Natural Language Processing – Exercise 3

Ori Dvir | Shir Rashkovits

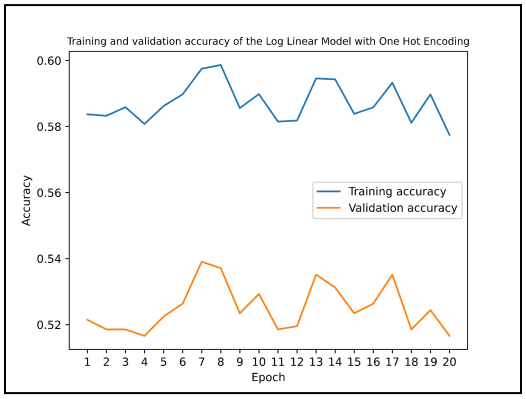
6.

a. A plot of the train loss value, as a function of the epoch number (and another similar one for the validation loss). Plot both curves on the same graph (one curve for each parameter).

b. A plot of the train accuracy value, as a function of the epoch number (and another, similar one, for the validation accuracy). Plot both curves on the same graph.

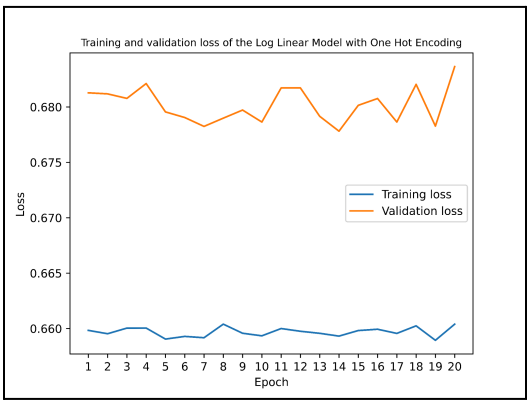
Answer:

accuracy



One hot encoding

loss



Please save and provide:

a. The test accuracy and loss.

b. The accuracy over each of the special subsets we've mentioned in section 1.

```
######################## Testing Log Linear Model with One Hot Encoding ########################

Test Loss: 0.6734647215780569
Test Accuracy: 0.5517578125
Negated Polarity Accuracy: 0.4838709677419355
Rare Words Accuracy: 0.28

######################## Testing Log Linear Model with One Hot Encoding ########################
```

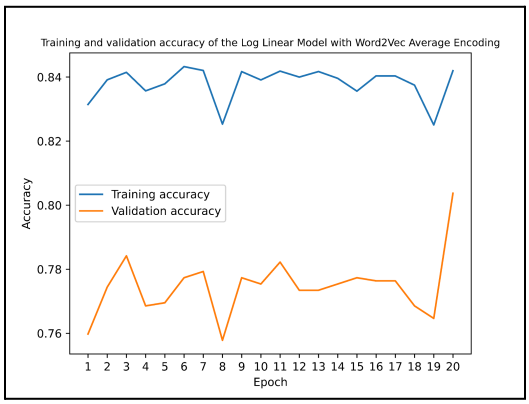
We can see that the simple log linear model using the one-hot encoding gets a bit better accuracy than random. Moreover, the received loss is big. In the next models we will see much better results.

7.

a. A plot of the train loss value, as a function of the epoch number (and another similar one for the validation loss). Plot both curves on the same graph (one curve for each parameter).

b. A plot of the train accuracy value, as a function of the epoch number (and another, similar one, for the validation accuracy). Plot both curves on the same graph.

