Romanian sub-dialect identification

A.I. project by Alexandru Glodeanu

Chosen feature set

I have chosen words as my feature set. In a similar manner to the bag-of-words approach, I am monitoring the occurrence of each word as a feature for training a classifier.

Code explanation

To get the data from Kaggle, I put them into the same folder as my AI project. I used 3 variables (filepath, filepath2 and filepath3) to store the paths of the .txt files I would need to open.

I created functions which required file paths as parameters.

For the train samples and test samples I created 2 similar functions (*get_samples* and *get_test_samples*). Both of them read the file line by line and called the *word_extraction* function for each line. *Word_extraction* was my function made to delete inadequate characters such as '!@#$%^&*()', using the re.sub function and then splitting the sentence into separate words. Lastly, it made all the words lowercase so they would be the same (although I have learned later that *Tfidf* already performs this operation). Afterwards, I just put all the transformed sentences into a new array (*every_sample* and *test_samples.* For labels, I took the second word from each line of the file.

After gathering all the data properly, it was time to test some classifiers. I have decided to use the Tfidf to vectorize. The classifiers I went through were SVC, Decision Tree, Random Forest, Logistic Regression, XGB, MLP. After some heavy testing, I have decided on the latter. To get to this result, I needed to do hyper parameter tuning. Therefore, I used the train_test_split to test each variance and accuracy_score to print the prediction.

Model

The model used in my python code is MLP classifier (Multi-layer Perceptron). This model optimizes the log-loss function using the stochastic gradient-based optimizer.

Hyperparameter choices

- Activation '*relu*' - rectified linear unit function, returns f(x) = max(0,x)

- Solver '*adam*' - refers to a 'sgd' proposed by Kingma, Diederik and Jimmy Ba
- Learning rate 'adaptive' - keeps the learning rate constant to '*learning_rate_init*' as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if 'early_stopping' is on, the current learning rate is divided by 5.
- Learning_rate_init = 1e-5 - controls the step-size in updating the weights.
- Max_iter = 600 - maximum number of iterations. In my stochastic solver case ('*adam*'), this determines the number of epochs
- Random_state = 1 - seed used by the random number generator
- Hidden_layer_sizes = 250 - the ith element represents the number of neurons in the ith hidden layer
- Verbose = True - prints progress messages

Macro F1 scores for each validation set

F1 score: 0.6681942908855725

Confusion matrix:

[[864 437]

[444 911]]