Reviews Analysis And Recommendation System For E-commerce Retails

Yinuo Chen, Yuwei Li, Simou Ma, Yating Zhou

Abstract

This paper uses E-commerce data sets with customers' characteristics and customer reviews to build NLP models to identify customers' attitudes toward specific products and predict customers' rating score based on natural language processing(NLP) techniques and machine learning models, while at the same time to build a recommendation system based on item-based collaborative filtering(IB-CF) to help startups or small retailers in the E-commerce platform.

Keywords: Natural Language Processing(NLP), Item-Based Collaborative Filtering(IB-CF), Recommendation System, Sentiment Analysis, Machine Learning

1 Introduction

With the thrive of E-commerce and the on-going situation of this pandemic, people nowadays are more likely to shop online and leave comments on products they've bought. Dealing with these comments and extract useful information from these comments requires the natural language processing(NLP) method. According to our research, small business platforms or Brand owned websites or even startups usually don't have a budget to achieve this. Our goal is to use multiple E-Commerce datasets especially in the Women's Clothing area with reviews written by customers to construct NLP models using bag-of-words, TF-IDF as the word vector representation method, and SVC, Random Forest, LR, and Multinomial Naive Bayes Classifier as the regression methods. Our second stage is a recommender system based on sentiment analysis results to provide our customers with top-rating products that are related to his or her current choices. Finally, we use another E-commerce dataset to test our recommender system model.

2 Background

The E-commerce business industry is exploding. Brick and mortar stores that temporarily closed may never open again. Shopping online used to be a convenience and a luxury, now – it's a necessity. E-commerce has been growing for a while, and will be sure to expand rapidly in a near future. Newly-established E-commerce startups have never been so much more, and many of them are struggling with a crucial problem: How to take advantage of the analytics to leverage their sales as well as improve customer satisfaction? This requires one startup to do business model research, customer demand research, website design, and so on. Unless they have a massive budget, they can't be the next Best Buy or Amazon. They have to niche down to run a profitable E-commerce store. That's where a small but systematic customer evaluation system is needed. Many small companies have been aware of the necessity of digitization but have trouble finding patterns from historical data and improving customer satisfaction as well.

3 Methodology

3.1 Data Overview

We use the data from a small E-commerce platform selling major women's clothes. Because this is real commercial data, it has been anonymized, and references to the company in the review text and body have been replaced with "retailer". After removing the outliers and missing values from the dataset. The whole dataset contains 23486 samples with 11 features. Each line is a unique customer comment with the demographic of the customer and product information as well. Features include customer age, review title, rating on the product, recommend this product or not, division name, department name, class name, and so on.

3.2 Exploratory Data Analysis

To explore data, we have to grasp the relationship between explanatory variables, that's why we need a heatmap to measure the correlation, as well as teasing out inappropriate variables, avoid multicollinearity. From Figure 1, it is quite obvious that 'Rating' and 'Recommend IND' have a strong correlation, which is reasonable since the higher the rating is, the higher probability for a customer to recommend this product. Apart from this pair of variables, there's no sign of multicollinearity, all other variable pairs have pretty low correlation coefficients, which is good for our model building.

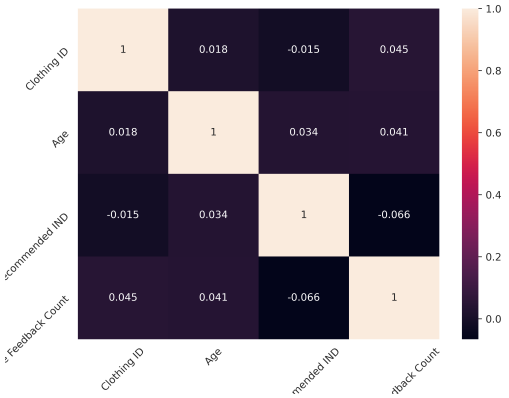

Figure 1: Correlation Matrix

Apart from "Review Text" and "Rating", there are variables like "Age", "Division Name", "Class Name" and "Department Name" that could reveal customer behavior patterns and help us understand the data on a deeper level.