Answer:



Word2Vec

Please save and provide:

a. The test accuracy and loss.

b. The accuracy over each of the special subsets we've mentioned in section 1.

```
######################### Testing Log Linear Model with Word2Vec Average Encoding ########################

Test Loss: 0.45478202777331944
Test Accuracy: 0.83203125
Negated Polarity Accuracy: 0.6129032258064516
Rare Words Accuracy: 0.72

######################### Testing Log Linear Model with Word2Vec Average Encoding ########################
```

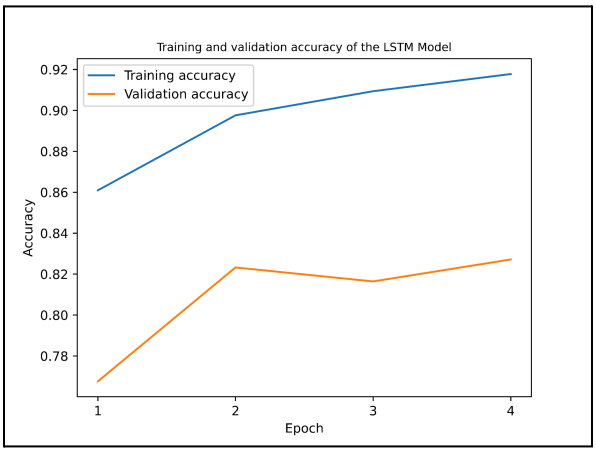
We can see that the enhanced log linear model using Word2Vec embeddings gets better results, with smaller loss and much bigger accuracy.

8.

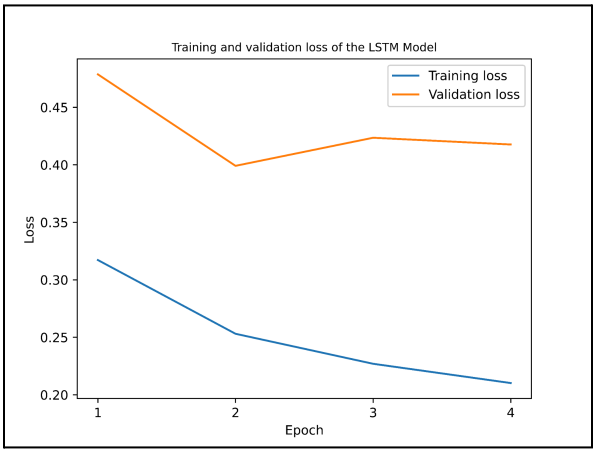
a. A plot of the train loss value, as a function of the epoch number (and another similar one for the validation loss). Plot both curves on the same graph (one curve for each parameter).

b. A plot of the train accuracy value, as a function of the epoch number (and another, similar one, for the validation accuracy). Plot both curves on the same graph.

Answer:



LSTM



Please save and provide:

a. The test accuracy and loss.

b. The accuracy over each of the special subsets we've mentioned in section 1.

```
####################### Testing LSTM Model #######################

Test Loss: 0.32441268750160823
Test Accuracy: 0.865234375
Negated Polarity Accuracy: 0.7258064516129032
Rare Words Accuracy: 0.8

####################### Testing LSTM Model #######################
```

We can see further improvements in the loss and accuracy of the model compared to previous models.

Note that in all the models implemented, we can also see that the models give better results (smaller loss, bigger accuracy) on the training set compared to the validation & test set, since the training is guided by the samples it sees, and although try to generalize, it is still limited to the samples it sees in the training.

Comparing the different models:

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.

Answer:

Log linear model validation accuracy using one-hot encoding: about 0.53
Log linear model test accuracy using one-hot encoding: 0.55

Validation accuracy using Word2Vec: about 0.78
Test accuracy using Word2Vec: 0.83

When comparing the performance of the simple one-hot encoding log-linear model against the log-linear model with Word2Vec embeddings, the latter gets better results both in test and validation, with bigger accuracy. This improvement can be attributed to several factors related to the Word2Vec embeddings:

