MIDAS TASK 3

Name: Harsh Sakhrani
Affiliation: Pune Institute of Computer Technology

Problem Statement:

Use a given dataset to build a model to predict the category using description. Write code in python. Using Jupyter notebook is encouraged.

- Show how you would clean and process the data
- Show how you would visualize this data
- Show how you would measure the accuracy of the model
- What ideas do you have to improve the accuracy of the model? What other algorithms would you try?

Methodology:

As an NLP Problem:

- **Data Pre-Processing and Cleaning:**
    - The primary category was extracted from the category-tree.
    - The primary categories which had a frequency less than 10 were removed.
    - Data Split:
        - Train: 15,724
        - Validation: 1,966
        - Test: 1,966

- **Model Architectures:**
    - Sentence Transformer + RandomForest (Model 1):
        - The idea with this architecture was to combine high quality pre-trained word embedding with lightweight Machine Learning models. We use sentence embeddings extracted from the pre-trained Sentence-BERT as an input to the Random Forest model.
        - Sentence-BERT adds a pooling layer on top of BERT which enables us to create a fixed-size representation for input sentences of varying lengths

        - One could've also used the CLS token embedding, but we felt that a pooling layer would help us create a more wholesome representation of the entire sequence.

    - Pre-Trained BERT + KimCNN (Model 2):
        - Combining pre-trained BERT representations with a CNN architecture seemed like an interesting combination.
        - We decided to go with KimCNN, which is one of the few architectures that makes use of a Convolutional Neural Network in an NLP task in a very intuitive fashion.
        - The idea with KimCNN is to have multiple convolutional filters of different sizes. (For eg. (2 x e), (3 x e), (4 x e), (5 x e) where e is the embedding dimension). This will help the model look at a combination of multiple words as the filter convolves.

    - Pre-Trained BERT + Transform Encoder Block + KimCNN (Model 3):
        - The intuition behind using KimCNN and BERT is the same as mentioned above.
        - The reason we decided to go with a Transformer Encoder Block in between was so that we could get more dataset specific/personalized embeddings being fed to the CNN which in turn should lead to better results.

- **Training Details:**
    - Learning Rate: 0.0001
    - Optimizer: Adam
    - Loss: Cross Entropy Loss

As a MultiModal Problem:

- **Data Pre-Processing and Cleaning:**
    - Text Modality:
        - The primary category was extracted from the category-tree.
        - The primary categories which had a frequency less than 10 were removed.
    - Vision Modality:
        - The images were downloaded using the requests module
        - They were resized to 224 x 224 and normalized.
    - Data Split:
        - Train: 14,415
        - Validation: 1,802
        - Test: 1,802

- **Model Architectures:**
    - MultiModal Model (Model 4):
        - The image column in the dataset has a product-image link for almost every row. Intuitively it made sense to combine both the text and vision modalities to achieve better results.
        - For the text modality we decided to go with *"Pre-Trained BERT + Transform Encoder Block + KimCNN"* and for the image modality we chose a *"Pre-Trained VGG-13 with 3 tunable conv layers".*
        - The PreTrained VGG-13 was trained on ImageNet none of which had e-commerce data. Therefore to customize this for our data, we decided to go with 3 tunable conv layers.
        - The representations from both the image and the text modality are concatenated and fed to an FC layer which finally outputs the class label.

- **Training Details:**
    - Learning Rate: 0.0001
    - Optimizer: Adam
    - Loss: Cross Entropy Loss

Evaluation of Different Architectures:

All the metrics mentioned below are the Test Set Metrics. Each model was trained for 10 epochs.

Model	**Train Time**	**Accuracy**	**F1-Score(Micro)**	**Kappa**	**MCC**
Model 1	10 mins	91.70%	0.9170	0.9006	0.9014
Model 2	45 mins	**97.91%**	**0.9791**	**0.9754**	**0.9754**
Model 3	57 mins	95.32%	0.9532	0.9451	0.9453
Model 4	245 mins	94.45%	0.9445	0.9358	0.9360

Ideas to improve the accuracy of the model:

- Tuning the hyperparameters and trying out various learning-rate scheduling algorithms can lead to a significant increase in the performance.
- Currently we are using pre-trained BERT(Base) embeddings. Rather than using pre-trained embeddings, if we make BERT's parameters tunable, it can lead to an increase in the performance.
- Using BERT(Large) instead of BERT(Base) can also lead to an increase in the performance.

References:

- Convolutional Neural Networks for Sentence Classification
  https://arxiv.org/abs/1408.5882
- BertNet : Combining BERT language representation with Attention and CNN for Reading Comprehension
  15783457.pdf
- PyTorch Docs
  https://pytorch.org/docs/stable/index.html
- Transformers by HuggingFace
  https://huggingface.co/transformers/