers, symbols etc. This reduces the complexity and dimension of the data and thus leads to less overfitted models. The corpus was then converted to document feature matrix so it can be used for modeling. To train the models, we split the dataset into training set (80%) and test set (20%). The training set was used to train the models and the test set was used to evaluate the performance of the models.

I chose Random Forest as my final model since it gave a much better accuracy rate of 97.4% vs Naïve Bayes Classifier with accuracy rate of 94.8%. Random Forest uses multiple decision trees and combines their output to improve the model. Random Forest uses bagging (BootStrap aggregating) algorithm to generate random samples which allows each individual tree to randomly sample from the dataset with replacement, resulting in different trees.

To train the Random Forest model I used R package irlba which works best with a larger dataset. For cross-validation, I created 10 random stratified samples, 5-folds repeated 2 times. The Stratified K-fold cross validation is better for an imbalanced dataset because it preserves the original class distribution in each fold and hence provides a more robust estimate of the model’s performance on unseen data. I used a default of 100 trees and a tune length of 5 which tells the algorithm to try 5 different configurations for hyperparameter tuning and select the best parameters as the final model. The parameter value used for the final model was mtry=76. The training of the model took 10 hours, but it gave a good accuracy rate of 97.4%. The out-of-bag (OOB) error rate was 2.6%. This means for the test observations; the model misclassifies the product label 2.6% of the time. OOB error rate is the prediction error on OOB samples which are the document rows that were left out from sampling due to the replacement sampling, and hence provides an unbiased estimate of error. The accuracy rate on the test holdout data was 97.3% almost the same as the accuracy rate on the train data of 97.4%, which provides the evidence that there is no overfitting of the Random Forest model.

Fig 1: Proportion of each product Label

A screenshot of a computer

Description automatically generated

**Results:**

The Random Forest model achieved a high accuracy rate of 97.4% which is pretty good and indicates that the model can effectively classify the product into its most appropriate label based on the product description.

**Conclusion:**

In conclusion, the Random Forest model I created mostly classified the products into correct labels based on the description provided. Large e-commerce companies can use such model to classify the products into the correct labels which can be a very manual task and very challenging with the large number of products and the complexity of the category taxonomy. The model prediction is quite high so the customers can have a positive shopping experiences which will consequently drive sales.

To further improve the model accuracy, we could run more cross validation samples, select different dimensions for LSA computation, select different number of decision trees, or try Linear SVM model.

**Data:**

https://www.kaggle.com/datasets/saurabhshahane/ecommerce-text-classification

**Reference:**

Shervin Malmasi, Surya Kallumadi, Nicola Ueffing, Oleg Rokhlenko, Eugene Agichtein, and Ido Guy, editors. 2020. Proceedings of the 3rd Workshop on e-Commerce and NLP. Association for Computational Linguistics, Seattle, WA, USA.

Xin Li and Dan Roth. 2002. Learning question classifiers. In COLING 2002: The 19th International Conference on Computational Linguistics.