Deploying a Sentiment Analysis Model

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Excellent work on the project! Your results look good

I hope the review helped you. If you feel there's something more that you would have preferred from this review please leave a comment. That would immensely help me to improve feedback for any future reviews I conduct including for further projects. Would appreciate your input too. Thanks!

Congratulations on finishing the project! 😄

Files Submitted

✓

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

Preparing and Processing Data

✓

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

Nice explanation!

The Porter Stemming algorithm reduces words to their root form. Words like "onlin" instead of "online" are

indicative of the latter. It also splits the string into individual words, thereby removing all punctuation. If you'd like to, you can learn more about this here - https://pythonprogramming.net/stemming-nltk-tutorial/

A question for you to think about -

How would you consider applying similar methods to another language (like French or Japanese, for example). Do you think the same rules would apply or do you think it would be highly data dependent?

✓

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

A good and concise implementation!

✓

Notebook displays the five most frequently appearing words.

Yes, you are correct.

An extension of this can be used for identifying important sentences in a particular data source. Here is a good resource if you'd like to understand that particular application - https://hackernoon.com/finding-the-most-important-sentences-using-nlp-tf-idf-3065028897a3

Question for you to think about -

If you were to assign your own "weights" to each of these words, how would you do that? The weights here imply the importance of each word. Or do you think that their occurrence/frequency is the best possible metric to define their importance for any kind of anaylsis?

✓

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

Good points!

Preprocessing the training set is important to reduce noise in the data set and padding helps bring uniformity across each data point that can help improve performance. The same unfiromity is required for the test data set at time of inference as well. Also, since `word_dict` is created using only the training data, that further reduces the chance of "data leakage" from training to test data, which is important for evaluating our models.

Question for you to think about -

Do you think it's better to apply preprocessing to the training and test sets separately (as has been done in this project) or do you think it should be applied to the entire dataset first, and then that preprocessed data should be split into the two sets. Do you think it's better to include a validation set here as well? Think about which cases would reduce the data leakage and thereby not affect the test set negatively for evaluation.

Build and Train the PyTorch Model

Build and Train the PyTorch Model

✓

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

You correctly implemented the train method.

Quick question for you to think about - Where and how does your learning rate get defined or passed as a hyperparameter? If it's a default value being used for the optimizer, how would you go about adding your own value and is that even required? Try it out if you'd like more control over your implementation!

✓

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

Nice work!

I do recommend trying to play around/experiment more with your `embedding_dim` and `hidden_dim` values and observe how that can improve your results.

While the Adam Optimizer is definitely one of the recommended optimizers to work with, here is a good resource discussing alternatives - http://ruder.io/deep-learning-nlp-best-practices/index.html#optimization In fact, this entire article/post could be quite useful to you!

Deploy the Model for Testing

✓

The trained PyTorch model is successfully deployed.

Nicely done!

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.

Good job.

This can be a tricky question. Often you will find that it's XGBoost is usually preferred for tabular data and for smaller datasets (limited/lesser variables). But as the datasets get bigger, a DNN based model is likely to start outperforming. In this case, based on some fine-tuning you can have either perform better than the other, but the neural network is likely to outperform XGBoost model especially for larger datasets.

✓

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

Nice job!

However, you are losing out on some accuracy here. Based on your model accuracy, you can expect to get higher than this. I would recommend checking out your implementation by printing out your test_data and

seeing if that matches with how your processed training data looks. And, of course, you can continue trying to experiment more if you'd like.

✓

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

Well done!

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`).

✓

Answer gives a sample review and the resulting predicted sentiment.

Your results look good!

If you'd like to, try out more reviews where you can find a reasonable mix of good and bad feedback - for example, a review which might praise the acting a lot, but might have extremely strong negative views on the story. How well do you think your model will perform then? And for some fun, try to feed it some "sarcastic" reviews and observe your results :D

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review