- Semantic Representation: Word2Vec embeddings are designed to capture the semantic relationships between words. Each embedding vector represents not just the presence of a word but its context and meaning relative to other words. This rich semantic representation allows the model to better understand the nuances of language, facilitating more accurate sentiment analysis.
- Dimensionality and Sparsity: Traditional one-hot encoding used in simple log-linear models leads to high-dimensional (in the size of the vocabulary, in the exercise=16271), sparse representations where most elements are zero. This sparsity can hinder the model's ability to learn effectively from the training data. In contrast, Word2Vec embeddings provide a more compact (in the exercise=300), dense representation that focuses on capturing meaningful attributes of words. Despite being lower-dimensional, these embeddings retain crucial information necessary for understanding sentiment, thereby reducing the negative impact of sparsity.
- Generalization: The ability of Word2Vec embeddings to represent semantic similarity allows the model to generalize better to unseen words or phrases. If a word did not appear frequently in the training data but is semantically similar to other words that did, the model can leverage this semantic relationship to make more accurate predictions. This is particularly beneficial in the context of sentiment analysis, where the sentiment conveyed by a text can often be inferred from the overall semantic context rather than specific word occurrences.

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

Answer:

LSTM validation accuracy using Word2Vec: about 0.81
LSTM test accuracy using Word2Vec: 0.87

The LSTM model performs better compared to both the simple log-linear and the Word2Vec-enhanced log-linear models in terms of both test and validation accuracy. This performance can be attributed to several advantages of the LSTM architecture:
- Temporal Dynamics: Unlike log-linear models, LSTMs are specifically designed to capture temporal dependencies in sequential data. This allows the LSTM to understand the order and context in which words appear, providing a more nuanced analysis of sentiment based on the sequence of the text.
Also, LSTMs can remember information over long sequences. This capability enables the model to make connections between words that are far apart in the text, capturing the overall sentiment more effectively. This temporal dynamics is in contrast to the log linear model which gets as input the average embedding of the whole sentence, which leads to a loss of the order of the words. For example, there is a big difference between the sentence: “I thought the movie will be great, but it was a disappointment” and “I thought the movie will be a disappointment, but it was great”. For the log linear model, these sentences get the same input representation and thus the same output, although clearly their sentiments are different.
- Bidirectional: the bidirectional architecture of the LSTM model significantly enhances its ability to comprehend the sentiment conveyed in a sentence. This is because sentiment can be nuanced and contextually dependent on information both before and after a given point in the sentence. By processing the sequence in both forward and backward directions, a bidirectional LSTM can capture dependencies and subtleties that might be missed by a unidirectional approach. This dual analysis offers a more comprehensive understanding of the sentence structure and sentiment, providing a clear advantage over the log-linear model that lacks this capability.

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.

Answer:

As seen above, the best results on both special sets are achieved by the LSTM model with:
Rare words set accuracy = 0.8
Negated Polarity set accuracy = 0.73
And the worst results are achieved by the one-hot encoding log linear model with:
Rare words set accuracy = 0.48
Negated Polarity set accuracy = 0.28

As we can see in the results above, the log-linear model with average one-hot encoding did a bit worse than chance on the rare words set, while the LSTM model with Word2Vec embeddings did much better. On one hand, this could be explained by sparsity of the one-hot encoding method, where different words and sentence can receive vastly different encoding, and on the other by the fact that the Word2Vec embeddings are able to catch semantic similarity between different words, which causes unseen sentences to have similar encodings to seen sentences if they have similar semantic structure, which in turn helps the model to generalize and predict more accurately on unseen sentences.

As for the results on the negated polarity sentences, we can see that the log-linear model with one-hot encoding did much worse than chance on this subset of sentences, whereas the LSTM model with Word2Vec embedding did much better. This could be explained by the fact that the LSTM model performs its analysis of the sentence sequentially, , i.e it receives the embedding of each word in the sentence as different input in different stages of the model prediction operation, thus the model can capture the change in polarity in different stages of the sentence. In contrast to that, the log-linear models (both with one-hot encoding and with Word2Vec embedding) performs their analysis of the sentence in bulk, i.e it receives a single representation of the sentence by averaging on the word's different encodings. Because of that the polarity of each word and each subset of the sentence is combined to a single representation, which makes it harder for the model to identify the change.