LeeAnn Markwalter

Text classification with the torchtext library

***Summary:***

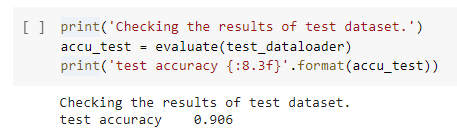
The objective of this project is to build a model for text classification analysis. Essentially the model is trained so that when used on a dataset, text can be classified by defined labels based on the content of the text. Torchtext library is used to build the dataset for text classification analysis. More specifically, the AG\_NEWS dataset is used. This dataset is used because it contains iterators which yield raw data as a pair of labels and text, making it possible to run text classification procedures. Many packages from Pytorch are used to successfully complete the modelling objective. The specific names of the tools, datasets, and pretrained models can be found in the Overview of Modelling Approach section.

***Overview of Modelling Approach:***

1. *Access to the raw data iterators* imports the AG\_NEWS dataset from the torchtext library into the Collab and creates a raw training dataset.
2. The *Prepare data processing pipelines* step uses the build\_vocab\_from\_iterator and get\_tokenizer packages to perform NLP data processing with tokenizer and vocabulary. The packages, along with the defined yield\_tokens function, convert tokens, text string, and labels into integers.
3. *Generate data batch and iterator:* This step uses the torch.utils.data.Dataloader package. The DataLoader works with map-style datasets that implement getitem() and len() protocols. The collate\_fn function works on the batch of samples that is generated from DataLoader and processes them according to the data processing pipeline of step #2. The text entries in the original data batch are packed into a list and combined as a single tensor for the input of nn.EmbeddingBag. Two tensors (offset and Label) are used to mark the beginning of individual sequences and save the labels of individual text entries.
4. *Define the model:* The model is composed of the nn.EmbeddingBag plus a linear layer for the classification purpose. With the default mode of mean, nn.EmbeddingBag computes the mean value of a bag of embeddings. The package also enhances the performance and memory efficiency to process a sequence of tensors.
5. *Initiate an instance:* The model is built with an embedding dimension of 64. The AG\_NEWS dataset has four labels, so the number of classes is set to 4.
6. *Define functions to train the model and evaluate results:* Time is imported into the collab, 2 functions are defined that train and evaluate the data loader.
7. *Split the dataset and run the model:* The original AG\_NEWS training dataset is split into train and valid sets with a split ratio of .95 (train) and .05 (valid). This is done by using the torch.utils.data.dataset.random\_split function in the PyTorch library. CrossEntropyLoss is incorporated to assist in training C classes. SGD implements stochastic gradient descent as an optimizer and StepLR is used to adjust the learning rate through epochs. 10 epochs are used with a learning rate of 5 and batch size of 64. This means that the dataset is passed over 10 times during the training step.
8. *Evaluate the model with test dataset:* The accuracy of the model is tested, and the test accuracy produces an output of .906.
9. *Test on a random news:* The model is tested using a random excerpt of text from the dataset. The model is able to accurately classify the excerpt as sports news.

***Accuracy of the Model:***

In step #8, the model is evaluated with the test dataset, and a test accuracy of .906 is found.

l

***Conclusion:***

In this exercise, a text classification model was built using the AG\_NEWS dataset from torchtext library. After importing the dataset, data processing pipelines were prepared that convert a list of tokens into integers. Then, data batches and iterators were generated using packages and functions imported from pytorch. Next, the model is defined with a composition of an EmbeddingBag layer plus a linear layer for classification. An instance is then initiated and functions are defined to train the model. Finally, the dataset is split into training and valid sets, and 10 epochs are run to produce an accuracy of .906. When the model is tested on a random text string, it accurately categorizes the text as sports news.

When completing the exercise, I was surprised by the amount of predefined packages and functions that were imported into the model. Building models seems to be less of building your own code, and more so knowing which functions and packages to import that will best serve your model. I was also impressed by the overall accuracy of the model being over 90%. I am curious to know what specific methods would be best in improving the overall accuracy of the model. I could see this model being usefully applied in many different ways. For example, a news company may find the categorization interesting to see what ratios of categories are being published at a given time. Overall, I found the exercise very interesting and educational.