PR1: Medical Text Classification

Published Date:
Feb. 21, 2019, 6:00 p.m.

Deadline Date:
Mar. 7, 2019, 11:59 p.m.

Description:

****
This is an individual assignment.
****

Overview and Assignment Goals:

The objectives of this assignment are the following:

- Choose appropriate techniques for modeling text.
- Implement a **min-epsilon k-nearest neighbor classifier** (cannot use libraries for the implementation – see details below).
- Use your version of the min-epsilon k-NN classifier to assign classes to medical texts.
- Think about dealing with imbalanced data.
- Evaluate results using the F1 Scoring Metric (can use libraries for this part).

Detailed Description:
Develop predictive models that can determine, given a medical abstract, which of 5 classes it falls in.

Medical abstracts describe the current conditions of a patient. Doctors routinely scan dozens or hundreds of abstracts each day as they do their rounds in a hospital and must quickly pick up on the salient information pointing to the patient's malady. You are trying to design assistive technology that can identify, with high precision, the class of problems described in the abstract. In the given dataset, abstracts from 5 different conditions have been included: digestive system diseases, cardiovascular diseases, neoplasms, nervous system diseases, and general pathological conditions.

The goal of this competition is to allow you to develop predictive models that can determine, given a particular medical abstract, **which one of 5 classes it belongs to.** As such, the goal would be to develop the best classification model, with the restriction that you can only use your own implementation of the min-epsilon k-NN classifier. Given your implementation, there are many choices in text pre-processing and modeling, proximity measures to use, and classifier meta-parameters that will lead to many different solutions for the problem. Your goal is to find the best one.

The min-epsilon k-NN classifier is defined similarly as the k-NN classifier with the exception that neighbors 2 to k are additionally restricted to have a minimum similarity of epsilon with the query object. In other words, restrict neighbors by both number of neighbors and minimum similarity, but always retrieve at least one neighbor. Given the retrieved neighbors, you must still decide on the way you aggregate their labels to make the final decision (e.g., majority count or weighted sum). If you are working with distances instead of similarities, you can change the problem into a max-epsilon k-NN problem, where epsilon is the maximum distance from the query.

As we have learned in class, there are many ways to represent text as sparse vectors. Feel free to use any of the code in activities or write your own for the text processing step. You may use external libraries for pre-processing and modeling, but not for the k-NN algorithm. You will loose 50% of the points if you use an external library there. Numpy and Scipy data structures (e.g., numpy.array, scipy.csr_matrix, etc.) and functions within those data structures are OK to be used. Functions outside those structures (e.g., numpy.linalg.norm) are not OK.

Since the dataset is imbalanced, the scoring function will be the **F1-score** instead of Accuracy.

Caveats:

+ The dataset has an imbalanced distribution i.e., there are different numbers of samples in each class. No information is provided for the test set regarding the distribution.

+ Use the data mining knowledge you have gained until now, wisely, to optimize your results. Don't cheat. You won't learn anything if you do.

Data Description:

The training dataset consists of 14438 records and the test dataset consists of 14442 records. We provide you with the training class labels and the test labels are held out. The data are provided as text in train.dat and test.dat, which should be processed appropriately.

train.dat: Training set (class label, followed by a tab separating character and the text of the medical abstract).
test.dat: Testing set (text of medical abstracts in lines, no class label provided).
format.dat: A sample submission with 14442 entries randomly chosen to be 1 to 5.

Rules:

- This is an individual assignment. Discussion of broad level strategies are allowed but any copying of prediction files and source codes will result in an honor code violation.
- Some of your classmates may choose not to see the leaderboard status prior to the submission deadline. Please do not share leaderboard status information with others.
- The public leaderboard shows results for 50% of randomly chosen test instances only. This is a standard practice in data mining challenges to avoid gaming of the

system. The private leaderboard will be released after the submission deadline, based on all the entries in the test set.

- In a given day (00:00:00 to 23:59:59), you are allowed to submit a prediction file only 5 times.
- The final ranking will always be based on the last submission or the submission you choose, not your best submission. Carefully decide what your chosen submission should be.
- Each time you submit a prediction file, you will also need to include the code that generated that prediction. Acceptable formats are: py, tar.gz, zip. Please do not submit Jupyter notebooks. Export your notebook to a .py file and ensure it can be independently executed if data files are in the same directory as the script. *Your submission will not be valid unless it produces the output in the prediction file.*

Deliverables:

- Valid submissions to the Leader Board website: https://coe-cmp.sjsu.edu/clp/ (username is your MySJSU email and your password is your MySJSU password).
- **Canvas submission of report:**
  - Write a 2-page, single-spaced report describing details regarding the steps you followed for text processing and classifier model development. The report should be in PDF format and the file should be called **<SJSU_ID>.pdf**. Be sure to include the following in the report:
  - Name and SJSU ID.
  - Rank & F1-score for your submission (at the time of writing the report). If you chose not to see the leaderboard, state so.
  - Your approach.
  - Your methodology of choosing the approach and associated parameters.
  - Any special instructions for running your code.

Grading:

Grading for the Assignment will be split on your implementation (70%), report (20%) and ranking submissions (10%). Extra credit (1% of final grade) will be awarded to the top-3 performing algorithms. Note that extra credit throughout the semester will be tallied outside of Canvas and will be added to the final grade at the end of the semester.

Files: In Canvas, you can find

- *Training Data:* train.dat
- *Test Data:* test.dat
- *Format File:* format.dat