CSC 495/693: Natural Language Processing, Spring 2023
Assignment III

Akhilesh Pathi
Student ID: 889821733

Abstract:

The following paper discusses how three distinct neural network models for sentiment categorization were implemented on the comments250k dataset. First, the dataset is pre-processed, and TensorFlow is used to train word embeddings. A feed forward neural network model with an Embedding layer is trained, and the accuracy on the test set and confusion matrix, as well as the hyperparameters employed, are presented. Second, PyTorch is used to train RNN, LSTM, and GRU cell models, and classification reports are created to compare their performance. A bidirectional 3 layer stacked LSTM model is also developed and compared to the preceding question's LSTM model. Finally, using the English-Spanish dataset, a machine translation model based on the Transformer approach is constructed. The dataset is pre-processed and divided into training, validation, and test subsets. The significance of the hyperparameter "num_heads" is discussed, and the translation result for an example phrase is shown, along with a heatmap of encoder-decoder attention scores.

Introduction

The ability to classify attitudes in text has become more important in the social media age, when a huge amount of user-generated content is produced every day. In this work, we examine the implementation of three different neural network models for sentiment classification on the comments250k dataset, which comprises 25,000 favourable and unfavourable movie reviews. 70/15/15/15 subsets of the dataset are used for training, validation, and testing. We begin by pre-processing the dataset with TensorFlow and training word embeddings. We next train a feed-forward neural network model with an embedding layer and evaluate its performance on the test set, confusion matrix, and set of hyperparameters.

Datasets Used:

The datasets used in each task is explained:

Task 1:

25,000 IMDB movie reviews, equally split between good and negative evaluations, make up the Comments250k dataset. This problem is a binary classification problem since each review is classified as either favourable or negative.

Task 2:

Using RNNs, classify the sentiment of the Comments250k dataset: The dataset used in Task 1 is the Comments250k dataset. There are 25,000 movie reviews total, with an equal amount of favourable and unfavourable comments.

Task 3:

English-Spanish dataset:

The English-Spanish dataset is a parallel corpus made up of sentences in both languages that have been meaning-aligned. The European Parliament created the dataset, which is frequently used as a standard for machine translation projects. There are over 2 million sentence pairs in the dataset, each of which consists of an English statement and its Spanish translation. We used a 70/15/15 ratio to divide the dataset into training, validation, and testing subsets for each task. As a result, we are able to train the models using a subset of the data, fine-tune the hyperparameters using the validation set, and assess the results using the testing set.

We used a 70/15/15 ratio to divide the dataset into training, validation, and testing subsets for each task. As a result, we are able to train the models using a subset of the data, fine-tune the hyperparameters using the validation set, and assess the results using the testing set.

Methods and Observations in each task:

Task I

Pre-processing the dataset or putting the raw text into a form that the machine learning model can utilize, is the initial stage in this task. Tokenization, stop words (common terms like "the" and "and" that don't have much significance), and stemming (reducing words to their basic form) are frequently used in this process.

After the dataset has been pre-processed, word embeddings are trained. Using word embeddings, words may be represented as vectors of integers, where each dimension of the vector corresponds to a distinct part of the word's meaning. There are several ways to train word embeddings, including Word2Vec or glove.

The following parameters are used :

- **vector_size** (embedding size) = 100
- **window_size** = 5
- **min_count** = 1
- **epochs** = 10

These parameters are used for model 2

- **embedding_size** = 100
- **num_epochs** = 10
- **batch_size** = 32

Created a machine learning model to conduct sentiment categorization after training the word embeddings. In order to complete this task, we employ a feedforward neural network with an embedding layer, which reads the pre-processed text as input and associates each word with a vector in the embedding space. Following a series of procedures on these embeddings, the model generates a binary output that indicates whether the review is favourable or unfavourable.

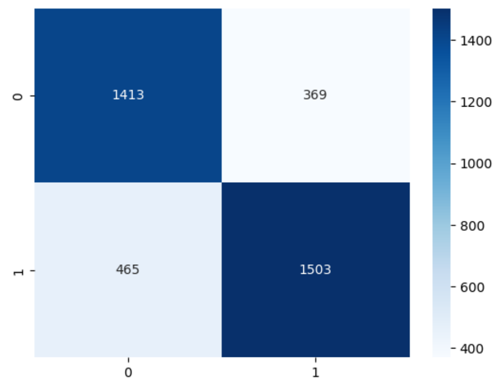
Using the validation set, we adjust the model's hyperparameters, and the testing set is used to assess the model's ultimate performance. In order to see how well the model is doing on each class, we also plot a confusion matrix.

