Transactions on Computational and Applied Mathematics (2023)
Clausius Scientific Press, Canada
DOI: 10.23977/tracam.2023.030102
ISSN 2616-1826 Vol. 3 Num. 1

Methods of Analysis of Amazon Product Reviews and Rating Prediction

Shixun Huang[1], Xiaowen Zhang[2], Mingzhi Wang[3]

[1]*Mathematical Applications in Economics and Finance, 27 King's College Cir, University of Toronto, Toronto, Canada*
[2]*Computer Science, 27 King's College Cir, University of Toronto, Toronto, Canada*
[3]*Computer Science and Technology, Civil Aviation Flight University of China, 46 Section 4 Nanchang Road, Guanghan, China*

Keywords: review rating prediction, keyword extraction, sentiment analysis, TF-IDF, recurrent neural network

Abstract: Online shopping reviews have become an important data source for merchants to make smarter decisions in product development, operations, and marketing. In this paper, we propose a modeling strategy to optimize data analysis and processing of online shopping review data. We address four main problems: identifying commonly used words in positive, negative, and helpful reviews, predicting the products to which the comments refer using semantic analysis, predicting the product rating based on the comments using sentiment analysis, and proposing ways to distinguish human comments from machine-generated ones. Additionally, we provide a recommendation letter to customers on how to read product reviews.

1. Introduction

1.1 Problem Background

In this modern day and age, online shopping has become an inseparable part of many people, including us. We love the convenience of sitting in a penthouse in LA (which we don't have), buying a product from South Africa that's probably made in China. Time and effort are saved because customers do not have to be at the physical location where the products are sold.

However, this advantage of online shopping also comes with a drawback. Because customers are no longer in the physical location of the shop and being in contact with the products directly, they are not able to inspect the products and make a reasonable purchase based on their own knowledge. Hence, customers participating in online shopping mostly rely on two information sources: the description written by the merchant, and the reviews written by customers.

The description written by the merchant is usually not a good representation of the overall quality of a product. Since the end goal of the average merchant is to sell as many products as possible, their description is usually one-sided and includes exaggeration and dissemblance.

Many customers are aware of this. Hence, they often go to the review section for more objective information. However, being able to obtain truly objective information from the reviews is difficult.

Reasons include: each person has their own rating scheme and expectation of the product; merchants sometimes pay people to leave fake review; machine-generated reviews could be more often than expected.

In order to help customers with gathering information on a product, this paper attempts to perform semantic and sentiment analysis of product reviews on amazon. A few evaluation criteria for human vs machine-generated reviews are proposed.

1.2 Restatement of the Questions

Based on our understanding of problem background and the questions listed in the problem statement, we need to perform the following task:

o Analyze word frequencies in the reviews in the appendixes provided.

o Extract keywords from the reviews.

o Perform semantic analysis on the reviews, predict the name of the product a review refers to.

o Perform sentiment analysis on the reviews, predict the overall rating that corresponds to the review.

o Propose evaluation criteria for distinguishing between human and machine-generated reviews.

2. Definitions and Notations

2.1 Definitions

NLP Natural Language Processing

Token (Term) The smallest meaningful unit in NLP, such as a word

TF Term frequency

DF Document frequency

IDF Inverse document frequency

Specificity The uniqueness of a token in a document by contrast with other documents that could belong to other domains [1]

Representativity The degree to which a token convey the meaning of the document, in contrast to other words that only reflect minor aspects [1]

Keyness The specificity and the representativity of a token

Stopword A word in a language that is frequently used but often have no semantics when not in context, such as “the”, “a”. and “it” in English

RNN Recurrent Neural Network **LSTM** Long Short Term Memory **GRU** Gated Recurrent Unit

2.2 Notations

The key mathematical notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper

Symbol	Description
w_i	the i^{th} word in the document
TF_{w_i}	the term frequency of w_i
IDF_{w_i}	the inverse document frequency of w_i
DF_{w_i}	the proportion of documents containing w_i
$\#$	number
$TF(U(t))$	the number of occurrences of the candidate term t starting with an uppercase letter
$SF(t)$	the sentence frequency of the candidate term t, i.e., the number of sentences in which t appears

