Shelly Gamliely

Yerus Mandfro

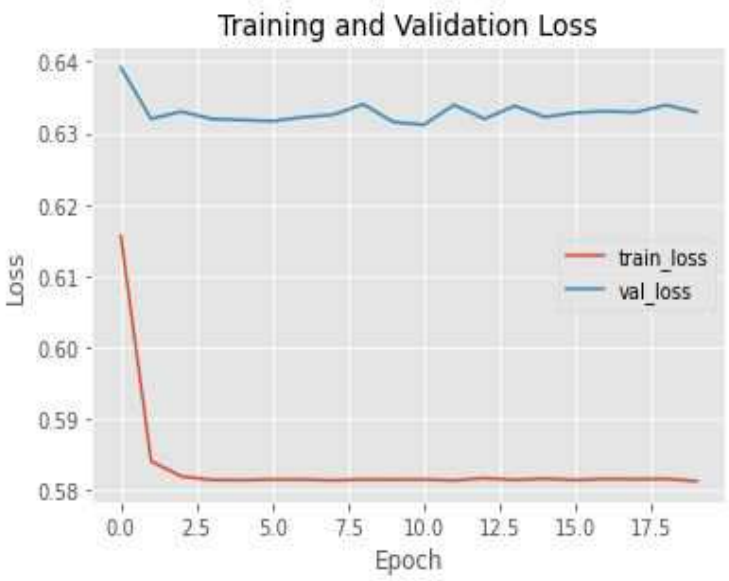
Nlp Ex4

Log-linear Model Results:

Test accuracy - 0.6671401515151515
Test loss - 0.6287311564069358

Negated Polarity Accuracy - 0.5645161290322581
Rare words Accuracy - 0.4

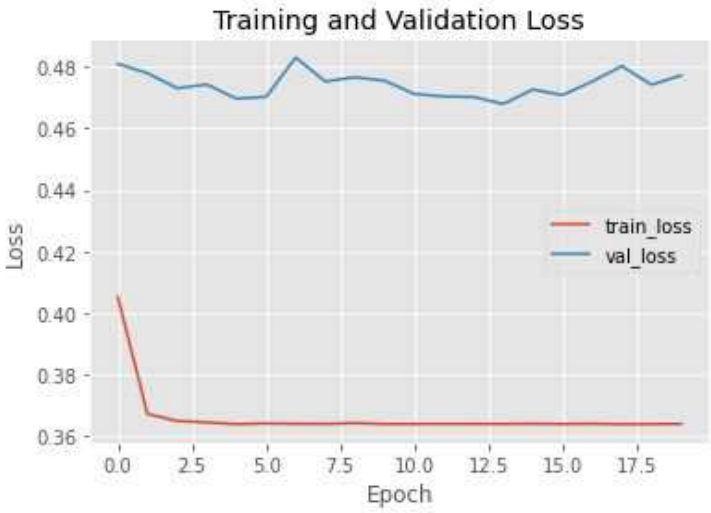
Train loss at step 19: 0.5812284664523939
Train accuracy at step 19: 0.7199480498721228
Validation loss at step 19: 0.6329312549167019
Validation accuracy at step 19: 0.673828125

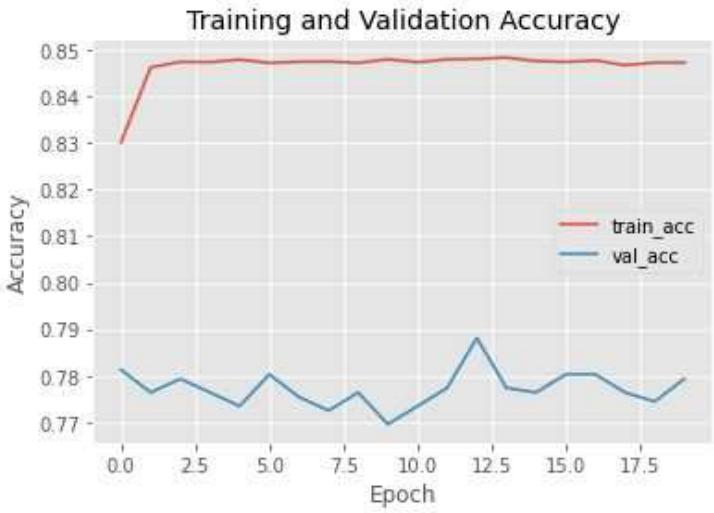


Word2Vec Log linear Model Results:

Test accuracy - 0.83203125
Test loss - 0.4129860132257997
Negated Polarity Accuracy - 0.5806451612903226
Rare words Accuracy - 0.78

Train loss at step 19: 0.36392043276893626
Train accuracy at step 19: 0.8473569373401535
Validation loss at step 19: 0.47727509229037357
Validation accuracy at step 19: 0.779296875





LSTM model Results:

We tried different values of the constant SEQ_LEN, as you can see at the table below we got different accuracies for each value:

Sequence length	Training Accuracy	Validation Accuracy	Test Accuracy	Negated Accuracy	Rare words Accuracy
20	0.899	0.782	0.859	0.597	0.78
32	0.906	0.788	0.867	0.629	0.84
52	0.905	0.811	0.86	0.71	0.82
64	0.906	0.825	0.862	0.677	0.8

The prediction of a model depends on the data set it is trained on, and so depends on sequence length which according to it we cut the sentence or pad it.
As mentioned in the exercise 'we must make sure that all sentences in the batch are of the same size' ,therefore there is a constant in the code which is SEQ_LEN. If the sequence length is long then we will pad short sentences so the vector will have many zeros. If the sequence length is short then we will cut long sentences and lose information.