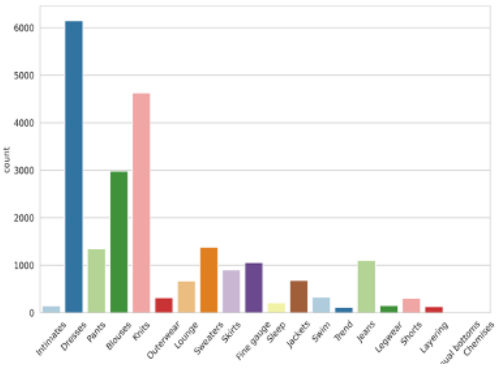

Figure 2: Breakdown of Product Class

Breaking down the distribution of age, we have found out that women between 30 and 40 years old took up most of our samples. Moreover, looking at the bar chart of the class name, we know that Dresses is the most popular product, followed by Knits and Blouses. They all belong to the department of Tops, so selling from this department is more than any else. Now we know that we are dealing with most comments from customers of around 30-40 years old who have bought Dresses, Knits or Blouses.

Let's focus on the review contents to get a quick understanding of our comments, we choose to calculate word frequency, construct a corpus and make a word cloud plot. We preprocessed the reviews, remove punctuations, numbers and keep only the noun words to build this word cloud. From the result, we can clearly see that "Dress", "Color", "Size" are the most frequent words, which is consistent with our common knowledge. Most of the content was focused on clothes' sizes, fabrics, sizes, and so on. However, when you look closely at less frequent words, many words (like 'X') are still meaningless to us, meaning that we still need to put more effort into preprocessing work for these comments.

Figure 3: Word Cloud from Review Corpus

Next, we want to know the simple polarity distribution before we move on to the real sentiment model building phase. Here we use TextBlob to simply judge the polarity of each review, and then plot polarity distribution covering all reviews under each Rating level.

Comments with a 5-star rating took up the biggest part, while the 1-star review group is the smallest part. Meanwhile, there's a right deviation

trend, the mean polarity of each score group tends to be larger with a higher score.

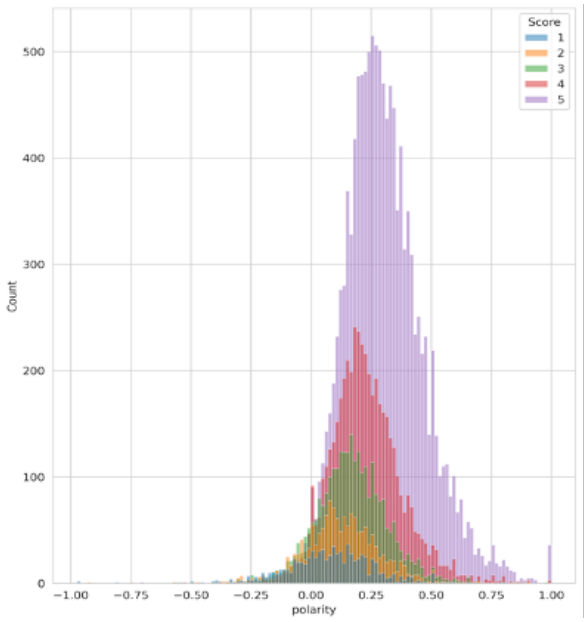

Figure 4: Polarity for Each Review

Most importantly, each group's polarity distribution follows a normal one. Considering a review with a polarity score over 0.5 to become a positive one, and ¡ 0 as negative, nearly 87% of the samples are classified as 'Neutral' with only 7.4% as 'Positive'. Results shown in Figure 4 reveal a necessity for further elaborate models to conduct sentiment analysis.