3. Model I: Rating Prediction with TF-IDF Keyword Extraction

3.1 Rating Prediction with TF-IDF Keyword Extraction Theory

According to the questions, we need to extract keywords from the reviews, predict the product names, and the ratings associated with the reviews. We realize that these three tasks are really one. We first compute the keyness of individual words in the review. Then, we combine the meaning of the words, each weighted by their keyness, to form the sum of the meaning of all words, i.e. the review. Using the sum, we can determine the sentiment, thus, the rating, of the review.

3.2 Rating Prediction with TF-IDF Keyword Extraction Method

Our Rating Prediction with TF-IDF Keyword Extraction model can be divided into the following components: DF calculation, keyness calculation, token vectorization, document vectorization, and model training.

3.2.1 TF-IDF Introduction

TF-IDF (Term Frequency-Inverse Document Frequency) is a popular technique used in information retrieval and NLP to represent the keyness of a term in a document of collection of documents. TF is the number of times a term occurs in a document. A higher TF usually means a higher representativity. IDF is a measure of how rare or unique a term is across a collection of documents. A higher IDF usually means a higher specificity.

$$f(x,t) = \begin{cases} 1, & if\ x = t \\ 0, & otherwise \end{cases}$$

$$\mathrm{TF(t,d)} = \sum_{x \in d} fr(x,t)$$

$$\mathrm{D(t,C)} \sum_{d \in c} \begin{cases} 1, & if\ x = t \\ 0, & otherwise \end{cases}$$

$$\mathrm{IDF(t,C)} = \ln \frac{count(C)}{DT(t,C)}$$

TF-IDF value is the product of the TF (representativity) and IDF (speficity) of a term. It captures our definition of keyness very well.

$$TFIDF(t,d,C) = TF(t,d) \times IDF(t,C)$$

3.2.2 DF Calculation

To calculate the TF-IDF value of a token in a document, we must first determine the DF of every token in the collection of documents. After the DF of every token is calculated, a mapping between tokens and their DF is created. This enables us to access the DF of any token later on in the process.

3.2.3 Keyness Calculation

As previously mentioned, the TF-IDF value captures our definition of keyness. Thus, we use the TF-IDF value of the tokens to determine their keyness in the document and rank them from the most key to the least key.

3.2.4 Token Vectorization

In order to perform calculations, we must find ways of transforming tokens into numeric values. One way of doing this is to use word embeddings. Word embedding is a technique used in NLP to represent words as dense numerical vectors in an Nd space, where words with similar meanings or context have similar vectors. The main idea behind word embedding is to transform the semantic and syntactic relationships between words into a format that can be used for machine learning. The embedding process is typically performed using neural network-based methods.

For this model, we use a pretrained embedding provided by Stanford University. It is trained on 2 billion tweets and transforms words into 100d vectors [3]. We create a mapping between all tokens in the documents and their corresponding 100d vectors.

3.2.5 Document Vectorization

For each document in the collection of documents, we attempt to transform them into 100d vectors as well.

First, we split a document into tokens using NLTK's tokenizer [5]. Then we calculate the TF of every token. Using the TF created at this step and the DF created in the previous step, we calculate the TF-IDF score of the tokens. Finally, using the TF-IDF score and the embedding, we calculate the 100d vector representing the document.

$$\vec{d}, \{d \in C\} = \frac{\sum_{t=d} TFIDF(t, d, C) \times \vec{t}}{count(d)}$$

3.2.6 Model Training

The most intuitive way of formulating this rating prediction is to consider it as a multiclass classification task. We would have a five-by-one-hundred weight vectors, each corresponding to a rating (1.0, 2.0 …). After computing the dot product between the document vector and all weight vectors, we obtain five values, each corresponding to a rating. To normalize the values, we feed them into a SoftMax function, which outputs the probabilities of the review being each of the rating. We also need to turn the scalar rating into one-hot vector of probabilities, in which the element at the index of rating would be one, and all others would be zero. Then we calculate the loss of each cell individually and update our weights accordingly.

$$\vec{z}(\vec{d}) = w^T \vec{d} + \vec{b}$$

$$soft\max(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z} j}$$

3.3 Rating Prediction with TF-IDF Keyword Extraction Result

3.3.1 DF Calculation

We first calculate the TF of all tokens across the reviews in Appendix I and obtain the data shown in *Figure 1*.

