Binary Sentiment Classification

Hardik Prabhakar

24095038

Introduction

In order for a neural network to process sequential data and make different predictions based on them, we need to use different architectures other than the Dense layer. In this report, we will study and implement two of these architectures. They are:

1. Recurrent Neural Network (RNN)
2. Long Short-Term Memory (LSTM)

We will be performing sentiment analysis on the Amazon Reviews Dataset using these models.

Data Pre-Processing

- Due to an error in reading the input `.csv` file, a record was mistakenly used as the column index, which was promptly recombined along with the renaming of the column index.
- We check for null records in the dataset. After finding 207 of them, we drop them as they are negligible in front of the total number of records.
- We absorb the `title` column into the `review` column as they share the same sentiment. We also change the value of the polarities from 1 & 2 to 0 & 1.
- We preprocess the reviews by expanding word contractions, removing extra whitespaces and non-alphabetic characters and converting them to lowercase. We then break the sentence into tokens, remove stopwords i.e common words that don't impart meaning to a sentence, lemmatize the tokens, i.e convert them to their root dictionary forms, and then finally join the tokens together.
- Finally, we truncate the word count of the processed review to the 95th percentile to ensure reasonable amount of padding is done by excluding outliers.

Implementation

- We load a pretrained `GloVe` model of 200 dimensions and 400k vocabulary using the `gensim` library and create a helper function to encode a sentence. If a token is not present in the model, we assign zero vector to it.
- Using a custom PyTorch `DataSet` class, we utilize the helper function in the `__getitem__` definition to do word embeddings on the fly. This was done to ensure that RAM usage of the program doesn't exceed reasonable levels. Even after taking such precautions, the program occupied around 8-9 GBs of the RAM.
- For the `DataLoader` object, we set the batch size to be 256, balancing between the speed of completion of the training and ensuring that the GPU has enough VRAM to load in the tensors. We also define a custom `collate_fn` to pad the batch inputs as they have variable sequence lengths.
- We use similar architectures for both the vanilla-RNN and LSTM model, assigning 32 hidden states to them, which are fed into a `Dense` layer with one output neuron, representing the probability of the `polarity` being 1. We use `BCEWithLogitsLoss` as our loss function and `Adam` as our optimizer with default parameter. This was done because this optimizer performs better than vanilla `SGD` on RNN training.
- During the training loop, we pack our padded inputs into a `PackedSequence` object. This is done to optimize the training process, as this enables the model to ignore padding, avoiding unnecessary computations.

Evaluation

Performance Metrics

	precision	recall	f1-score
0	0.79	0.76	0.78
1	0.77	0.80	0.78
accuracy			0.78
macro avg	0.78	0.78	0.78
weighted avg	0.78	0.78	0.78

(a) Vanilla-RNN

	precision	recall	f1-score	support
0	0.88	0.95	0.91	200000
1	0.94	0.87	0.90	199999
accuracy			0.91	399999
macro avg	0.91	0.91	0.91	399999
weighted avg	0.91	0.91	0.91	399999

