Sample Task for Interview Position

Note:

Explain your code with sufficient comments and explanations. Jupyter notebook should contain both code and text cells with sufficient comments.
Use **PyTorch** to implement the **neural network models** in this task.

1. Dataset Generation

We will use the Amazon reviews dataset used in task 1. Load the dataset and build a balanced dataset of 100K reviews along with their labels through random selection similar to task 1. You can store your dataset after generation and reuse it to reduce the computational load. For your experiments consider a 80%/20% training/testing split.

2. Word Embedding

In this part the of the task, you will generate Word2Vec features for the dataset you generated. You can use Gensim library for this purpose. A helpful tutorial is available in the following link:
https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html

(a)

Load the pretrained “word2vec-google-news-300” Word2Vec model and learn how to extract word embeddings for your dataset. Try to check semantic similarities of the generated vectors using three examples of your own, e.g., King − Man + Woman = Queen or excellent ∼ outstanding.

(b)

Train a Word2Vec model using your own dataset. You will use these extracted features in the subsequent questions of this task. Set the embedding size to be 300 and the window size to be 13. You can also consider a minimum word count of 9. Check the semantic similarities for the same two examples in part (a). What do you conclude from comparing vectors generated by yourself and the pretrained model? Which of the Word2Vec models seems to encode semantic similarities between words better?
For the rest of this task, use the pretrained “word2vec-google- news-300” Word2Ve features.

3. Simple Models

Using the Google pre-trained Word2Vec features, train a single perceptron and an SVM model for the classification problem. For this purpose, use the average Word2Vec vectors for each review as the input feature ($x = \frac{1}{N}\sum_{i=1}^{N} W_i$ for a review with N words). Report your accuracy values on the testing split for these models similar to task 1, i.e., for each of perceptron and SVM models, report two accuracy values Word2Vec and TF-IDF features.
What do you conclude from comparing performances for the models trained using the two different feature types (TF-IDF and your trained Word2Vec features)?

4. Feedforward Neural Networks

Using the Word2Vec features, train a feedforward multilayer perceptron net- work for classification. Consider a network with two hidden layers, each with 50 and 5 nodes, respectively. You can use cross entropy loss and your own choice for other hyperparamters, e.g., nonlinearity, number of epochs, etc. Part of getting good results is to select suitable values for these hyperparamters.
You can also refer to the following tutorial to familiarize yourself:
https://www.kaggle.com/mishra1993/pytorch-multi-layer-perceptron-mnist
Although the above tutorial is for image data but the concept of training an MLP is very similar to what we want to do.

(a)

To generate the input features, use the average Word2Vec vectors similar to the “Simple models” section and train the neural network. Report accuracy values on the testing split for your MLP.

(b)

To generate the input features, concatenate the first 10 Word2Vec vectors for each review as the input feature ($x = [W_1^T, ..., W_{10}^T]$) and train the neural network. Report the accuracy value on the testing split for your MLP model.

What do you conclude by comparing accuracy values you obtain with those obtained in the “’Simple Models” section.

5. Recurrent Neural Networks

Using the Word2Vec features, train a recurrent neural network (RNN) for classification. You can refer to the following tutorial to familiarize yourself:
https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html

(a)

Train a simple RNN for sentiment analysis. You can consider an RNN cell with the hidden state size of 10. To feed your data into our RNN, limit the maximum review length to 10 by truncating longer reviews and padding shorter reviews with a null value (0). Report accuracy values on the testing split for your RNN model. What do you conclude by comparing accuracy values you obtain with those obtained with feedforward neural network models.

(b)
Repeat part (a) by considering a gated recurrent unit cell.

(c)
Repeat part (a) by considering an LSTM unit cell.
What do you conclude by comparing accuracy values you obtain by GRU, LSTM, and simple RNN.

Note: In total, you need to report accuracy values for:
2 (Perceptron + SVM) + 2 (FNN) + 3 (RNN) = 7 cases.