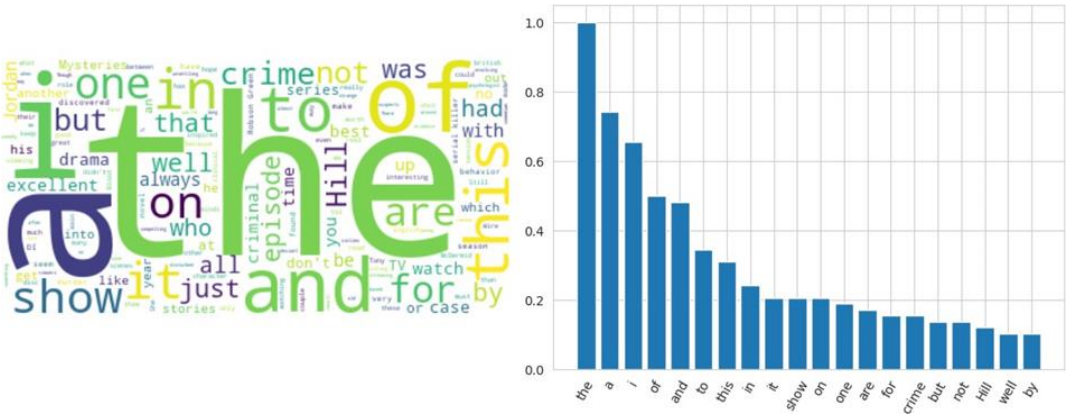

Figure 1: Wordcloud and TF of Appendix 1

Unsurprisingly, the wordcloud is flooded with stopwords. To make meaningful analysis, we decide to remove the stopwords and obtain the data shown in *Figure 2*.

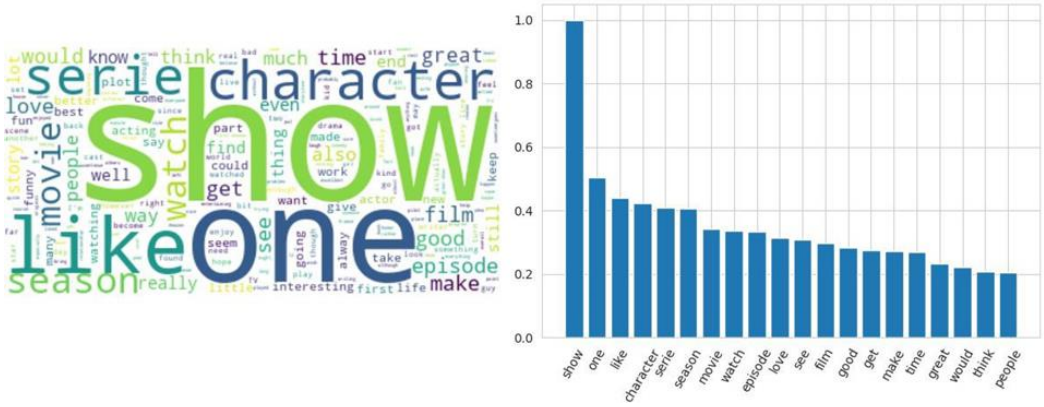

Figure 2: Wordcloud and TF of Appendix 1 Without Stopwords

After removing the stopwords, we can see some meaningful patterns. The word "show" lead the pack, appears approximately twice as often as the second most common non-stopword. From this, we conclude that the reviews in Appendix II are about TV shows. We look up eleven most frequent ASIN in the data, accounting for 10 percent of the reviews. And indeed, six of them are TV shows, and five of them are episodes of TV shows!

