Foundations of AI
Instructor: Rasika Bhalerao
Assignment 8
Due December 2

This assignment is to write a text classifier by hand, using Scikit-learn, and using BERT. We will classify SMS messages as spam or ham.

Learning goals:

- Text classification using Naive Bayes
- Text classification using Tfidf and Logistic Regression
- Finetuning BERT
- Text classification using a finetuned large language model

What to do:

1. Download a text classification dataset and read it using Pandas.
    a. Here's an example: https://www.kaggle.com/datasets/team-ai/spam-text-message-classification
2. Naive Bayes "by hand": Train a Naive Bayes classifier on the training set.
    a. Use Laplace (add-1) smoothing.
    b. Report the precision, recall, and F1 score on the test set.
    c. Hints:
        i. Use a regular expression such as `re.split(r"\W+", document)` to tokenize documents. (You may use Scikit-learn's, but regular expressions may be easier.)
        ii. You may use lists, dictionaries, sets, and any other built-in Python data structures.
        iii. The key to knowing you are doing it "by hand" is that you should be writing code to calculate probabilities.
3. Tf-idf "by hand": Create tf-idf vectors for the test set by hand and use them for logistic regression.
    a. Use logistic regression from Scikit-learn
    b. Prepare a vector for each document in the training set using tf-idf. Do this by hand.
    c. Report the precision, recall, and F1 score.
    d. Hints:
        i. Make sure to only take the training set into account when calculating training frequencies.
        ii. When calculating the tf-idf vectors, you may choose to include only words that appear at least `n` times in your training set.

        iii. You will need to use frequencies from the training set when calculating inverse document frequencies for the test set (you will also need to use the same word index order for the vectors that you used in the training set).

4. Naive Bayes and Tf-idf using Scikit-learn: do the classification using the tools available in Scikit-learn!
    a. Naive Bayes: https://scikit-learn.org/stable/modules/naive_bayes.html
    b. Tf-idf: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html with Logistic Regression as before.
    c. Report the precision, recall, and F1 scores for both methods.
    d. Hints:
        i. Make sure to know the difference between `fit`, `transform`, and `fit_transform`, so that the test set frequencies are not included when calculating the training set vectors!
    e. Experiment with one of the parameters in the Tfidf vectorizer (character ngram analyzer, stop words, norm, etc.). Briefly explain what the parameter is, and show how it affected the scores when you changed it.
5. BERT: Finetune BERT to perform text classification!
    a. Some important syntax is available in the LLMs in-class assignment.
    b. Take a BERT model pretrained by Huggingface, add a layer to the end, and train the modified model to do text classification. Perform KFold cross validation to determine the right learning rate (hyperparameter) and use it to finetune the language model to classify each document as spam or ham.
        i. You will need to first split the data into a training set and test set.
        ii. Then, do the KFold cross validation on the training set. Use K = 5 folds. For each fold, four fifths of the training dataset is the "training set" and the remaining fifth is the "dev set" on which to evaluate the model trained during that fold. Rotate so that in each fold, a different fifth has a chance to be the "dev set."
        iii. Pick three different values for the learning rate. Test each value of the learning rate hyperparameter on each fold. Take the learning rate that did the best on any fold (lowest average loss per document) and claim it as your ultimate value for the learning rate.
    c. Each time you train the model, you need to choose a number of epochs. Pick a maximum number of epochs such as 20, and implement early stopping: if the loss does not decrease for 4 epochs in a row, stop training.
    d. Use mini-batches with a batch size of 32 (or smaller). It may help to use the built-in DataLoader (`from torch.utils.data import DataLoader`).
    e. Train your model on the whole training set using your ultimate learning rate hyperparameter value, and evaluate your overall model on the test set. On the test set, report the precision, recall, and F1 score.
    f. Hints:

i. For help with the functions used here, I recommend going directly to Huggingface's documentation, including these two pages: https://huggingface.co/transformers/training.html and https://huggingface.co/transformers/v3.2.0/custom_datasets.html, rather than a separate tutorial. While most tutorials work, some include fatal flaws such as training on the test set.
ii. You may take a random small percent of the data (make sure to include positive and negative labels) and do the assignment using only this smaller dataset. (Make sure your code is running fine, and the slowness was not due to some other error!) The point of this assignment is to learn the NLP material, not about waiting for GPUs!
iii. If you did this correctly, parts b-d should result in loops nested within each other, processing the training set. The test set should only be touched after the loops, for part e.

What to turn in:
Please submit these via Gradescope:

- Your Python code
- A text or pdf file with your answers to these questions:
    - Questions specific to this assignment:
        - Precision, recall, and F1 scores:
            - Naive Bayes by hand
            - Tfidf by hand
            - Naive Bayes using Scikit-learn
            - Tfidf using Scikit-learn
            - Finetuned BERT
        - Experiment with one of the parameters in the Tfidf vectorizer (character ngram analyzer, stop words, norm, etc.). Briefly explain what the parameter is, and show how it affected the scores when you changed it.
    - The usual questions:
        - How long did this assignment take you? (1 sentence)
        - Whom did you work with, and how? (1 sentence each)
            - Discussing the assignment with others is encouraged, as long as you don't share the code.
        - Which resources did you use? (1 sentence each)
            - For each, please list the URL and a brief description of how it was useful.
        - A few sentences about:
            - What was the most difficult part of the assignment?
            - What was the most rewarding part of the assignment?
            - What did you learn doing the assignment?
            - Constructive and actionable suggestions for improving assignments, office hours, and class time are always welcome.