Individual Project Report-Group 5

Siva Gogineni

Natural Language Processing

DATS 6312

May 6, 2023

Movie Reviews sentiment identification

i.

1.Introduction:

In today's world, we all know how powerful the feedback provided by consumers. This applies to any area but in this project, I focused on Movie reviews sentiment identification. Movie reviews sentiment identification is a process that involves analyzing the text of movie reviews and identifying the sentiment expressed in them. The sentiment can be positive, negative, or neutral and reflects the reviewer's overall opinion of the movie. Sentiment identification can be done using natural language processing (NLP) techniques, which involve analyzing the language and structure of the text to determine the sentiment. This process can be useful for filmmakers, studios, and movie enthusiasts to gauge the general reception of a movie and to identify areas that may need improvement. By understanding the sentiment of movie reviews, studios and filmmakers can make more informed decisions about marketing and promoting their movies and can improve the overall quality of their productions.

NLP have various techniques to process the text data and for this project we have used the LSTM technique to build the model which can perform the sentiment analysis.

In order to build this model, I have used the text data from Kaggle dataset, which we used to train and test the model.

In this project, also added logic that converts data in the numerical vector format to original text format.

2.Description of Data Set:

I have used the dataset from Kaggle, which have the information on over 50,000 movie reviews. each of the row have movie review and sentiment information. Dataset split into training and test datasets. Training dataset have 37500 records and testing dataset have 12500. And, to minimize the computation time, I have selected first 1000 frequently occurring tokens.

3.Description of NLP Network and Training Algorithm:

NLP (Natural Language Processing) networks are artificial neural networks designed to process and analyze natural language data, such as text, speech, and human-generated language inputs. One of the popular NLP network architectures is the LSTM (Long Short-Term Memory) network, which is a type of recurrent neural network (RNN) that can process sequential data and handle long-term dependencies in the data.

During training, the LSTM network learns to adjust the weights of its parameters, such as the weights of its gates, based on the input data and the desired output. The training algorithm used for LSTM networks is backpropagation through time (BPTT), which is a variant of the backpropagation algorithm used for training recurrent neural networks.

In the BPTT algorithm, the LSTM network is unrolled over time, and the error gradients are calculated at each time step. The gradients are then propagated backward through time to adjust the weights of the network. This process is repeated for multiple epochs until the network's performance on the training data reaches a satisfactory level.

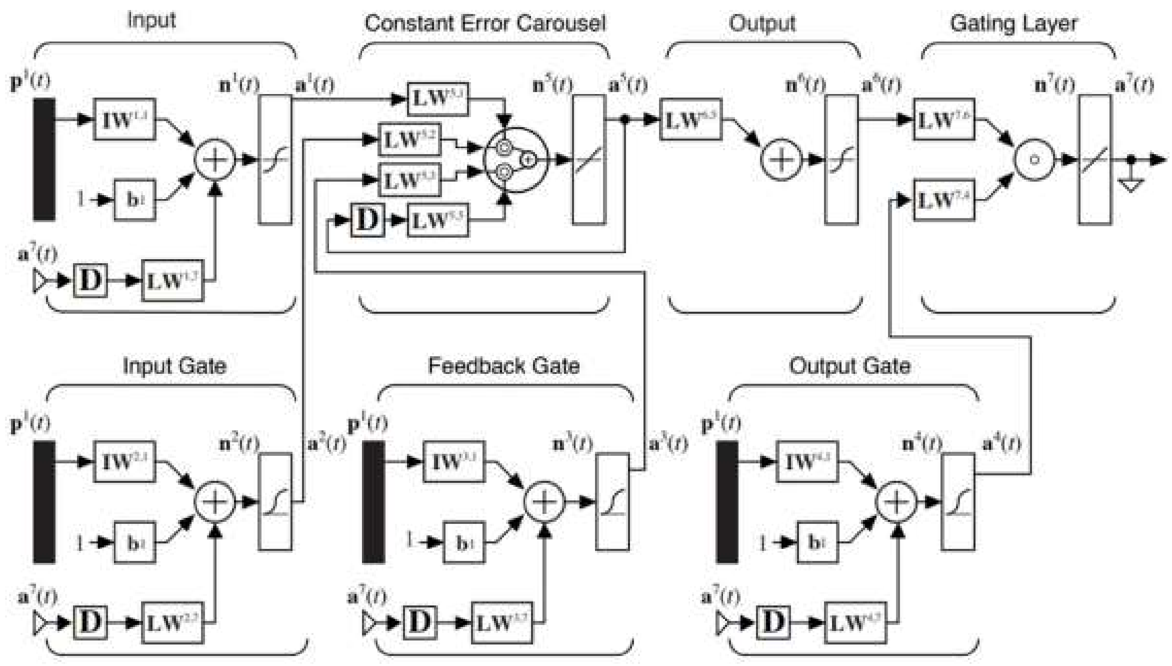
Following are the components of the BPTT algorithm:

1. Forward Pass: In the forward pass, the input sequence is fed into the RNN one timestep at a time, and the output is generated for each timestep. The input sequence is typically represented as a sequence of vectors or embeddings, and the output can be a scalar value or a sequence of vectors.
2. Loss Function: The loss function is used to measure the difference between the predicted output of the RNN and the true output. The loss function can be customized based on the specific task, such as mean squared error for regression tasks or cross-entropy for classification tasks.
3. Backward Pass: In the backward pass, the error gradients are calculated for each timestep based on the difference between the predicted output and the true output. The gradients are then propagated backward through time using the chain rule of calculus to compute the gradients of the weights and biases of the RNN.
4. Weight Updates: After the gradients are computed, the weights and biases of the RNN are updated using an optimization algorithm, such as Stochastic Gradient Descent (SGD) or Adam. The weights and biases are updated in the opposite direction of the gradients to minimize the loss function.