Our goal for this step though, is to calculate the DF of all tokens in the collection, not the TF of them across the collection. Using Appendix 2, we repeated the process of drawing wordcloud and bar plot, but this time using DF instead of TF. The word cloud maps are shown in *Figure 3 and Figure 4.*

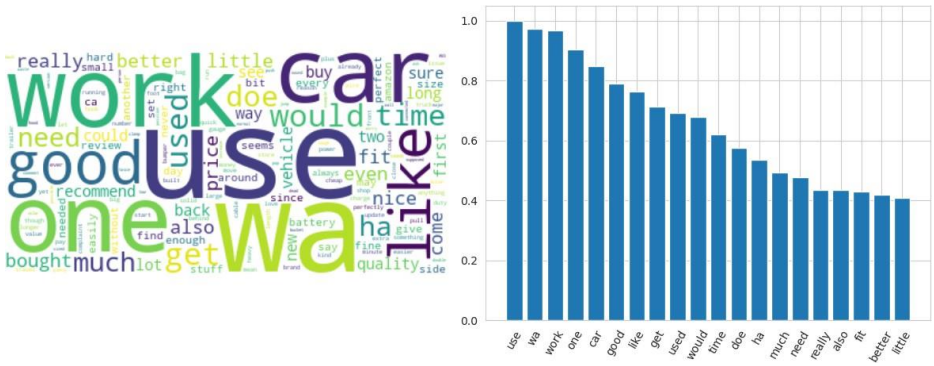

Figure 3: Wordcloud and DF of Appendix 2

It is curious to see that "wa" has the second highest document frequency. However, having misspelled words in our analysis is not desired. Thus, we limit the words to nouns provided by Princeton University [4].

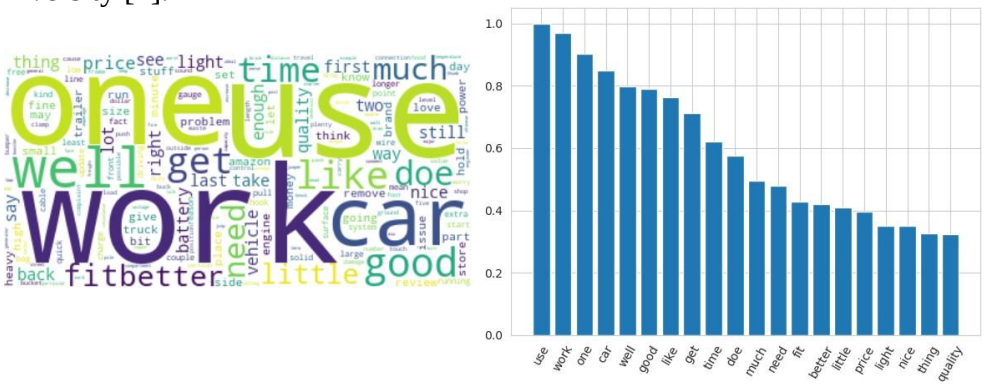

Figure 4: Wordcloud and DF of Appendix 2 (Nouns Only)

Now only nouns are included in our wordcloud. Based on the words "use", "work", and "car", we conclude that the products in Appendix 2 are car-related products. We look up eleven most frequent ASIN in the data, accounting for five percent of the reviews. They are all car accessories, confirming our hypothesis.

3.3.2 Keyness Calculation

To calculate the TF-IDF of a token, we need to first calculate its TF. Then we multiply its TF with its IDF. The product is its TF-IDF value.

To demonstrate this, we randomly sample a review from Appendix 3, and perform TF-IDF calculation on all tokens as shown in *Figure 5*.