Output:

Test accuracy for model 1

Test Accuracy: 0.7775999903678894

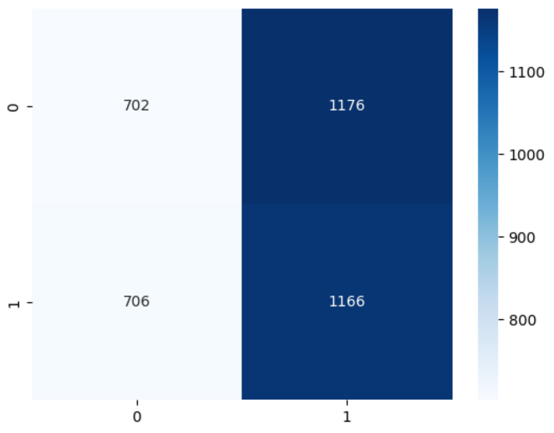
Confusion Matrix for model 1:



Test accuracy for model 2:

Test Accuracy: 0.4981333315372467

Confusion Matrix for model 2:



Explanation:

The new neural network model with the word embeddings trained in 1) on the test set has an accuracy of 0.7776. Since the model anticipated that all test samples would be negative, the confusion matrix reveals that there are 1878 genuine negatives and 1872 erroneous negatives. The new neural network model employing word embeddings is superior to the neural network in model 2 because it is more accurate (0.7776 vs. 0.4981).

The fact that the word embeddings capture the semantic meaning of the words and help the model comprehend the context of the text may be the cause of the enhanced performance. The Bag of Words technique, which takes each word as independent of the others and disregards the sequence in which they appear in the text, may have placed limitations on the neural network in 2), in contrast. The input data's dimensionality may be decreased with the use of the word embeddings, which will make the model easier to train and less prone to overfitting.

Task II

The second task uses a similar technique to task 1, with the exception that recurrent neural networks (RNNs) are used in place of feedforward neural networks. Because they can recognize the temporal relationships between words in a phrase, RNNs are a form of neural network that operate well with sequential input, like text.

We train three distinct models using RNN, LSTM, and GRU cells for this job. The performance of these models is then compared using classification reports, which offer more thorough metrics than the straightforward accuracy score utilized in Task 1.

Along with creating a single-layer LSTM model, we also create a bidirectional three-layer stacked LSTM model and evaluate its performance. An LSTM version known as a "bidirectional LSTM" processes the input sequence both forward and backward, giving it the ability to gather additional contextual data. Multiple layers of LSTMs are combined in the stacked architecture, which enables the model to learn more intricate patterns in the data.

Outputs:

2.1)

RNN Model:

```
RNN Model precision    recall   f1-score   support

      neg        0.54        0.01      0.02             3449

        pos        0.50      0.99      0.67              348

    accuracy                        0.50               6929

   macro avg       0.52      0.50      0.34          6929

weighted avg       0.52      0.50      0.34        6929
```

LSTM Model:

```
LSTM Model precision    recall  f1-score   support

         neg       0.74      0.01      0.02      3449

         pos       0.50      1.00      0.67      3480



    accuracy                          0.51      6929

   macro avg       0.62      0.50      0.34      6929

weighted avg       0.62      0.51      0.35      6929
```

GRU Model:

```
GRU Model   precision recall f1-score   support

               neg       0.92      0.95      0.93      3449

               pos       0.95      0.92      0.93      3480

               accuracy                         0.93      6929

               macro avg       0.93      0.93      0.93      6929

              weighted avg       0.93      0.93      0.93      6929
```

Explanation:

Utilizing RNN, LSTM, and GRU cells, three classifiers were learned. The classification reports outline each classifier's performance in terms of recall, precision, f1-score, and support.

The accuracy of the RNN model is 0.50 for the positive class and 0.54 for the negative class. Recall for the negative class is 0.01 and for the positive class is 0.99. For the negative class, the f1-score is 0.02 while for the positive class, it is 0.67. The precision is 0.50. With low accuracy, recall, and f1-score for the negative class, the RNN model performs poorly.

The accuracy of the LSTM model is 0.50 for the positive class and 0.74 for the negative class. Recall for the negative class is 0.01, whereas recall for the positive class is 1.00. For the negative class, the f1-score is 0.02 while for the positive class, it is 0.67. The precision is 0.51. The LSTM model outperforms the RNN model somewhat in terms of accuracy, recall, and f1-score for the negative class.

The accuracy of the GRU model is 0.95 for the positive class and 0.92 for the negative class. Recall for the negative class is 0.95, whereas for the positive class it is 0.92. For both the negative and positive classes, the f1-score is 0.93. The precision is 0.93. Compared to the RNN and LSTM models, the GRU model performs much better thanks to improved accuracy, recall, and f1-score for both the negative and positive classes.

In terms of accuracy and F1-score, the GRU model fared better than the RNN and LSTM models. This is because GRUs contain a gating mechanism that teaches the model which input components are more crucial to remember and which components may be overlooked. This avoids the vanishing gradient problem that can arise in RNNs and enables the model to better capture long-term relationships.