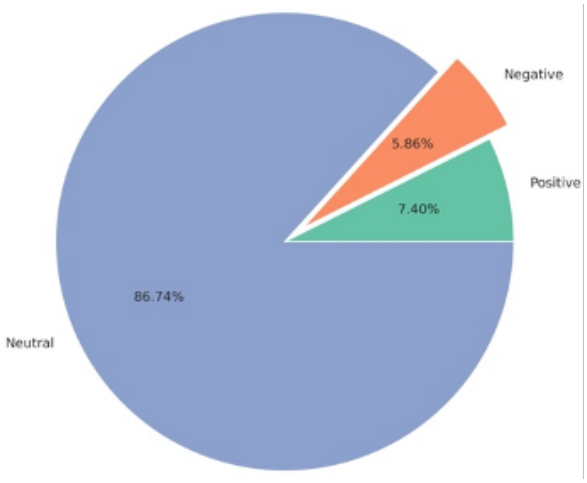

Figure 5: Pie Chart of Polarity

3.3 Text Preprocessing

A dataset of text without preprocessing will undercut the quality of text representation and model performance, especially when text data are comments from E-commerce websites. An extensive combination of the preprocessing method is wise to apply and likely to improve accuracy(HaCohen-Kerner et al., 2020). A closer observation of the dataset suggests that we need to take a few steps to filter the confounding information, reduce the data noises, and thus improve data quality.

The first step is to clean up the HTML tags, as they do not add value to the analysis. For the contractions to be treated equally with their full form, the contraction should be expanded. It is also crucial to exclude the return character (/r) and newline characters(/n) before replacing the punctuations with whitespace. In addition to these changes, we also remove the extra whitespace to avoid creating empty strings during tokenization.

As informal writing, the dataset contains elongated words, which will unnecessarily increase feature dimensions in one-hot encoding or TF-IDF, thus leading models to overfit. The way to address elongation in this article is to drop letters that repeat consecutively more than 2 times to a single letter. Removing accent marks is applied to further prevent needless feature inflation. As there are only five categories in the output, a word with less than 5-word frequency in the corpus may not be helpful to prediction. Hence, they will be removed. Following that, all words in the stopwords list from the NLTK package will be filtered. It is safe to ignore them because these words are ubiquitous, nearly of no worth to the text classification.

Finally, all words will be rendered to lower case and then reduce to their root form with Porter stemming algorithm.

3.4 Text Representation

Text representation, a process of converting words into numeric vectors, is a key step to take, as computers are good at dealing with numbers and finding their patterns(Yan, 2009). One of the simplest ways is the one-hot encoding. It works by representing words in a vector of binary, where only the index column has the value of 1. The length of the vector is determined by the unique words in a corpus.

The Bag-of-Words model builds a vocabulary from a corpus of review text and counts how many times the words appear in each review. To put it another way, each word in the vocabulary becomes a feature, and one review written by a customer is represented by a vector with the same length of the vocabulary. But this approach causes a significant

dimensionality problem: the more documents we have the larger is the vocabulary, so the feature matrix will be a huge sparse matrix. So before we set up the Bag-of-Words model, we preprocessed the review column, eg. clean the unnecessary words, remove the stop words, and stemming/lemmatization, to reduce the problem caused by dimensionality (Blanco, 2018).

Term frequency rewards words that appear many times in the same document. If a word occurs often in a review, then it is more important to the meaning of the review. But in fact, we could also find in the corpus common words with the highest frequency but little predictive power to the response variable. So IDF also rewards the words that appear many times in the whole dataset. That is, if a word occurs in two reviews, then it is more important to the meaning of each review. Bring them altogether, the total TF-IDF means that the value of a word increases proportionally to count, but it is inversely proportional to the frequency of the word in the corpus(Stephen, 2004). We also train the customized Word2Vec model on reviews to build the vocabulary. Word2vec represents words in vector space representation. Words are represented in the form of vectors where similar meaning words appear together and dissimilar words are located far away. It is a shallow two-layered neural network, where there is input one hidden layer and output (Mikolov et al., 2013). But neural networks do not understand text instead they understand only numbers. Word embedding provides the way to convert text to a numeric vector. To better represent sentence embedding, we also average the embedding of all words in a sentence, then we will get a one-dimensional vector for each review.

3.5 Data Split

When building sentiment classification model and rating prediction model, we split data into training set and test set according to the ratio of 80:20. We use training set to fit models and focus on the performance on test set.