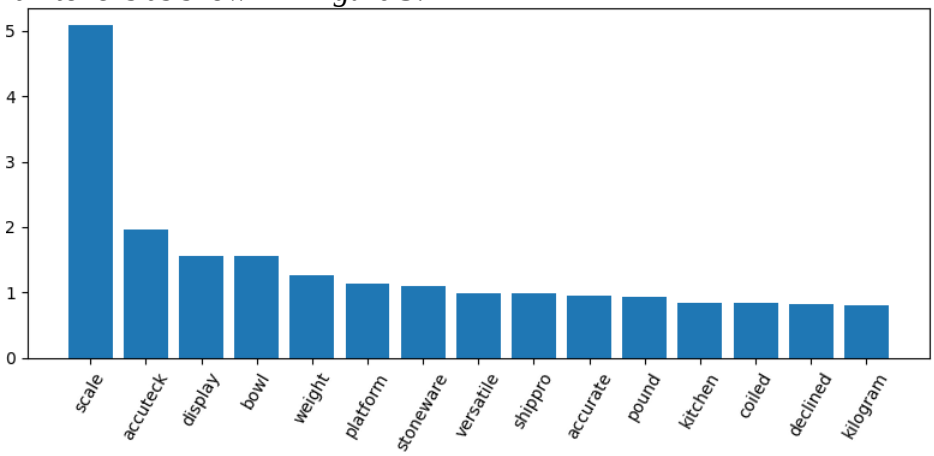

Figure 5: TF-IDF of tokens in a random review in Appendix 3

From this graph, we could see that the TF-IDF matrix does produce reasonable results. As the name and the brand of the product are the top two words according to this metrics.

We also attempt to incorporate several other metrics proposed in YAKE! (Campos, et al., 2020) [1] for better performance as shown in *Figure 6*.

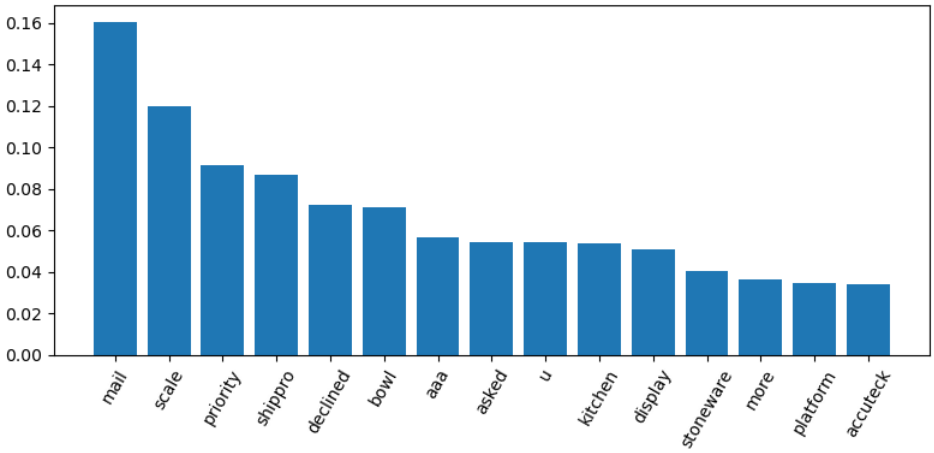

Figure 6: Customized Metric Value of tokens in a random review in Appendix

Mail, a word that is not related to the product in any way, is incorrectly labeled as the most key token in this review, which is not desired. Further calculation shows that the inter-agreement between the TF-IDF method and the modified YAKE! Method is less than 0.5. Because of this, we decide to stick with the TF-IDF metric.

3.3.3 Keyword Extraction

Now we have the keyness values of all tokens in a review, we can rank them and take the top k as keywords [2]. Using the keywords, we can predict what words are in the name of the product.

We run our TF-IDF keyword extractor on all reviews in Appendix 4. We then randomly select two asins, manually search for the product name, and compare the actual name with the keywords generated by our keyword extractor.

o Yamaha FC5 Compact Sustain Pedal for Portable Keyboards, black, asin B00005ML71

o Our top five keywords are keyboard, pedal, piano, sustain, Yamaha.

o Nady SP-4C Dynamic Neodymium Microphone, asin B00009W40D

o Our top five keywords are mic, nady, karaoke, interest, cube.

