

[Return to Classroom](#)

Deploying a Sentiment Analysis Model

REVIEW

CODE REVIEW **3**

HISTORY

Meets Specifications

Dear Learner,

Well done. You have deployed the model in AWS by constructing all required components. .py files are updated with correct expressions and URL updated in html file.

- Tested the deployed model and it performed well
- You have gained an understanding on how to deploy a trained model which can be consumed by users. It is an important steps in the learning process and as a Data scientist , after training a model you would also need to deploy it for larger user consumption.

Continue in this space and understand various deployment options and process involved.

All the best

Files Submitted



The submission includes all required files, including notebook, python scripts and html files.

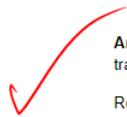
All required files submitted as part of the attachment

Preparing and Processing Data



Answer describes what the pre-processing method does to a review.

Well done. You have identified all the functionalities of the pre-processing method and listed down.



Answer: This method will reduce the Word token by converting multiple words which holds same information in to a single word. so that it will reduce the training cost too. Additionally

Remove HTML and any XML tags. Convert all the words to lower case and remove unnecessary characters by allowing only chracters and numbers. finally it Removes stopwords.



The `build_dict` method is implemented and constructs a valid word dictionary.

Good work. Constructed the function with correct expressions.

```
def build_dict(data, vocab_size = 5000):  
    """Construct and return a dictionary mapping each of the most frequently appearing words to a unique integer."""  
  
    # TODO: Determine how often each word appears in `data`. Note that `data` is a list of sentences and that a  
    #         sentence is a list of words.  
  
    word_count = {} # A dict storing the words that appear in the reviews along with how often they occur  
    for review in data: # Looping through the list of reviews  
        for word in review: # Looping through the list of words per review  
            if word not in word_count:  
                word_count[word] = 1  
            else:  
                word_count[word] += 1  
  
    # TODO: Sort the words found in `data` so that sorted_words[0] is the most frequently appearing word and  
    #         sorted_words[-1] is the least frequently appearing word.  
  
    sorted_words = sorted(word_count, key=word_count.get, reverse=True)  
  
    word_dict = {} # This is what we are building, a dictionary that translates words into integers  
    for idx, word in enumerate(sorted_words[:vocab_size - 2]): # The -2 is so that we save room for the 'no word'  
        word_dict[word] = idx + 2                                # 'infrequent' labels  
  
    return word_dict
```



Notebook displays the five most frequently appearing words.



Answer describes how the processing methods are applied to the training and test data sets and what, if any, issues there may be.

Good work on explaining the functionality of the method.
However , the ask is to list down if there will be issues while applying the same set of methods on both train and test dataset

Build and Train the PyTorch Model



The train method is implemented and can be used to train the PyTorch model.

Good work. You have correctly updated the train.py file with the training logic. it covers the important steps such as forward prop , loss calculation , backward propagation and update of parameters.



The RNN is trained using SageMaker's supported PyTorch functionality.


Model trained in the sagemaker environment

Deploy the Model for Testing



The trained PyTorch model is successfully deployed.

Good work , this is an important step in the project. You have correctly deployed the model using a GPU instance.



```
: # TODO: Deploy the trained model
predictor = estimator.deploy(initial_instance_count = 1, instance_type = 'ml.m4.xlarge')

Parameter image will be renamed to image_uri in SageMaker Python SDK v2.
'create_image_uri' will be deprecated in favor of 'ImageURIProvider' class in SageMaker Python SDK v2.
Using already existing model: sagemaker-pytorch-2020-07-29-19-25-34-092

-----!

: predictor.endpoint

: 'sagemaker-pytorch-2020-07-29-19-25-34-092'
```

Use the Model for Testing



Answer describes the differences between the RNN model and the XGBoost model and how they perform on the IMDB data.

Well done , you have identified the difference between both the model. The main advantage of the RNN model is that it can analyze the patterns in the sequence data which can help the model to perform better in the long run with better training and corpus.



The test review has been processed correctly and stored in the `test_data` variable.

Good work , you have tested the model and the model predicted correctly



The `predict_fn()` method in `serve/predict.py` has been implemented.

Deploying the Web App



The model is deployed and the Lambda / API Gateway integration is complete so that the web app works (make sure to include your modified `index.html`).

Well done. model deployed and the URL is updated in the `index.html` file



Answer gives a sample review and the resulting predicted sentiment.

You have test the deployed model. well done

 [DOWNLOAD PROJECT](#)

3

[CODE REVIEW COMMENTS](#)



[RETURN TO PATH](#)