3.6 Cosine Similarity

Cosine similarity is the most widely used measure to calculate the similarity of two non-zero vectors in product space. Given two vectors of attributes, X and Y, the cosine similarity can be represented as:

$$\begin{aligned} \mathit{CosSim}(X, Y) &= \frac{X \cdot Y}{\|X\|\|Y\|} \\ &= \frac{\sum_{i=1}^{d} X_i \times Y_i}{\sqrt{\sum_{i=1}^{d} {X_i}^2} \times \sqrt{\sum_{i=1}^{d} {Y_i}^2}} \end{aligned}$$

where X_i and Y_i are features of vector X and Y. Cosine similarity captures the orientation of vector X and Y. The value of cosine similarity ranges from -1 to 1. Vector X and Y in the same orientation have cosine similarity of 1. Vector X and Y in the opposite direction have cosine similarity of -1. And cosine similarity of 0 indicates decorrelation between vector X and Y.

In k nearest neighbors(KNN) models, cosine similarity can be used to target the k nearest neighbors for a given vector X. The nearest neighbors of X are those vectors with the largest values of cosine similarity(Qamar et al., 2008).

4 Model Building

4.1 Overview

Figure 6 shows the interaction process between users and our recommendation system. When a customer browses a web page, the recommendation system will filter out other similar products that he or she may be interested in based on the products he clicked on or placed an order. When a customer purchases a product and leaves a review on the web page, the review will be used as input for NLP models to predict the customer's corresponding attitude and rating score for the specific product. The extracted information of new reviews will be used to continuously update the product matrix in the recommendation system, and then retrain the recommendation system to make the recommendation system generate more accurate products.

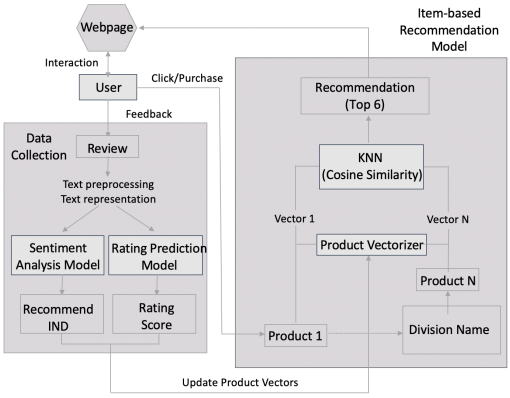

Figure 6: Framework of Model

4.2 Recommended Model with Sentiment Analysis

In this section, we will focus on predicting recommended columns based on customers' reviews. We use the count vectorizer, tf-idf vectorizer, and averaging word embedding three methods to construct the feature words matrix. But there were thousands of text features, and not all of these features we found in the text might be useful in building the model to make the necessary prediction. So we decide to do the feature selection to remove some useless tokens, which was also a good method to avoid overfitting.

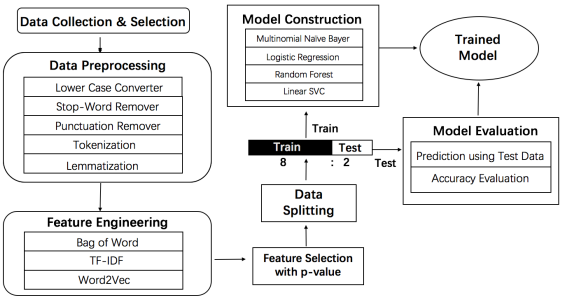

Figure 7: Sentiment Analysis Model

Correlation and p-value are both actional measures that we can use on the dataset to select the right features. (R, 2018) Here we choose to use p-value, which is a probability value for a given statistical model that, if the null hypothesis is true, a set of statistical observations is greater than or equal in magnitude to the observed results. Removal of different features from the big dataset will have different effects on the p-value for the dataset. We can remove different features and measure the p-value in each case. These measured p-values can be used to help us decide whether to keep a certain token feature or not. We set the p-value was equal to 0.997 to remove the unimportant features, which stood out of three standard deviations.

Then three models will be used to predict whether the customer would recommend the clothing he bought or not: Logistic Regression Model, Random Forest Model, Linear Support Vector Classifier Model, and Multinomial Naive Bayes Classifier Model. We would train the model with our training data and calculate the accuracy of the model using the test dataset.

