IMDb Sentiment Analysis (COMP3125 Individual Project)

John Grieco
School of Computing & Data Science

***Abstract*—This project is focused on comparing a neural network made from NumPy and Pandas, and another made from TensorFlow and Keras using IMDb sentiment dataset.**

***Keywords*—Machine Learning, Neural Network, Data Processing**

I. Introduction

The purpose of this project is to build a neural network and compare its effectiveness to packaged neural networks in python. This is interesting to me because it allows me to build a neural network to analyze IMDb review sentiment using NumPy and Pandas. This would allow me to learn more about python in a more complex project while also learning about the ins and outs of neural networks. There are a couple things I want to compare my network accuracy, weighted words, overall model performance, and noisy inputs against a model using TensorFlow and Keras, both packages in python used to make neural networks. There have been a lot of projects on this topic [1], but still doing it provides me with valuable experience in this field.

II. Datasets

A. Kaggle, IMdb Dataset of 50K Movie Reviews [2]

This dataset has collected 50 thousand reviews from the website *IMDb* and has marked them as positive or negative in the sentiment column. This was done by hand by the author to make sure the sentiment is accurate. This is a credible data source since it has been used many times by other people and reviewed multiple times.

B. Reviews, Sentiment

This dataset is a .csv format providing excel use, while keeping the size of the data set small, 66.21 MB. There are two columns in the dataset, reviews and sentiment. Each has 50 thousand reviews with no missing values. Each review has an associated value as "positive" or "negative." The reviews have everything associated from the original review provided. This means to be used the reviews need to be cleaned of special characters, and sentiment needs to be changed to 0 or 1 for logistic prediction.

III. Methodology

A. NumPy/Pandas Neural Network Method

The NumPy neural network method is a simple feedforward neural network implemented from scratch for binary sentiment classification. The input data is represented as a bag-of-words vector. The model consists of an input layer, a hidden layer with a hyperbolic tangent activation, and an output layer with a sigmoid activation to produce a probability between 0 and 1.

The network computes its output using the following equations:

$$Z_1 = XW_1 + b_1$$

$$A_1 = \tanh(Z_1)$$

$$Z_2 = A_1W_2 + b_2$$

$$A_2 = \sigma(Z_2)$$

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\left(A_{2,i}\right) + (1 - y_i)\log\left(1 - A_{2,i}\right)\right]$$

X is the input matrix, W_1 and W_2 are weight matrices, b_1 and b_2 are bias vectors, σ is the sigmoid function, and N is the number of samples.

The parameters are updated using gradient descent computed from the loss. The main assumptions are that sentiment can be learned from a bag-of-words representation and that a single hidden layer is sufficient for this task.

Advantages of this method include transparency and educational value, as every calculation step is explicit. Disadvantages are limited scalability and flexibility compared to deep learning frameworks, and the lack of advanced features such as embeddings.

This method was chosen to provide a transparent baseline. The implementation uses the NumPy module for all computations. Input cleaning and vectorization were performed manually. Hyperparameters such as hidden layer size and learning rate were chosen empirically.

B. TensorFlow/Keras Neural Network Method

The TensorFlow/Keras method applies a neural network model using the Keras API, with an embedding layer to learn word representations from raw text data, followed by a global average pooling layer and dense layers for classification.

The sequence of transformations is as follows:

1. The input text is tokenized and mapped to integer indices.
2. Each sequence of word indices is padded to a fixed length L.
3. The padded sequences S are passed through the embedding layer:
$$E = \text{Embedding}(S)$$
4. The embedding outputs are averaged:
$$G = \frac{1}{L}\sum_{j=1}^{L} E_j$$
5. The pooled vector G is passed through dense layers with activation functions, and finally through a sigmoid output:
$$y = \sigma(W_2 \cdot \text{ReLU}(W_1G + b_1) + b_2)$$

σ is the sigmoid function, ReLU is the rectified linear activation, and W_1, W_2, b_1, b_2 are the trainable parameters. The model is trained using binary cross-entropy loss and the Adam optimizer.

The main assumption is that semantic information captured by learned word embeddings and average pooling is sufficient for sentiment classification.

Advantages include efficiency, scalability, and the ability to easily use advanced features. Disadvantages are reduced transparency and reliance on external libraries.

This model was chosen for its effectiveness and rapid prototyping capabilities. The implementation uses the TensorFlow Keras API, specifically the Embedding, GlobalAveragePooling1D, and Dense layers. The Tokenizer and pad_sequences utilities were used for preprocessing. Hyperparameters such as embedding dimension, sequence length, batch size, and number of epochs were selected based on typical practices and adjusted for convergence.