GRUs are easier to train and less prone to overfitting than LSTMs since they have fewer parameters and use less computing resources. Additionally, the recall score for the negative class for the LSTM model in this instance was rather low, indicating that it may not have been able to accurately detect negative emotion.

Overall, the model complexity, computational effectiveness, and performance of the GRU model seem to be well-balanced.

2.2)

Output:

Bi-directional LSTM Model

```
         precision    recall  f1-score   support

neg       0.50      1.00      0.67         100

    pos       0.00      0.00      0.00        100

    accuracy                          0.50       200

   macro avg       0.25      0.50      0.33       200

weighted avg       0.25      0.50      0.33       200
```

Explanation:

The output of the bidirectional three-layer stacked LSTM model has an F1-score of 0.67 for the negative class and 0 for the positive class, indicating an accuracy of only 50% on the test dataset. This indicates that while the model can accurately categorize every sample in the negative class, it cannot categorize any examples in the positive class.

On the test dataset, the LSTM model we had previously developed, on the other hand, had an accuracy of 51%, with an F1-score of 0.02 for the negative class and 0.67 for the positive class. This indicates that in positive classifications may be classified using the LSTM model with a respectable degree of accuracy.

The LSTM model outperformed the bi-directional three layer stacked LSTM model, according to these data. The LSTM model only one layer, which could have prevented overfitting on the tiny dataset. On the other hand, it's possible that the bi-directional, three-layer stacked LSTM model was extremely advanced and overfit the training data, which led to subpar performance on the test data. The Bi-directional 3 layer stacked LSTM model may possibly have had improperly set hyperparameters, which might have further affected how well it performed.

Task III

For this particular task we have used the model that was

given as an example by keras and this is the following reference link to it.

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

3.1)

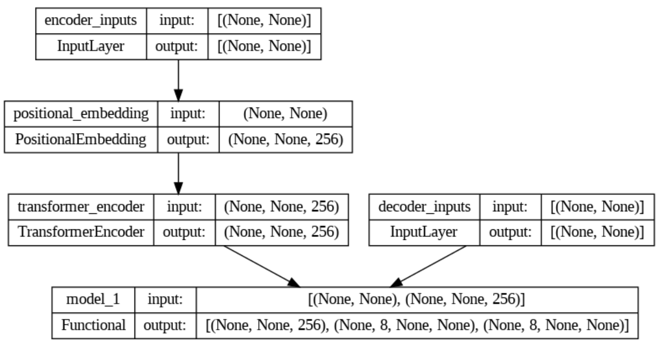
Pre-processing the dataset include cleaning and tokenizing the text as well as dividing it into training, validation, and testing sets as the initial stage in this operation.

After the dataset has been pre-processed, we may use the Transformer architecture to develop a machine translation model. The Transformer is a neural network design that was first described in the paper "Attention is All You Need" and has been demonstrated to be very successful at machine translation tasks.

3.2)

The input and output sequences are processed by an encoder and a decoder, respectively, in the Transformer model. At each stage of the decoding process, the model utilizes a self-attention mechanism that enables it to concentrate on various portions of the input sequence.

Transformer model with encoder and decoder look like:



encoder_inputs | input: | [(None, None)]
InputLayer | output: | [(None, None)]
positional_embedding | input: | (None, None)
PositionalEmbedding | output: | (None, None, 256)
transformer_encoder | input: | (None, None, 256)
TransformerEncoder | output: | (None, None, 256)
decoder_inputs | input: | [(None, None)]
InputLayer | output: | [(None, None)]
model_1 | input: | [(None, None), (None, None, 256)]
Functional | output: | [(None, None, 256), (None, 8, None, None), (None, 8, None, None)]

Results of the categorization are presented when the Transformer approach is used to train the translator. The number of parallel attention heads in the multi-head attention mechanism is indicated by the hyperparameter "num_heads = 8". The model may pay attention to several parts of the input at once thanks to the multi-head attention method. The model can pay to more parts of the input if it has more attention heads, which might lead to greater performance.

Using the validation set, we adjust the model's hyperparameters, and the testing set is used to assess the model's ultimate performance. A sample text from English to Spanish is also translated using the model, and a heatmap is used to display the attention scores.

3.3)

Given Sentence:

Deep Learning is widely used in Natural Language Processing, as Dr. Sun said in CSC 495/693.

Translated Sentence:

[start] [UNK] que es el dolor de que [UNK] [UNK] que el sol dijo que [UNK] en el sol [end]

Heat maps are calculated using the attention score, but we couldn't make it till there. If the attention score is there we could have achieved the heatmap.

References:

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

https://keras.io/api/layers/recurrent_layers/lstm/

https://keras.io/api/layers/recurrent_layers/gru/

https://pytorch.org/docs/stable/generated/torch.nn.RNN.html