(b) LSTM

Figure 1: Metrics for both models

Confusion Matrix

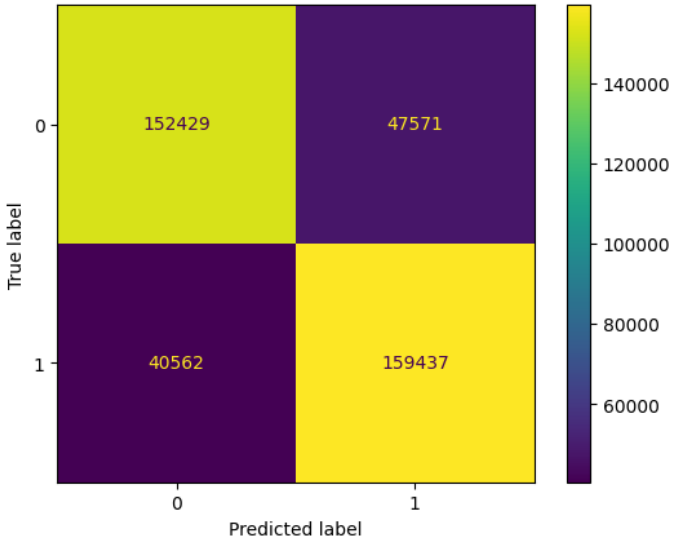


Figure 2: Vanilla-RNN

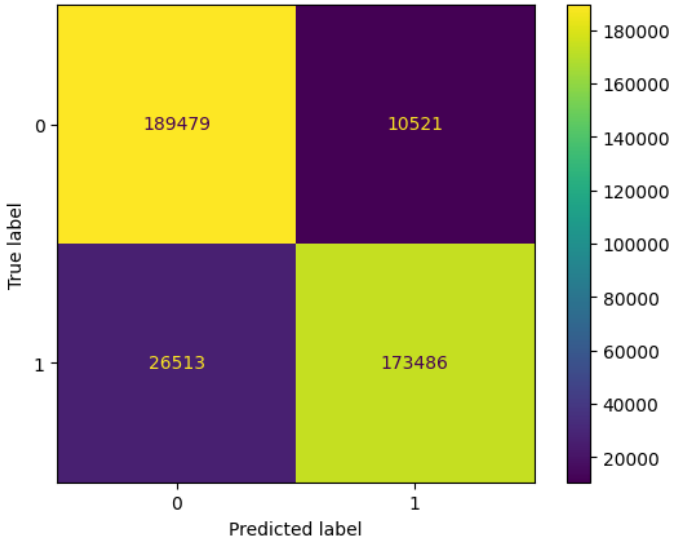


Figure 3: LSTM

Analysis

Analyzing the incorrect predictions made by our model, we can see that it makes errors when it encounters rare words or if it encounters a 'negative' word with a positive review and vice-versa. For example, one review read:

"Good looks but shabby quality. I was really excited to receive this pair of gorgeous sandals, but upon a few hours of wearing them, the links broke! Needless to say, I had to return them - very reluctantly."

The usage of words like *"good"* and *"gorgeous"* along with relatively rare words like *"shabby"* made our model classify this example as a false positive.

Bonus: Minimal Clue Challenge

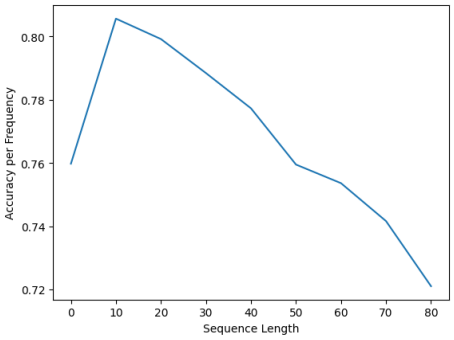
	polarity	review	processed_review
394679	0	.... This is not a vary good book. There are m...	vary good book informitive book one
373604	0	Debi. It works just as well as all of those di...	debi work well diet pill word work
91817	0	not happy. it's an demo and some of the show w...	happy demo show good would rate
111530	1	This book is not out of print. Call 1-800-777-...	book print call book second printing well
168895	0	Not his best. This book is not what it is hype...	best book hyped informative thought would

```
list(sample_short['review'])
```

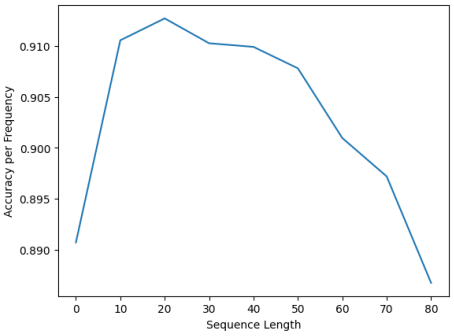
```
['.... This is not a vary good book. There are more informitive books out there. And this is not one of them.',
 'Debi. It works just as well as all of those diet pills out there. In other words, it does not work at all.',
 "not happy. it's an demo and some of the show was off. but it's was good. but wouldn't rate it at all.",
 'This book is not out of print. Call 1-800-777-2295. The book is now in its second printing and doing well!',
 'Not his best. This book is not what it is hyped to be, Not very informative and not what I thought it would be.']
```

- Looking at the processed versions of these short reviews, we can see that not much context or meaning can be inferred from them. The major loss of information comes from the removal of the word *"not"*, which is a stopword and filtered during preprocessing.
- In its current state, even a human would misclassify these examples. If stopwords that endow negative qualities weren't filtered, a human can recognize subtleties like sarcasm and classify the examples correctly.
- I would fix these issues by first, not letting stopwords like *"not"* be filtered. Then, for the model to pickup on subtle qualities, I would try increasing the number of hidden states so that more 'features' get captured.

Stretch Goals



(a) Vanilla-RNN



(b) LSTM

Analyzing the above two graphs, we can see that even though the accuracy of both architectures decreases with increase of sequence length, the rate of decrease is more in the vanilla-RNN model than the LSTM one. The better performance of LSTM is attributed to its ability to retain long-term information with the help of 2 hidden states.