Based on these two examples, we are confident with the ability of extracting keywords of our keyword extractor.

3.3.4 Token Vectorization and Document Vectorization

These two steps involve turning human-readable data into machine-learning-friendly data that are not human-readable. The outputs of these two steps are 100d vectors. Hence, we will not show any outputs.

3.3.5 Model Training

After transforming text into vectors, we now can leverage machine learning to make predictions. Before training starts, we are going to inspect the data in Appendix 5 as shown in *Figure 7*.

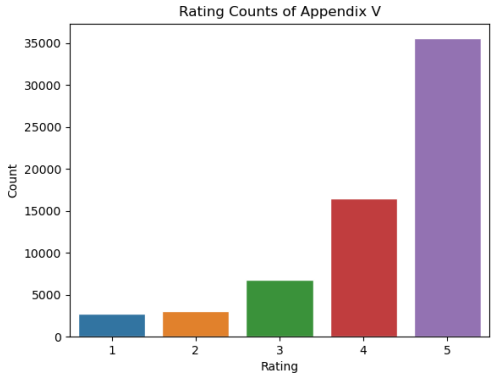

Figure 7: Rating Counts of Appendix 5

This data is heavily biased towards higher ratings. We see similar patterns in all other Appendixes. It is not unexpected, as many people tend to give a rating of five when the product meets expectations. Ratings of one and two are only given when there is a significant problem, which happens rarely. To mitigate this issue, we use class weights during training, so the underrepresented classes have more impact on the weights than the overrepresented classes. The class weights are calculated and mapped in *Figure 8* as follows:

$$classweights(c, classes) = \rightarrow \frac{\sum i \in classes\ count(i)}{count(c) * count(classes)}$$

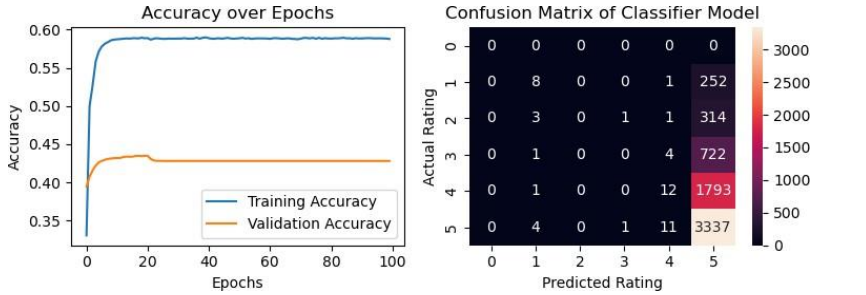

Figure 8: Accuracy and Confusion Matrix of Classifier without Class Weights

Initially, we train the model without using the class weights. The validation accuracy of our model plateaued around 0.43, which is much lower than we anticipated. Despite effort at hyperparameter tuning, our final test accuracy is 0.52. Inspecting the confusion matrix, we find that the bias in the original training data has a huge impact on model performance. More than 90% of the predictions made by our model are fives. See *Figure 9* for details.

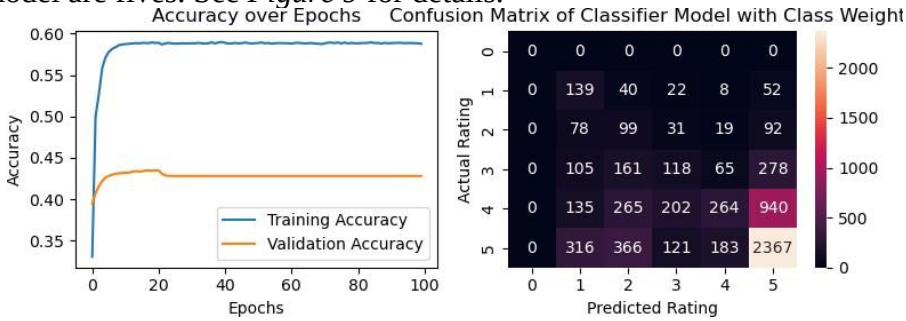

Figure 9: Accuracy and Confusion Matrix of Classifier with class weights

Despite a drop in the validation accuracy and test accuracy by 0.01, we are happy with the improvements made by our model. Instead of constantly predicting five, which gives a high recall but

a low precision, our model is now making useful predictions.