Repeat: Steps 1-4 are repeated for multiple epochs until the RNN's performance on the training data reaches a satisfactory level. At each epoch, the RNN is trained on a different random batch of the training data to prevent overfitting and improve generalization.

The BPTT algorithm is computationally intensive due to the need to store and compute the gradients for each timestep, which can result in vanishing or exploding gradients. To mitigate this issue, techniques such as gradient clipping or using variants of RNNs, such as LSTMs or Gated Recurrent Units (GRUs), are often used.

4.Architecture of the LSTM:



• In this architecture, the input text is first converted into a sequence of word embeddings using a pre-trained embedding model. These embeddings are then fed into an LSTM layer, which processes the input sequence and produces a fixed-length vector representation of the input text. This vector representation is then fed into a fully connected output layer, which produces a prediction for the class label.

• The model is trained using a dataset of labeled text examples, and the weights of the LSTM and output layers are adjusted using backpropagation and gradient descent. The accuracy of the model can be evaluated using a separate test set of labeled examples.

• Overall, developing an LSTM-based text classification model requires careful selection of pre-trained embedding models, optimization of hyperparameters such as the number of LSTM layers and hidden units, and careful tuning of training parameters such as the learning rate and batch size.

5. Experimental Setup:

The data from Kaggle split into train and test datasets. These two data set splits were used in our experiment. The training set was used to update the parameter weights in the different model layers after passing the inputs through the model and calculating the token-wise cross-entropy loss compared to the target. The validation set was used to evaluate how effectively the model was learning during each epoch of training by also calculating cross-entropy loss on data not explicitly used for training. The test data set was only used after the model had completed all of its training.

6. Accuracy Metric:

Accuracy is a commonly used metric in natural language processing (NLP) to measure the performance of a model in predicting the correct class for a given text document. In NLP, accuracy is defined as the ratio of the number of correctly classified examples to the total number of examples in the dataset.

Accuracy Metric, which is somewhat similar to computing the F1-score, are used to measure the performance of the model in NLP. It used to measure the performance at each epoch. When accuracy is higher at certain epochs and degrading it in subsequent epochs. Will stop the training model and start using model for predictions.

7. Optimizations:

To improve the base model's performance, there are various parameters and hyperparameters to tune but some of the parameters like weight and bias are adjusted by algorithm itself during the model training and other hyper parameters and parameters like below ones may need to be tuned based on the performance of the model.

- no_layers
- embedding_dim
- output_dim
- hidden_dim
- batch_size

Some of the observations we have here that improving the embedding_dim,layers helping to improve the model performance and leading to over fitting on the expense of model complexity. So, it is required to maintain balance between underfitting and overfitting.

As systems have limitations to process all of the input data in one pass, used the Data loader utility to process the data in batches, which will help to free up some of the system resources.

8.Result:

Optimized hyper-parameters and parameters	
embedding_dim	**120**
output_dim	**1**
hidden_dim	**34**

batch_size	**60**
no_layers	**8**
Epochs	**10**
Learning rate	**0.001**

Sample Output:

```
Epoch 1
train_loss : 0.5471346073627472 val_loss : 0.39473757477333915
train_accuracy : 0.7099466666666666 val_accuracy : 0.82072
Epoch 2
train_loss : 0.3640967009782791 val_loss : 0.35100717119012886
train_accuracy : 0.8460266666666667 val_accuracy : 0.84712
Epoch 3
train_loss : 0.32484233593940737 val_loss : 0.3392997787405665
train_accuracy : 0.8632533333333333 val_accuracy : 0.8536
Epoch 4
train_loss : 0.3014868754148483 val_loss : 0.3481324433993835
train_accuracy : 0.8765066666666667 val_accuracy : 0.85304
Epoch 5
train_loss : 0.2813331615924835 val_loss : 0.3334555452546248
train_accuracy : 0.8833866666666667 val_accuracy : 0.85976
Epoch 6
train_loss : 0.2587001905798912 val_loss : 0.3452791302249982
train_accuracy : 0.89584 val_accuracy : 0.85496
Epoch 7
```

```
train_loss : 0.23782333887219428 val_loss : 0.34611357347323346

train_accuracy : 0.9064533333333333 val_accuracy : 0.84912

Epoch 8

train_loss : 0.2177919050514698 val_loss : 0.37928623566403985

train_accuracy : 0.9162666666666667 val_accuracy : 0.856

Epoch 9

train_loss : 0.1935856711924076 val_loss : 0.4068276945215005

train_accuracy : 0.9268266666666667 val_accuracy : 0.84984

Epoch 10

train_loss : 0.16863005894720554 val_loss : 0.4388680230253018

train_accuracy : 0.9401333333333334 val_accuracy : 0.84464


Process finished with exit code 0
```

9. Summary and Conclusions:

In this project, I successfully added logic that can convert numerical vector to original text and build model that can identify the audience feelings about movie review. Model I developed predicting the output with 86% accuracy. Because of execution time constraints, I couldn't able to train and test model with wide range of values of hyper parameters. However, If we can use wide range of values for hyper parameters using grid search, we certainly improve the performance of the model further. And also, I observed that model is little bit overfitting as training accuracy is going up in each epoch but not the validation accuracy. We can certainly enhance this model by taking steps to reduce the complexity of the model, regularization of model, improve the quality of the training dataset and increase the sizing of the training dataset.

References:

1. Class Notes by Amir Jafari.

2. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron:

https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/

3. Natural language processing - Wikipedia