Unraveling Sentiment Trends in TripAdvisor Reviews using Recurrent Neural Networks

1st Xiaokai Rong
Department of Computer Science
The University of Texas at Dallas, USA
xxr230000@utdallas.edu

2nd Ching-Yi Chang
Department of Electrical Engineering
The University of Texas at Dallas, USA
cxc190002@utdallas.edu

***Abstract*—In order to thoroughly explore recurrent neural networks (RNNs), this paper created a unique model from scratch. We explore the underlying theories, concentrating on recurrent connections and hidden states. The back-propagation via the time algorithm for training and the architecture of the customized RNN are also described. A thorough analysis of benchmark tasks shows how well the model captures sequential patterns. The new information improves our comprehension of RNNs and advances deep learning and sequential data analysis.**

I. Introduction

Recurrent Neural Networks (RNNs) have taken the lead in processing sequential data because of their cyclic connections, which allow for information retention over multiple time steps. This study investigates RNNs by building a unique model from the start to gain knowledge about their fundamental concepts and their basic concepts. We stress the RNNs' recurrent nature and the importance of hidden states in maintaining historical data. Our aim is to understand the mathematical foundations and algorithmic design underpinning the RNNs' behavior. By creating the RNN model from scratch, we get insightful knowledge of the design and training procedures, laying the groundwork for understanding more intricate RNN variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks. This research thoroughly comprehends the principles underlying RNNs by disregarding current libraries, paving the path for taking on real-world applications.

II. Related Work

The vast repository from the internet as well as class instructions, play an important role in our journey. All our referenced work is listed in the appendix. There is tremendous work related to LSTM and how to implement that network. We just came up with a very straightforward, easier-to-implement network for this project; the model is fundamental and contains the fundamental concept of what RNNs should be. Adding more layers or weight parameters is very easy based on our model [1]

III. Methods or Methodology

A. details on data collection

Our dataset is the TripAdvisor hotel reviews dataset, collected from an online Kaggle competition platform. TripAdvisor is the largest travel review website. It contains millions of user-hand-written reviews covering a variety of aspects of the hotel from diverse destinations. This dataset is a commonly used dataset when it comes to sentiment analysis and some natural language processing (NLP) demonstration. Since the dataset is plain text, we want to preprocess it into a simpler format that makes it easier for our machines to tokenize and label each word and incorporate user ratings.

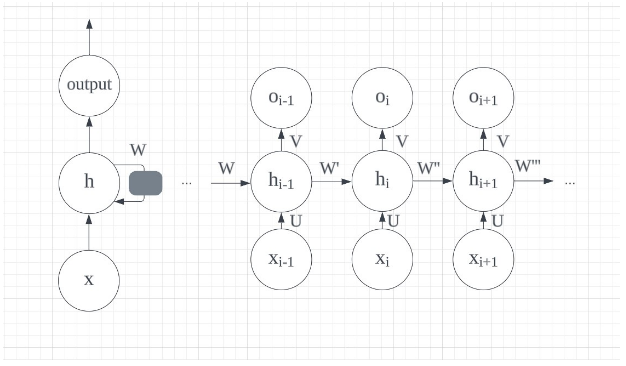


Fig. 1. Simple Model Showcase

The dataset primarily contains guest reviews and their corresponding labels. From rating 1 to 5. We aim to analyze the text or user review and give a reasonable "guess" of their rating of that specific hotel. We limit each entry(review sentence) to the max of 200 words of plain text with a labeled rating. The dataset contains more metadata like location, name, review date, etc., both positive (scores above 4) and negative (scores below 2); however, we do not use data other than ratings, Because this will greatly increase the complexity of the model.

B. experimental design

To achieve our model shown in Fig.1, we assume our activation function is $tanh$, and second, we assume the output from our model will be discrete. The left side of the figure shows the simple nonation of RNNs model. Input to hidden to output layers. On the right side, is the full network showing how RNNs will function through iterations.

X represents the input or input layer, into the network throughout the time. H represents the hidden layer. Current input and previous hidden state with respect to the time step will be calculated in the hidden layer, using the tanh activation function. There are three weights in this model: U, V, W. U is the input-to-hidden feed weight matrix, W is the hidden-to-hidden feed weight matrix, and V is the hidden-to-output layer feed weight matrix. Another important thing is the role of hidden size in sentiment analysis, using RNNs to capture and maintain information about the context and dependencies in the input text sequence. In this project, we set 64 hidden units.