Method-Model	Train	Test
Mul NB with BOW	0.8894	0.8743
LR with BOW	0.9238	0.8827
RF with BOW	0.8848	0.8271
LinearSVC with BOW	0.9022	0.8704
Mul NB with TF-IDF	0.8859	0.8594
LR with TF-IDF	0.9408	0.8876
RF with TF-IDF	0.8646	0.8222
LinearSVC with TF-IDF	0.9479	0.8828
LR with word2vec	0.8219	0.8093
RF with word2vec	0.8993	0.7895
LinearSVC with word2vec	0.8197	0.8061

Table 1: Different model performance with different text approaches.

Multinomial NB assumes that features have multinomial distribution, which is a generalization of the binomial distribution. Neither binomial nor multinomial distributions can contain negative values. Since Multinomial NB couldn't take negative values as input, we only tried the Logistic Regression model, Random Forest, and Linear SVC model with word2vec text presentation. Table 1 shows that TF-IDF performed better than bag of words and word2vec in the text representation. Linear SVC Model with TF-IDF predicted best in the training dataset with higher accuracy in both the training dataset and test dataset. It makes sense because Linear SVC is always effective in high dimensional space in text classification.

4.3 Rating Prediction Model

In this section, we will focus on predicting ratings based on customers' reviews. Two models will be used: one is the conventional model SVC; the other is the neural network. Before feeding into models, the reviews will be mapped into word vectors. For the word embedding representation, 200 dimension vectors are generated through CBOW and are aver-

aged to capture the semantics of reviews. As with the neural network model, an early-stopping based on the accuracy is included to prevent the model from overfitting.

Method-Model	Train	Test
NN with ono-hot	0.7554	0.6182
NN with TF-IDF	0.7419	0.6125
NN with word2vec	0.6136	0.6083
LinearSVC with one-hot	0.7943	0.6121
LinearSVC with TF-IDF	0.7333	0.6275
LinearSVC with word2vec	0.5982	0.5922

Table 2: Rating prediction accuracy from two models with three text representations

Table 2 shows that linear SVC with TF-IDF outperforms all the other models in term of accuracy, even though the differences among the models are subtle. Besides, the accuracy of linear SVC are likely to fluctuate as the text representation method changes.

Considering that the majority of label belongs to rating 4 and 5, F1-macro score is another metric worthy noticing. As far as F1 score is concerned, Neural Network with one-hot encoding has the best performance.

Method-Model	Train	Test
NN with ono-hot	0.6366	0.4437
NN with TF-IDF	0.6210	0.4212
NN with word2vec	0.3560	0.3469
LinearSVC with one-hot	0.7452	0.4035
LinearSVC with TF-IDF	0.6471	0.3832
LinearSVC with word2vec	0.3177	0.3001

Table 3: F1-macro score of predictions from two models with three text representations

When using one-hot or TF-IDF encoding, the model performs well in the training set, as their predictions are more accurate than that of the model using word embedding. However, this advantage is not as noticeable as in the test set, where their accuracy become much lower. This is a strong indicator that models with one-hot or TF-IDF encoding have the issue of overfitting. The model pays too much attention to details, picks up the noises from training, and fails to generalize the learned rule to the testing dataset. The problem of overfitting suggests that the model will not predict well on the future, unseen dataset.

The reason that overfitting happens only in the one-hot and TF-IDF is that there are too many features in their training set. The word vectors generated by these two methods are sparse matrices. Both matrices have 3356 features, almost 15 times the dimension of the matrix produced by word embedding.

Feature selection is an effective way to resolve overfitting. Its selection metrics can be divided into two groups: one-sided, such as Correlation Coefficient (CC), and Odds Ratio (OR), and two-sided, for example, Chi-square (CHI)(Zheng et al., 2004). The method we use is to reduce features by Chi-squared test. To conduct the test, we choose one category in the response variable and combine the rest categories into another group. With a null hypothesis that the response variables are independent from predictors, a Chi-squared test is performed on all matrix columns to filter out the significantly dependent features, whose p-value on the test is lower than 0.003. After all categories go through the feature selection process, 812 features are kept for the one-hot encoding, and 207 for the TF-IDF encoding.

