Assignment 1: Language Identification

Motivation

The motivation of this assignment is to get the students some practice with text categorization using machine learning. We do for the task of language identification, which is a crucial step for various natural language processing tasks such as machine translation, sentiment analysis, and information retrieval. In this assignment, you will use machine learning techniques to build a supervised classification model for language identification.

Problem Statement

You must develop a classifier that inputs a text in Roman script, and outputs the language in which the text is written. We provide the supervised dataset in JSON format for building this classifier.

1. train.json and valid.json contain training and validation data, respectively. The files use UTF-8 encoding. You can use the following code snippet to read the data correctly.

```
import json
with open('train.json', 'r', encoding='utf-8') as fp:
    train_data = json.load(fp)
print(len(train_data ))
```

2. The train and test data consist of about 800k and 100k samples, respectively. Each sample is a dictionary as given below

```
{
    "text": "esto es español",
    "langid": "es"
}
```

3. The data consists of text from 13 different languages. Text within each sample is written in a single dominant script, i.e., the Roman script.

The Task

In this assignment, you will explore non-neural techniques for classification like Naïve Bayes, Logistic Regression, SVMs, Random Forests, etc. You can start by building a baseline using any of these models with unigrams as features. We recommend that you go over the data

thoroughly, build new features that you think are best for this problem, and improve over the baseline. We provide a few suggestions below.

1. We have provided a train and validation split of the training data. You don't have to use this specific split. You can do anything you wish with this data (including resplitting to create a test set, etc)
2. You can explore different balancing strategies to handle class imbalance.
3. Experiment with different regularisation strategies for your model. For instance, you can consider L2-regularization for Logistic Regression.
4. Besides unigrams, you can consider bi-gram and tri-gram features. On the other end, you can explore sub-word-level features, too.
5. Consider working with the features by using techniques such as removal of stop words and infrequent words, and TF-IDF weighting.
6. Design regular expressions to identify characteristic patterns in the samples and use them as features.
7. Using trained word embeddings or neural models or any method that uses neural models is strictly prohibited.
8. Perform a good hype-parameter search and select the ones that work the best.
9. You can also explore strategies like One v/s One, One v/s All for Multiclass classification.

Test Data

Test data will be stored in a JSON file using UTF-8 encoding. You can load it as discussed in the Problem Statement. Each sample in test set is a dictionary given below

```
{
    "text": "esto es español"
}
```

Observe that "langid" is absent in the test set.

Evaluation Metric

We will assess the predictive power of your submission using the MicroF1 and MacroF1 scores. Lots of material like this is available on the internet. For this assignment, we compute MicroF1 and MacroF1 on test set as follows

1. We initialize counts for true positive (TP), false positive (FP) and false negative (FN) to zero.
2. For a test text written in English, we increment the FP and FN by 1 if the submission makes a mistake.

3. For a test text written in a language other than English, we increment TP by 1 if the submission identifies the language correctly. Otherwise, we increment both FP and FN by 1.

We compute Precision, Recall and F1 using TP, FP and FN counts. Observe that our MicroF1 has a special treatment for English differently. Similarly, we compute these values for each language (other than English) to compute Macro F1.

Submission Instructions

The submission deadline for the assignment is 12th Feb 2024. We will use Moodle for submission and provide detailed guidelines over Piazza. We will use Piazza for doubt clarifications as well.

Submission Format

You must submit a zip file on moodle with name <kerberos_id.zip> (E.g. csz208845.zip). Unzipping this

should generate a directory with name <kerberos_id> (E.g. csz208845). This directory must contain

a run_model.sh and writeup.txt.

The command to train the model is

```
bash run_model.sh train <path_to_data_json> <path_to_save>
```

This command should save the trained model using the data given at location path_to_data_json. Further, it must save the trained model and other necessary files in path_to_save directory.

The command to run inference using model is

```
bash run_model.sh test <path_to_save> <path_to_test_json> output.txt
```

This command load the model saved in path_to_save and predict language for each sample from the JSON file located at path_to_test_json. Further, it must store the predicted languages in output.txt. Each line in output.txt should include language ids corresponding to each sample in the test data.

We will release detailed instructions for submitting a trained model soon.

Submission Guidelines

Your submission must adhere to the following instructions

1. The assignment is to be done individually.
2. You should use Python 3.10 for this assignment. Scikit-learn (version 1.4.0) and its dependencies are the only extra packages your code can use. Use of text processing libraries like NLTK and Spacy is strictly prohibited. Any additional packages can be requested over Piazza and are only allowed after verification.
3. Using any external source of data is prohibited.
4. Your code will be tested on a Linux system with 16GB of memory and 4 cores. You are responsible for making sure that it runs smoothly on Linux without any errors.
5. You must not discuss this assignment with anyone outside the class. Make sure you mention the

names in your write-up in case you discuss with anyone from within the class. Please read

academic integrity guidelines on the course home page and follow them carefully.

6. We will run plagiarism detection software. Any person found guilty will be awarded a suitable penalty as per IIT rules.
7. Your code will be automatically evaluated. You will get a 20% flat penalty if it does not conform to output guidelines.
8. Please do not use ChatGPT or other large language models for solutions to the problem. Our TAs will ask language models for solutions to this problem and add its generated code in plagiarism software. If plagiarism detection software can match with TA code, you will be caught.