C. algorithms and model architecture

Several important steps are involved in the data preprocessing and model architecture: **Tokenization**, before feeding text data into the RNN, it needs to be tokenized. Tokenization is the process of breaking the text into individual tokens. Each token represents a basic unit of the input sequence. **Padding**, in a batch of sentences, each sentence may have a different length. However, RNNs require inputs of the same length. Padding is the technique used to make all sentences in the batch of equal length by adding special padding tokens to shorter sentences. Each sentence is constrained to a maximum of 200 words. **Word-to-Index Conversion**, once tokenized, each word in the text needs to be converted into a unique numerical representation. This is typically done by creating a vocabulary, assigning an index to each word in the vocabulary, and then converting the tokenized text into sequences of word indices. **RNN Structure**, the core of sentiment analysis lies in the RNN structure, which plays a crucial role in understanding the sequential nature of text data. The RNN used for sentiment analysis typically employs Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells. These cells are designed to capture long-term dependencies and maintain context information over the sequential input data. By processing each token in the sequence and updating its hidden state, the RNN learns to capture the context and sentiment information throughout the text. **Softmax Layer**, the final layer of the RNN model is usually a softmax layer. Softmax is a mathematical function that converts the output scores of the RNN into probabilities. It transforms the RNN's raw output into a probability distribution over different classes (e.g., positive, negative) for sentiment analysis. **Argmax Operation**, After obtaining the probabilities from the softmax layer, the argmax operation is used to find the index corresponding to the class with the highest probability. This index represents the predicted sentiment class for the given input sentence. By effectively combining these steps and components, the RNN model can analyze sentiment in text data and provide predictions for positive or negative sentiment based on the input sentences. However, the heart of sentiment analysis lies in the RNN structure, which allows the model to grasp the sequential dependencies and context within the text, leading to more accurate sentiment predictions.

1) Forward Pass: Forward Pass: Perform the forward pass as described earlier to compute predictions using the current model weights.

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh\left(\boldsymbol{a}^{(t)}\right) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}\left(\boldsymbol{o}^{(t)}\right) \end{aligned} \tag{1}$$

2) Computing Gradient: Update the model's weights using gradient descent to minimize the loss function. The weights are adjusted in the opposite direction of their respective gradients.

$$W \leftarrow W - \alpha \frac{\partial L}{\partial W} \tag{2}$$

3) Back Propagation Through Time(BPTT): Backpropagation through time is the specific algorithm used to compute gradients in RNNs during the training process. Since RNNs have a recurrent structure that involves processing sequential data, traditional backpropagation (used in feedforward neural networks) cannot be directly applied. BPTT extends the concept of backpropagation to RNNs by unfolding the recurrent computation over time. As following U, W, V.

$$\begin{gathered} \nabla_{c} L - \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}}\right) \nabla_{\boldsymbol{o}^{(t)}} L - \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \\ \nabla_{\boldsymbol{b}} L - \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^{\top} \nabla_{\boldsymbol{h}^{(t)}} L - \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L \\ \nabla_{\boldsymbol{V}} L - \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{\boldsymbol{V}^{(t)} o_i^{(t)}} - \sum_t \left(\nabla_{\boldsymbol{o}^{(t)}} L\right) \boldsymbol{h}^{(t)^{\top}} \\ \nabla_{\boldsymbol{W}} L - \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} - \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{h}^{(t-1)^{\top}} \\ \nabla_{\boldsymbol{U}} L - \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} - \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \boldsymbol{x}^{(t)^{\top}} \end{gathered} \tag{3}$$

IV. RESULTS

To evaluate the effectiveness of sentiment analysis, when train size = 80% and test size = 20%, we could refer to train result as Fig. 2, test result as Fig. 3, classification report as Fig. 4. Impressively, this model achieved an accuracy of 77.1% as Fig. 5 and Table I, showcasing its capability to effectively capture sentiment patterns.

V. DISCUSSION

A. Preprocessing Techniques for Sentiment Analysis

Limiting Maximum Review Length have both advantages and disadvantages, and it depends on the specific context

TABLE I
SUMMARY OF OUR MODEL

Train	Test	Epoch (min)	Words	Accuracy
2%	0.5%	1	2975	77.5%
5%	1.25%	2	5367	72.3%
10%	2.5%	4	7934	67.2%
20%	5%	12	12369	75.1%
50%	12.5%	45	21573	77.4%
80%	20%	90	28991	77.1%