With the new dataset, we retrained linear SVC and Neural Network and get the result as Table 4 and Table 5

Method-Model	Train	Test
NN with ono-hot after FS	0.6989	0.6315
NN with TF-IDF after FS	0.6439	0.6191
LinearSVC with one-hot after FS	0.6767	0.6098
LinearSVC with TF-IDF after FS	0.6150	0.6034

Table 4: Accuracy of predictions after feature selection

Method-Model	Train	Test
NN with ono-hot after FS	0.5451	0.4177
NN with TF-IDF after FS	0.4584	0.4030
LinearSVC with one-hot after FS	0.5337	0.3662
LinearSVC with TF-IDF after FS	0.3809	0.3214

Table 5: F1-macro scores of predictions after feature selection

The result shows that overfitting has been mitigated in both scenarios. Among all the models, NN with one-hot encoding after feature selection is preferred because it has highest test accuracy among models with slight overfitting.

4.4 Item-Based Recommendation System

For startups and small retailers in the E-commerce industry, it is wise to develop a recommendation system to stimulate customers' purchasing behavior. If a customer clicks or purchases a product from the website, we can use the recommendation system to suggest relevant products to customers.

There are several algorithms for recommendation systems, user-based collaborative filtering(UB-CF), item-based collaborative filtering(IB-CF), and the combination of these two methods.(Adomavicius and Tuzhilin, 2005) UB-CF method assumes that similar users will have similar behaviors. It recommends items by finding similar users to active users. Instead of focusing on similar users, IB-CF focuses on what products from all other options are more similar to what we know he or she enjoys. In terms of item-item similarity, we don't mean whether two products have the same attributes like blouses and knits because both of them are tops. Instead, item-item similarity means how people treat two items the same in terms of likes and dislikes.

One status quo for startups and small retailers is that they don't have much information about customers. Using UB-CF to look for similar users and predict customer behaviors would be very difficult. Also, the attributes of customers could be easily changed, it would force the company to recompute similar users and re-predict customer's behaviors frequently.(Deshpande and Karypis, 2004) Instead, IB-CF focuses on similarity between items, which have more stable attributes. Therefore, It would be more feasible to recommend products by item-based collaborative filtering method.

Take the women's clothing E-commerce platform as an example. The process of building our recommendation system is:

1. Use historical data to create product vectors. The attributes of each product includes the proportion of purchases of this product in different age groups, the average rating of each age group, the proportion of recommendation in each age group, as well as the division the product belongs to.

2. Divide products into different groups based on the division they belong to.

3. Build a K-Nearest Neighbors model to look for 6 most similar products in the same division by calculating the cosine similarity between products.

4. For customers who give a high score to a product, we will recommend 6 products with highest similarity to this product.

The advantages of applying IB-CF in such a platform are as follows:

- The results of IB-CF are much more stable than UB-CF since the average rating information of each product is far more than the average rating information of each customer.

- Over time, the amount of data for product reviews will continue to increase. The increase of the data size will make the value of the product vector more stable and reliable, thereby making the recommended solution given by the recommendation system more reliable.

In our recommendation system, we classify products by division and then recommend similar products in the same division. The reason we classify products is that different divisions often serve different types of customers. For example, products in division "General" are mainly for customers of average size, while division "General Petite" serves customers with petite shape and division "Intimates" serves customers who are looking for intimate products. Some products are not only designed for customers of average size, but also for customers of petite size. In this case, these products will be included in both divisions.

"Department" is the subcategory of "Division", while "Class" is the subcategory of "Department". As for these two subcategories, we are not limited to recommending products of the same class in the same department to customers because the same customer may also need products from different departments under the same division.

Table 6 shows the results of our recommendation system for an intimates product(Clothing ID: 767). The result of our model seems reliable. The similar products of product No.767 all belong to the Intimate department. Except for product No. 264, they are very popular with customers between 30 to 49. Even though product No. 264 seems to be more popular in the younger age group(18-29), the result is still acceptable. If more products or more reviewing data are put into the recommendation system, the result will be more accurate.