One cell that stands out in the confusion matrix is when the actual rating is four but our model predicted five. To find out why, we train the models again on Appendix 6 and attempt to find a few examples in which our model give a false rating of five.

One review reads “BEEN USING NOW FOR 3 YEARS. SLOWLY REPLACING ALL HOSES WITH THESE. KEEP THEM WRAPPED IN A WIND UP STORAGE CONTAINER AND IT LOOKS LIKE I SHOULD HAVE MANY YEARS OF QUALITY HOSES.” Usually, when a human sees a review like this, they would usually think this has a rating of five. However, its true rating is four. So, it is difficult to determine the real value of our model compared with the average human.

4. Model II: Rating Prediction with RNN

4.1 Rating Prediction with RNN Theory

The approach used in Model I is ultimately a bag-of-words approach. This means a document is solely characterized by the tokens in it. The relationship between tokens, however, is not considered. This model considers the relationship between tokens. We can capture the context of a token if we keep a record of previous and following tokens.

4.2 Rating Prediction with RNN Method

Two steps are conducted when we train this model to predict rating: data preprocessing and machine learning. This model relies heavily on the machine learning part.

4.2.1 Data Preprocessing

Different from the previous model, we do not convert the document to a bag-of-word format. Instead, we lemmatize all tokens in the document using tools provided by [5], and create a mapping between lemmas and 100d vectors, using a pretrained word embedding (Pennington, Socher, & Manning, 2014) [3].

4.2.2 Machine Learning (RNN)

A specific type of machine learning architecture is employed in this mode, RNN. RNNs are designed to handle sequential data, making them well-suited for processing text, which is inherently sequential. RNNs can maintain a hidden state that captures information from previous words or tokens in the text, allowing them to model the context and dependencies between words effectively.

In this model, we use the GRU variation of RNN as shown in *Figure 10*.

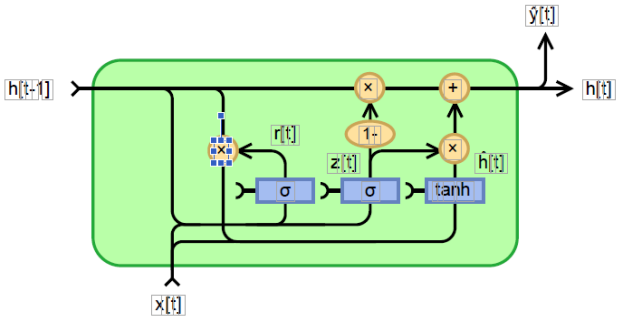

Figure 10: Gated Recurrent Unit, Jeblad, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 11: GRU Formulae https://en.wikipedia.org/wiki/Gated_recurrent_unit

The update gate (z) determines how much of the previous hidden state should be retained and how much of the new input should be added to the current hidden state. It controls the balance between remembering old information and updating with new information. The GRU formula is shown in *Figure 11*.

The reset gate (r) controls the amount of information from the previous hidden state that is incorporated into the current input. It helps the model decide which part of the past information to forget and which part to pass on to the next step.

The hidden state (h) carries information from one time step to the next. The output of the GRU at each time step is usually the hidden state, which can be used for predictions or further processing.

In addition to the usage of GRUs, this model uses bidirectional architecture as well. As shown in figure 12.

Bidirectional RNNs offer several advantages for text processing tasks. In traditional RNNs, information flows only from past to future time steps. However, bidirectional RNNs process the input sequence in both forward and backward directions simultaneously. This means that the model can capture context from both past and future words, allowing it to have a more comprehensive understanding of the entire input sequence.