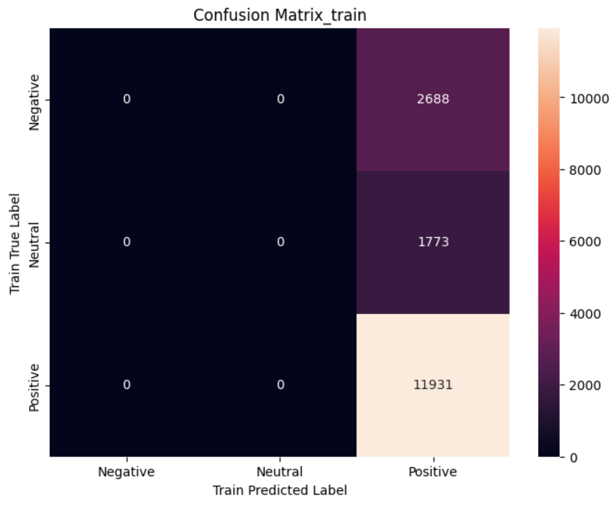


Fig. 2. Confusion Matrix of train data

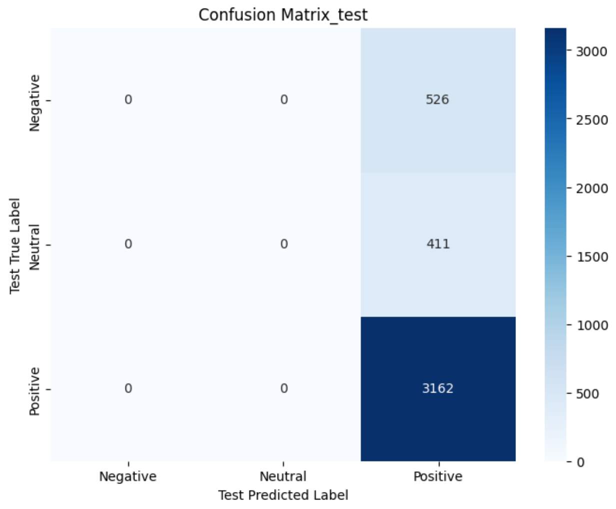
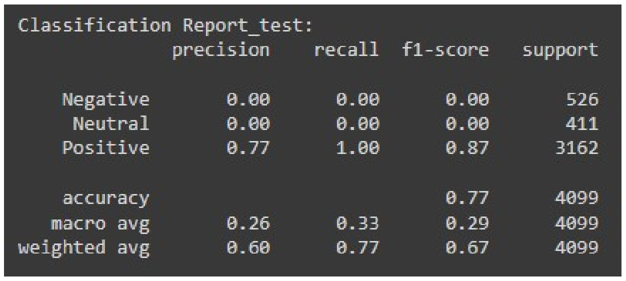


Fig. 3. Confusion Matrix of test data

and requirements of the application. Advantages for **Memory and Computation Efficiency**, by limiting the maximum review length, the RNN model requires less memory and computational resources during training and inference. This is particularly beneficial when dealing with large datasets or limited hardware resources. Also good for **Reduced Overfitting**, longer reviews may contain more noise or irrelevant information that could negatively impact the model's generalization. By limiting the length, the model is encouraged to focus on the most relevant parts of the review, potentially reducing overfitting. However, there are still some side effects, **Loss of Context**, limiting the review length may result in the loss of important contextual information. Some sentiment-critical phrases or opinions might be truncated or excluded, leading to potentially less accurate sentiment predictions. **Sentence Fragmentation**, if a review is too long and gets cut off, it might lead to sentence fragmentation, causing the model to lose the complete syntactic and semantic structure of the sentence, making it challenging to interpret sentiment accurately.


```
Classification Report_test:
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       526
     Neutral       0.00      0.00      0.00       411
    Positive       0.77      1.00      0.87      3162

    accuracy                           0.77      4099
   macro avg       0.26      0.33      0.29      4099
weighted avg       0.60      0.77      0.67      4099
```

