

1 specification requires changes

Dear Student,

Great work 🙌 you can train and deploy the model on the AWS server.

I appreciate your efforts in creating a words dictionary for XGBoost training. Even, I also like that you have trained the LSTM model using the P2 server on AWS and the m4 server has been used to deploy the trained model.

But, during the testing on the local system, you have forgotten to add review length at the beginning of the test data.

So, please make changes as per the review comments and resubmit the assignment. till that time, happy learning 📖

Files Submitted

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

Make sure your submission contains:

- The `SageMaker Project.ipynb` file with fully functional code, all code cells executed and displaying output, and all questions answered.
- An HTML or PDF export of the project notebook with the name `report.html` or `report.pdf`.
- The `train` folder with all provided files and the completed `train.py`.
- The `serve` folder with all provided files and the completed `predict.py`.
- The `website` folder with the edited `index.html` file.

Good 🙌 all required files are available in the submission.

- ✓ notebook file
- ✓ train.py
- ✓ predict.py
- ✓ index.html

Preparing and Processing Data

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

Good 🙌 you have explained the other three preprocessing methods very well.

you can also explain more about the following topics with an example:

- Remove HTML tags
- Convert to lower case
- Split string into words
- Remove stopwords
- stemming

For reference, You can find the brief discussion for preprocessing of sentiment analysis [here](#)

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

Good 🙌 you have explained that preprocess should be applied to both datasets and explained both preprocessing functionalities `preprocess_data` and `convert_and_pad_data`.

but here, you have forgotten one more point to explain that is **data leakage**.

in the answer you can also add, the applied processes are correct because `word_dict` is generated by the training dataset only so there won't be a leaking issue.

for more information about data leakage [click here](#)

Notebook displays the five most frequently appearing words.

Great work 👍

you are able to get the top five frequent words as follows:

```
['movi', 'film', 'one', 'like', 'time']
```

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

Good 🙌 The implementation is a very basic programming approach.

To make the implementation more pythonic, you can implement the `build_dict` function as follows:

```
from collections import Counter

def build_dict(data, vocab_size = 5000):
    word_count = Counter(np.concatenate(data))

    sorted_words = sorted(word_count, key=word_count.get, reverse=True)

    word_dict = {word:idx + 2 for idx, word in enumerate(sorted_words[:vocab_size - 2])}

    return word_dict
```

There are some other techniques that can help you to create a dictionary for specific use cases.

[link 1](#)

[link 2](#)

Build and Train a PyTorch Model

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

Great, the model is trained properly, as the loss is decreasing over the epochs.

```
Epoch: 1, BCELoss: 0.6960339426994324
Epoch: 2, BCELoss: 0.6861911296844483
Epoch: 3, BCELoss: 0.6780856251716614
Epoch: 4, BCELoss: 0.6695327758789062
Epoch: 5, BCELoss: 0.6595928788185119
```

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

perfectly done 👍

I can see the logs which show that the model was trained properly.

```
2021-09-20 14:29:22,344 sagemaker-containers INFO Reporting training SUCCESS
```

Deploy a Model for Testing

The trained PyTorch model is successfully deployed.

it just one-liner code 😊 you remembered it very well

```
predictor = estimator.deploy(instance_type='ml.m4.xlarge', initial_instance_count=1)
```

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.

Make sure your answer includes:

- The comparison between the two models
- Which model is better for sentiment analysis

In this situation, yes XGboost performs similarly compared to LSTM but LSTM has an advanced structure in deep learning, so LSTM performs better overall.

There are some points which I want to highlight:

- input features for XGBoost are Bags of words and for the LSTM model, it is Word embedding. so the difference between the two features should be explained.
- The architectural difference between XGBoost and LSTM.
- Limitation of XGBoost should be mentioned.
- LSTM model complexity should be explained.

for more detail about LSTM and XGboost [click here](#)

The test review has been processed correctly and stored in the `test_data` variable. The `test_data` should contain two variables: `review_len` and `review[500]`.

Good 🙌 you have used `convert_and_pad` and `review_to_words` correctly here.

But if you read TODO which is as follows:

```
TODO: Using the review_to_words and convert_and_pad methods from section one, convert test_review into a NumPy array test_data suitable to send to
```

our model. Remember that our model expects an input of the form review_length, review[500].

here you can read the last sentence which is explaining the expected input should contain form review_length, review[500]. but as per the code, I can see that you have only added review[500]. I have also given the solution as follows:

```
test_review_words = review_to_words(test_review)    # splits reviews to words
review_X, review_len = convert_and_pad(word_dict, test_review_words) # pad review

data_pack = np.hstack((review_len, review_X))
data_pack = data_pack.reshape(1, -1)

test_data = torch.from_numpy(data_pack)
test_data = test_data.to(device)
```

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

- The predict script should include both the data processing and the prediction.
- The processing should produce two variables: data_X and data_len.

Good 🙌 you are using the same method for sagemaker

```
def predict_fn(input_data, model):
    print('Inferring sentiment of input data.')

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    if model.word_dict is None:
        raise Exception('Model has not been loaded properly, no word_dict.')

    # TODO: Process input_data so that it is ready to be sent to our model.
    #
    # You should produce two variables:
    # data_X - A sequence of length 500 which represents the converted review
    # data_len - The length of the review
```

```

processed = review_to_words(input_data)
data_X, data_len = convert_and_pad(model.word_dict, processed)

# Using data_X and data_len we construct an appropriate input tensor.
Remember
# that our model expects input data of the form 'len, review[500]'.
data_pack = np.hstack((data_len, data_X))
data_pack = data_pack.reshape(1, -1)

data = torch.from_numpy(data_pack)
data = data.to(device)

# Make sure to put the model into evaluation mode
model.eval() #will notify all your layers that you are in eval mode,
that way,
#batchnorm or dropout layers will work in eval mode ins
tead of training mode.

# TODO: Compute the result of applying the model to the input data. T
he variable `result` should
# be a numpy array which contains a single integer which is eit
her 1 or 0
with torch.no_grad(): #impacts the autograd engine an
d deactivate it
    output = model.forward(data)

raw = output[0].item() ### assuming only one tensor value
result = 1.0 if raw > 0.5 else 0.0

return result

```

Deploying a 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`).

Good! the link seems valid 👍

<https://50j7myj0tk.execute-api.us-west-2.amazonaws.com/prod>

The answer includes a screenshot showing a sample review and the prediction.

Perfect 🔥 you have attached a test screenshot in the notebook.