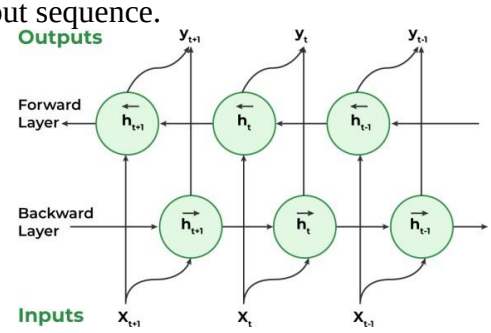

Figure 12: Bidirectional RNN

4.3 Rating Prediction with RNN Result

As we discovered in 4.3.5, Appendix 5 and 6 are heavily skewed towards higher ratings.

Thus, we default to using class weights to compensate that problem. We used the data in Appendix 6 for this training task.

Eventually the model reaches a test accuracy of 0.45 on Appendix 6, which is still quire underwhelming. However, by observing the confusion matrix in *Figure 13*, we can see that the model is capable of predicting the general sentiment of the reviews i.e. whether they are positive or not. We see that most of the predictions are with in one from the true rating.

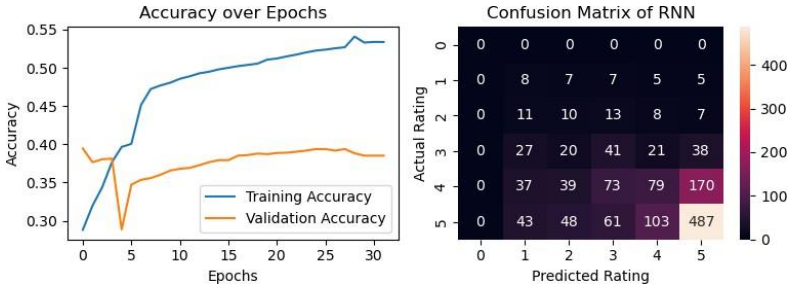

Figure 13: Accuracy and Confusion Matrix of Model II on Appendix 6

5. Comparison between Model I and Model II

Model 1, our keyword extractor, has shown promising results in predicting the products to which comments refer using semantic analysis. This is an essential tool for merchants to gain a deeper understanding of consumer needs and preferences, which can inform their product development and marketing strategies.

Model 2, Bidirectional RNN with GRU, reaches higher accuracy than model 1 in sentiment analysis and review rating prediction.

Overall, the performance of Model II is significantly better than Model I. Its test accuracy is 30% - 50% higher than Model I across the Appendixes. However, Model I is more light-weight, being easier and faster train, uses less parameter, and still reaches an accuracy that's at least 80% higher than random on all datasets.

6. Conclusion

In conclusion, online shopping review data is a valuable resource for merchants to optimize their products, services, and operational strategies. Our modeling strategy provides a comprehensive and effective approach for analyzing and processing online shopping review data.

We believe that our recommendations will help merchants make better decisions and improve their competitiveness in the market. Additionally, our recommendation letter to customers provides guidance on how to read and interpret product reviews, helping them make informed purchasing decisions.

Overall, our research has shown that online shopping review data is a valuable resource for merchants. By effectively utilizing this data, merchants can gain valuable insights into consumer needs and preferences, which can inform their decision-making processes. Our models have demonstrated their potential for extracting valuable insights from review data, which can be used to optimize product development, operations, and marketing strategies.

References

[1] Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. Information Sciences, 509, 257-289.

[2] Firoozeh, N., Nazarenko, A., Alizon, F., & Daille, B. (2020). Keyword extraction: Issues and methods. Natural Language Engineering, 26(3), 259—291.

[3] Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In Empirical Methods in Natural Language Processing (pp. 1532-1543).

[4] Princeton University. (2010). Retrieved from WordNet: https://wordnet.princeton.edu/.

[5] Zhu, T. (2021). From Textual Experiments to Experimental Texts: Expressive Repetition in "Artificial Intelligence Literature". Theoretical Studies in Literature and Art, 41(5), 140—147. (n.d.). Retrieved from NLTK :: Natural Language Toolkit: https://www.nltk.org/index.html#