Fig. 4. Test data classification Report

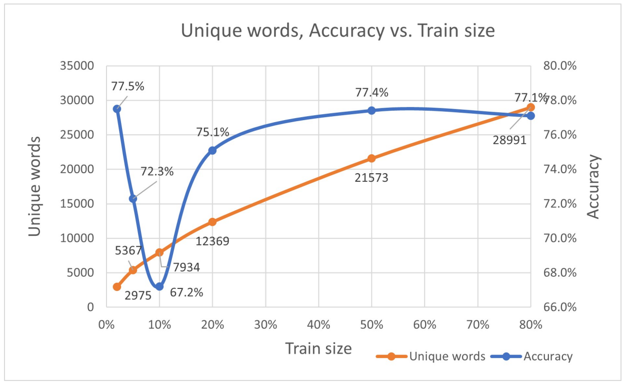


Fig. 5. Unique words, Accuracy vs. Train size

B. *Utilizing Word Vectors and argmax for Sentiment Identification*

Some advantages for **Better Generalization**, utilizing word vectors helps the RNN model generalize better to unseen words and sentences. Word embeddings can capture word similarities and analogies, enabling the model to infer sentiments for words that might not appear frequently in the training data. **Argmax for Sentiment Prediction**, using the argmax operation to obtain the final sentiment prediction from the softmax probabilities simplifies the decision-making process. It selects the class with the highest probability, providing a clear and straightforward sentiment identification. In contrast of, **Out-of-Vocabulary (OOV) Words** Word embeddings are limited to the vocabulary used during training. If the input contains words not present in the word embeddings' vocabulary, they will be considered OOV words and may not contribute to sentiment analysis accurately. **Contextual Information Loss** Word embeddings treat each word independently and do not fully capture the context in which words appear within sentences. This can lead to some loss of contextual information, which is essential for sentiment analysis.

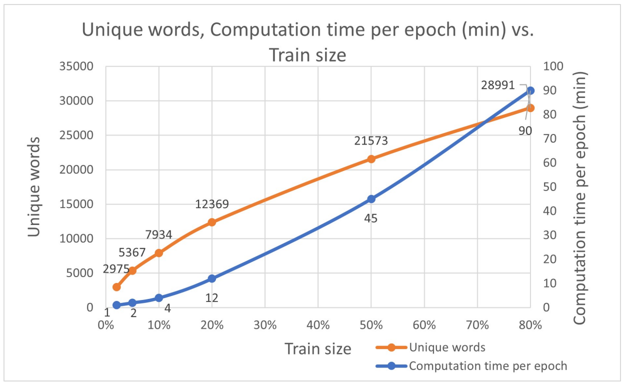


Fig. 6. Unique words, Computation time per epoch (min) vs. Train size

VI. Conclusion

In this research project, we conducted an extensive sentiment analysis of TripAdvisor reviews, employing various preprocessing techniques to enhance the accuracy of our analysis. We successfully standardized sentence lengths and efficiently processed a large dataset of 20,491 reviews by implementing stop word removal, punctuation handling, and word vectors for sentence representation. During our exploration of the unique words present in the reviews (approximately 28,991), we made a fascinating discovery. Despite the diverse vocabulary used by reviewers to express their sentiments, we observed recurring patterns and similarities in the words utilized.

Additionally, we compared our custom RNN accuracy 77.1% with existing library-based implementations. Utilizing a single RNN model and a Bidirectional LSTM-based RNN model, we achieved accuracy rates of approximately 79.0% and 82.0% as Table II, respectively. These results highlight the significance of both the custom RNN model and the additional LSTM layer in sentiment analysis tasks.

Surprisingly, as we increased the size of the training data, the classification accuracy improved, resulting in more precise predictions. However, this progress came with the trade-off of longer training times as Fig. 6, Importantly, we observed a non-linear relationship between accuracy and computational time, with the model's complexity playing a crucial role in determining classification accuracy, while larger training datasets required higher computational resources.

In conclusion, our study underscores the importance of preprocessing techniques and the incorporation of word vectors in sentiment analysis tasks. The exceptional performance of the custom RNN model demonstrates its ability to discern meaningful patterns in review data. Furthermore, we emphasize the need for a careful balance when increasing the training data size, considering its impact on both accuracy and computation time. Armed with these insights, we can make informed decisions to optimize sentiment analysis outcomes in various applications.

TABLE II
Summary of our model RNN from scratch and from library

Scratch.RNN.LSTM	**Keras.RNN**	**Keras.RNN.LSTM**
77.1%	79.0%	82.0%

References

In this paper, we have used information from the following Kaggle website for reference. [2]–[5].

References

[1] Ramapuram, J. (2016, June 6). RNN backprop through time equations. Back Propaganda. https://jramapuram.github.io/ramblings/rnn-backrpop/

[2] Kaggle. (2020). Trip Advisor Hotel Reviews Dataset. Alam, M. H., Ryu, W.-J., Lee, S., 2016. Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. Information Sciences 339, 206–223. https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews

[3] Kaggle. (2020). Hotel Review Keras Classification Project. https://www.kaggle.com/code/lunamcbride24/hotel-review-keras-classification-project

[4] Kaggle. (2020). Covid19 tweets- Sentiment Prediction RNN [85% acc]. https://www.kaggle.com/code/shahraizanwar/covid19-tweets-sentiment-prediction-rnn-85-acc/notebook

[5] Github. (2019). rnn-from-scratch-1. https://github.com/ramnathv/rnn-from-scratch-1/tree/master