IV. Results

A. *NumPy/Pandas Model*

This model is able to predict reviews with 86.748% accuracy. This means after being trained on the training reviews, it is able to predict the test data with said accuracy.

The following table shows the most influential words for both positive and negative reviews. When these words are encountered, they have the highest weight in predicting what the encountered review will be.

TABLE I. NUMPY/PANDAS

Rank	**Top Positive and Negative Words**			
	Positive Words	***Influence***	***Negative Words***	***Influence***
1	Excellent	0.71	Worst	-1.00
2	Great	0.67	Bad	-0.84
3	Best	0.54	Awful	-0.72
4	Wonderful	0.48	Boring	-0.71
5	Perfect	0.47	Waste	-0.71
6	Amazing	0.44	Poor	-0.60
7	Loved	0.43	Nothing	-0.60
8	Fun	0.39	Terrible	-0.56
9	Still	0.39	Stupid	-0.47
10	Favorite	0.37	Worse	-0.44

Fig. 1. This is the top ten positive and negative words using the NumPy/Pandas model.

This model uses "Bag of Words Count." This is a type of method used for machine learning where each word is placed into a vector. Then when reviews are tokenized, the method looks at the word vector and creates a new vector of all the counts of each word in reference to the original vector. An example of this is, there is a vector, ["movie", "good", "bad"]. This would be the starting vector after compiling all the words from the reviews. Then when tokenizing a review, "Good movie good." It would make a new vector as follows, [1, 2, 0]. It counts each word of the review, then passes it to the hidden layer to analyze and predict. However, another way of doing this is called "Binary Bag of Words." Instead of counting the same word multiple times, it will only count each word once even if there is more than one in the review. Using this new method, the model accuracy decreases slightly to 86.348%.

Finally, the model is able to handle noisy reviews with a 85.84% accuracy. Noisy reviews in this case means there is a 15% chance for each letter to be replaced with a different letter or symbol making the word unrecognizable to the model. This is a test to see how well model can handle something it doesn't recognize

B. *TensorFlow/Keras Model*

This model using TensorFlow/Keras is able to predict reviews at an 84.29% accuracy.

The following table shows the most influential positive and negative words. This uses a different scale than NumPy/Pandas. However, the words that are in the list a very close with only minor differences in position and words.

TABLE II. TENSORFLOW/KERAS

Rank	**Top Positive and Negative Words**			
	Positive Words	***Influence***	***Negative Words***	***Influence***
1	Great	6.47	Bad	-6.98
2	Excellent	4.85	Worst	-6.72
3	Perfect	4.15	Waste	-5.40
4	Best	3.92	Awful	-5.08
5	Wonderful	3.89	Terrible	-4.55
6	Love	3.69	Boring	-4.38
7	Loved	3.65	Horrible	-4.13
8	Amazing	3.38	Stupid	-4.02
9	Highly	3.14	Nothing	-3.89
10	7	3.09	Poor	-3.78

Fig. 2. This is the top ten positive and negative words using the TensorFlow/Keras model.

The first input of TensorFlow/Keras is different than NumPy/Pandas, so a direct comparison cannot be made. It is possible to directly compare "Binary Bag of Words." This model had a significantly lower accuracy at 49.93%.

This model had a much more difficult time with noisy reviews at a 49.24% accuracy at the same 15% chance of randomness within the reviews.

C. *Results C*

Both models using their respective initial inputs both had near accuracy. The model using TensorFlow/Keras has much more adaptability than NumPy/Pandas.

Fig. 1, Fig 2, The models had similar influential words tables not including the scale both models use.

Both models using the same input method have very similar accuracy.

The NumPy/Pandas model is able to handle noisy reviews much better than the TensorFlow/Keras model. This may be due to how both models handle their inputs to each other.

V. Discussion

If more time permitted, I could have done more thorough analysis of both models and tried different input methods to test both models. I would also like to spend more time exploring different ways to train the models with different batch sizes or pass throughs while avoiding overfitting.

VI. Conclusion

This project is a small example of how many large language models work. Doing this project gave me insight into the different algorithms and processes that are used in the world around us, on a much smaller scale than what is done by companies. Being able to understand that large language models are not truly AI, but algorithms used to predict responses using existing data.

Acknowledgment

Thank you, Professor Pang, for teaching this course.

References

[1] A. Mohamed, “Sentiment Analysis on IMDB Movie Reviews: A Beginner’s Guide,” *Medium*, Oct. 19, 2023. https://amustafa4983.medium.com/sentiment-analysis-on-imdb-movie-reviews-a-beginners-guide-d5136ec74e56 (accessed Aug. 09, 2025).

[2] “IMDB Dataset of 50K Movie Reviews,” *www.kaggle.com*. https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data