Product ID	40	439	264	283	322	767
Division	Intimates	Intimates	Intimates	Intimates	Intimates	Intimates
Department	Intimate	Intimate	Intimate	Intimate	Intimate	Intimate
Class	Intimates	Lounge	Legwear	Swim	Swim	Intimates
18-29	0	0	1	0	0	0
30-39	0.667	0.5	0	0.5	0.5	0.5
40-49	0.333	0.5	0	0.5	0.5	0.5
50-59	0	0	0	0	0	0
60-69	0	0	0	0	0	0
70+	0	0	0	0	0	0
Ave. Rating(18-29)	0	0	5	0	0	0
Ave. Rating(30-39)	4	4	0	4	4	4
Ave. Rating(40-49)	5	5	0	5	5	5
Ave. Rating(50-59)	0	0	0	0	0	0
Ave. Rating(60-69)	0	0	0	0	0	0
Ave. Rating(70+)	0	0	0	0	0	0
Recommend Prop(18-29)	0	0	1	0	0	0
Recommend Prop(30-39)	1	1	0	1	1	1
Recommend Prop(40-49)	1	1	0	1	1	1
Recommend Prop(50-59)	0	0	0	0	0	0
Recommend Prop(60-69)	0	0	0	0	0	0
Recommend Prop(70+)	0	0	0	0	0	0

Table 6: The outputs of recommendation system for product (Clothing ID: 767)

4.5 Testing Recommendation System on Another Dataset

To see if our recommender system can be applied to E-commerce startups of retail stores from a different industry, we use multiple datasets from an online bookstore to check if it will still work. This bookstore data have 3 csv files consist of customer demographic data, customer reviews, and book info. We randomly choose 100k samples to form each file and merge them, filter out variables that we want, cleansee the data and create a dataframe. The dataframe contains 49156 rows and 8 columns, which is similar to what we have done using the previous clothing dataset. We follow the same procedures, create product vectors and build the K-Nearest Neighbors model to predict 6 products with the highest similarity. The major difference is on feature engineering and selection, where we choose more variables including 'Year-of-Publication'. For a customer who has bought the book "Clara Callan" (ISBN:0002005018), we recommend him "Your Naturally Healthy Home", "The Last of the Mohicans", "Eye of the Beholder", "Make Them Cry", "Fade to Black" and "The Jasmine Trade". The test result shows that our K-NN recommend model has a functional and universal usage for different types of dataframe whatever the company sells.

4.6 Variations of Recommendation System

The difference between recommending outcomes using our original online clothing store data and book store data is mainly on the feature engineering part, where we choose different features and group by unique variable to create product vectors. For the bookstore dataset, we found one interesting feature "Year-of-Publication", which inspired us to develop a new method to recommend our customers the most similar books in each decade. We transformed this unique variable and integrated them into a function, which will end up recommending the most similar books based on each book's ID and the decade you want to see. This variation in our recommender model gives us a brand new view of what we could do. By modifying the features, we could keep adding new levels and recommend products on combinations of characteristics of the product, significantly improving our precision as long as the data have so much information. With this integrated function and method, we could provide our simple recommend system to meet different clients' demands, whether they

are small grocery stores, electronics stores, or even online pharmacies.

5 Conclusion

In this paper, we use E-commerce data sets with customers' characteristics and customer reviews to build NLP models to identify customers' attitudes toward specific products and predict customers' rating score based on natural language processing(NLP) techniques and machine learning models, while at the same time to build a recommendation system based on item-based collaborative filtering(IB-CF), which could be modified to help startups or small retailers conduct analytical work and to thrive in the E-commerce industry.

References

G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

José Blanco. 2018. Hacking scikit-learn's vectorizers.

Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation. *PLOS ONE*, 15(5):1–22.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

A. M. Qamar, E. Gaussier, J. Chevallet, and J. H. Lim. 2008. Similarity learning for nearest neighbor classification. In *2008 Eighth IEEE International Conference on Data Mining*, pages 983–988.

Vishal R. 2018. Feature selection - correlation and p-value.

Robertson Stephen. 2004. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520.

Jun Yan. 2009. *Text Representation*, pages 3069–3072. Springer US, Boston, MA.

Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. 2004. Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89.