SNLP assignment 3: Language identification

Deadline: Jun 13, 2018 @ 10:00 CEST

In this set of exercises, we will train a language identifier using the corpora we built collaboratively in the first exercise set. Our aim is to identify the language of a given text. Language identification is generally considered as a 'mostly solved' problem, although interesting issues remain with closely-related languages and multi-lingual texts (e.g., including code-switching).

For the sake of exercise, we will build relatively 'dumb' models in this assignment. Particularly, the features you are asked to use are not ideal (though not too bad either) for the task at hand. You are encouraged to try better models/methods in Exercise 7, but, otherwise, please follow the instructions closely for the submitted assignment.

Why am I doing this?

- Experiment with classification using a real NLP application
- A first, practical exposure to n-grams and *bag-of-words* (or bag of n-grams) representation
- Experiment with regularization and model selection

General instructions

This assignment consists of seven exercises. Your solutions should include a single Python script (`download-tweets.py`) and a CSV-formatted data file (`tweet-corpus.csv`) for Exercise 1. Please create a single Python script, `language-detect.py`, for exercises 2 through 6.

For Exercise 7, you can create as many additional code/data files as you need (or alternatively use the results from Exercise 6 without further effort). However, the requirement is a single data file `predictions.csv`, which contains the predicted language codes from your best model.

All source code and data files must be pushed to your assignment repository before the deadline.

Exercises

Exercise 1. *Gather your data*

A subset of the tweet IDs collected by the each participant of Assignment 1 is provided in your assignment repository.

Write a python script, `download-tweets.py` that

- takes a list of file names as in our data set from the *command line*[1]
- downloads the tweets with the ids
- writes a single comma-separated-value (CSV) file with the name `tweet-corpus.csv` where the first column is the three-letter language code, second column is the tweet id, and the last column is the tweet text[2]

[1] Although we will only use this fixed data set in this exercise, your code should be able to handle any file list given on the command line.

[2] You are recommended to use a library, e.g., Python `csv` library, for writing (and later reading) the data.

Exercise 2. *Feature extraction*

Write a Python function that[3]

- reads the CSV file created in Exercise 1
- counts the number of character bigrams in each document,

[3] **Do not use** `sklearn` 'vectorizer' models in this exercise. Although, this exercise takes only a few lines to solve using `sklearn.feature_extraction.text.CountVectorizer`, you are required to implement it yourself for the sake of exercise.

- and returns a tuple containing:
  - a matrix whose rows correspond to the tweets, and columns correspond to the every bigram that occurs in the data set[4]
  - associated language codes

[4] You may want to use a sparse matrix implementation from `scipy` package in case you encounter memory issues.

Exercise 3. *Logistic regression*

Train a Logistic regression classifier (using `sklearn.linear_model.LogisticRegression`) on the complete data set created in Exercise 2.

Print out the accuracy of your model on the training set using `score()` method of the model.

Exercise 4. *Precision, recall, F-score*

- Write a function that takes two sequences (gold-standard labels and predicted labels) and a label (language code), and returns the precision, recall, and F1-score with respect to the given label.[5]
- Print out *macro averaged* precision, recall, and F1-score of the model trained in Exercise 3 on the training set.

[5] Again, **do not use** the implementation from `sklearn.metrics` package, but implement the function yourself.

Exercise 5. *K-fold CV*

Implement a function that performs k-fold validation.[6]

Using your function, output the mean of the macro-averaged precision, recall and F1-score over 5-fold cross validation.

[6] Yet again, **do not use** the implementation from `sklearn.model_selection` package, but implement the function yourself.

Exercise 6. *Model selection*

The model we studied here is rather simple, it has a single hyperparameter, the regularization constant.

Tune your model, searching the value of the hyperparameter that yields the best macro-averaged F1-score. Output the value of the hyperparameter as well as the precision, recall and F1-score obtained with it.

Exercise 7. *A challenge*

Tune a classifier that would achieve the best macro-averaged F1-score with the given training set. For this exercise, you can use any classification algorithm, or any features that you can extract from the given training set.[7]

You will be required to produce predictions on a test set that will be released the day before the deadline. The test data for this exercise will be in the same format as the CSV file, except the language codes and tweet IDs will be replaced by an underscore. A sample test set file is provided as `testset-sample.csv` in your assignment repository. The output file (`predictions.csv`) should be in the same format, but include the language codes as the first field of the CSV file. Further instructions will be provided together with the real test data.

The teams with the best scores (if statistically not different from the best score) will receive and additional bonus point.

[7] The use of external resources are not allowed. Your model should be tuned and tested only on the data provided (you can also use the sample gold-standard data set from the file `testset-sample+gold.csv` for tuning in the previous and current exercise).

You are also recommended to stick to the classifiers implemented in the `sklearn` package.