The results that are shown below have SEQ_LEN = 52 that were given in the exercise.

Test accuracy - 0.8603515625
Test loss - 0.3571062292517848

Negated Polarity Accuracy - 0.7096774193548387
Rare words Accuracy - 0.82

Epoch 4:
Train loss at step 3: 0.2351364572024203
Train accuracy at step 3: 0.9052221867007674
Validation loss at step 3: 0.49627036357702003
Validation accuracy at step 3: 0.8115234375

Comparing the different models

	Test Accuracy	Test Loss	Validation Accuracy	Validation Loss	Negated Subsets Accuracy	Rare words Accuracy
Log-linear	0.667	0.629	0.674	0.6333	0.564	0.4
Word2Vec Log linear	0.832	0.413	0.779	0.477	0.58	0.78
LSTM	0.86	0.357	0.811	0.496	0.71	0.82

1.Compare the results (test accuracy, validation accuracy) you've received for the simple log-linear model, and the Word2Vec log-linear model. Which one performs better? Provide a possible explanation for the results you have.

Test Accuracy

- Log-linear model : 0.667
- Word2Vec Log-linear model : 0.832

Validation Accuracy

- Log-linear model : 0.674
- Word2Vec Log-linear model : 0.779

As we can see the Word2Vec model had better performances, it generalizes better when checking accuracy of sentences from the test set and validation set.

The reason that the Word2Vec model has higher accuracy, might be that it can capture contextual word-to-word relationships in a multidimensional space. The vectors that represent the words on the vocabulary are learned by understanding the context in which words appear. The result is vectors in which words with similar meanings end up with a similar numerical representation.

One-Hot Encoding is unable to recognize semantics of items, while embeddings reflect words similarity patterns by grouping commonly co-occurring items together in the representation space.

Notice that In one hot encoding representation we have a vector of zeros except for the element at the index representing the corresponding word in the vocabulary. Therefore all the words are independent of each other.
Using word2vec introduces some dependence of one word on the other words. The words in context of this word would get a greater share of this dependence.

2.Compare the latter results with the results of the LSTM model. Which one performs better? Provide an explanation for the results you received.

Test Accuracy

- Log-linear model : 0.667
- Word2Vec Log-linear model : 0.832
- LSTM model : 0.86

Validation Accuracy

- Log-linear model : 0.674
- Word2Vec Log-linear model : 0.779
- LSTM model : 0.811

As we can see, the LSTM model had the highest accuracy compared to the other models.

A possible explanation is that LSTM models can capture long-term dependencies between word sequences. LSTM learns to keep the relevant content of the sentence and forget the

non relevant ones based on training. The hidden state holds information on previous data the network has seen before. It can pass relevant information down the long chain of sequences to make predictions. The idea behind bi-directional networks is to capture information of surrounding inputs.

More specifically, in this task we are requested to predict whether a review is positive or negative. In order to make this decision we need to remember important keywords like : 'amazing' or 'boring'. Those words are more meaningful to understand whether a review is positive or negative than words like 'this' or 'in'.

3.Last, compare the results that all the models had on the 2 special subsets of sentences we've provided you. For each subset, state the model that has the highest result (and the lowest result) and provide a possible explanation for these results.

Negated Subsets

The model that had the higher results is LSTM
The model that had the lowest results is Log-linear

- Log-linear model : 0.564
- Word2Vec Log-linear model : 0.581
- LSTM model : 0.71

Rare words

The model that had the higher results is Word2Vec
The model that had the lowest results is Log-linear

- Log-linear model : 0.4
- Word2Vec Log-linear model : 0.78
- LSTM model : 0.82

Negated polarity - all sentences in the test set where the sentiment polarity of one of the main sub-phrases in the sentence is opposite of the whole sentence sentiment polarity.

The range of the words that are affected by negation words could be different from one sentence to another. For example, in the sentence "The movie was not interesting," the scope is only the next word after the negation word. But for sentences like "I do not call this film a comedy movie," the effect of the negation word "not" is until the end of the sentence. The original meaning of the words changes if a positive or negative word falls inside the scope of negation. Moreover, Negation can be explicit or implicit - meaning it carries a negative sentiment, but no negative words are used. Negation can also be morphological where it is either denoted by a prefix ("dis-", "non-") or a suffix ("-less").

Since LSTM remembers the history and learns to keep the relevant content of a sentence, it can identify better negation. In the example we gave it could identify the word 'interesting' as relevant and also the word 'not' so although 'interesting' is a positive word, the word 'not' before it changes the meaning of the whole sentence. Log- linear model that uses

one-hot encoder is unable to understand the context of words in sentence, and therefore has poor results.

Rare words - all test sentences ranked by their maximal frequency of a word which has non-neutral sentiment value ,we take the 50 sentences ranked lowest.

LSTM has the highest accuracy when using sentences with rare words.
Notice that Word2Vec has also high accuracy, and those results improved by Lstm that uses Word2Vec embedding.

Word2Vec is a pre-trained word embedding that can capture the semantic and syntactic meaning of a word as it was trained on large datasets. So Word2Vec generalizes better on rare words.
When using a one-hot encoder we have a problem of sparsity .Most real-world problems contain a dataset that has a large volume of rare words. The vectors learned from these datasets cannot arrive at the right representation of the word. This is why the log-linear model had poor results when testing on rare words.

Regarding the LSTM model as we mentioned before, it has the ability to choose the relevant words, so this could be the reason why the results improve a bit compared to Word2Vec alone.