# Social Computing [CS60017] 2020A

Assignment 3: *Detecting Hate Speech on Social Media posts*

**Deadline for submission: 14 November 2020, 23:59 IST.**

---

## General Instructions

1. Read the instructions given in each section carefully.
2. You can use any libraries you want (like scikit-learn). For Task 2 you are free to use any codes from the internet (such as from Github), without restriction.
3. Your codes should print out the output files in the specified format (check the sample given). Do **NOT** output anything extra that isn't asked for. The results will be evaluated by an automated checker. There will be a **penalty** if the specified output format is not followed.
4. Try writing codes into different functions and add several comments. Make sure your code can be easily read. Illegible codes will be **penalised**!
5. **How and what to submit:** Solutions should be uploaded via the CSE Moodle website (see course website for details). Submit one .zip or .tar.gz file containing a compressed folder that should contain all source codes, all files to be submitted (as per the task descriptions given below) and an instructions file (see next point). Name the compressed file the same as your roll number. Example: name the compressed file "19CS60R00.zip" or "19CS60R00.tar.gz" if your roll number is 19CS60R00.
6. Along with the source codes and files asked in the tasks, also submit an additional text file called "**instructions.txt**" where you should state how to run your codes as well as any additional information you want to convey, such as the version of Python or C++ compiler. The instructions.txt file should also contain your *name* and *roll number*. Also mention all the libraries you have used for the assignment, and a brief description of the model you submitted for Task 2. If this file isn't present, your submission will **NOT** be evaluated!
7. We should be able to run your submitted code in a computer with a reasonable configuration (for instance 4GB or more RAM) by following your submitted instructions. If any part of your code takes a long time to run (e.g., more than 15 minutes) report that in the instruction file with an estimate of time required.
8. The assignment should be done individually by each student. You should not copy any code from one another. Students with very similar codes will be awarded zero for the whole assignment.

## Dataset

You will find two datasets as given below. Keep them in a folder called **"data"**.

- **train.tsv:** This file contains the data for training purposes. Each line represents a tweet with 3 columns, separated by tabs ( \t ). The first column is a unique identifier, the second column contains the tweet text, and the third contains the label of whether the tweet is hate speech (1) or not (0). There are 16,112 tweets in this file.
- **test.tsv:** This file contains the data that your model should predict, and you will be scored on the predictions. It has the same format as before, but with only the first two columns. There are 5,000 tweets in this file.

## Task 1: Generic Classifiers [85 points]

We want a classifier to get a baseline classification of whether a given tweets is hateful or not. Firstly we want to preprocess the tweet texts to make them easier to work with. Preprocess the texts in **train.tsv** and **test.tsv** by removing punctuations and lowercasing all characters. Then we are going to try three types of classifiers:

1. **[30 points] TFIDF vectors with Random Forrest Classifier:** TFIDF vectorization is a popular way to convert a piece of text into a vector embedding (Details given here: [link1] [link2]). You can use the Scikit-Learn library for this part.

   Preprocess each of the tweets and convert them into TFIDF vectors, excluding words with document frequency more than 80% and less than 5. After converting the tweet texts to vectors of "features" train a basic Random Forrest classifer on the tweets from **train.tsv** to be able to predict the corresponding labels. Use the trained model to predict if the tweets from **test.tsv** are hateful or not.

2. **[30 points] Word2Vec Embeddings with SVM Classifier:** Word2Vec is another way to convert words to embeddings (vectors). We shall follow a bag of words approach, in which the feature vector of a tweet is the mean of the embeddings of all the words in the tweet.

   After preprocessing the tweets, generate their vectors using pre-trained embeddings from the SpaCy medium English language model *(en_core_web_md)* [link to tutorial]. Alternatively, if you want you can also train your own word2vec embeddings on the training data using a library like Gensim. Then train an SVM classifier with RBF kernel on the vectors from **train.tsv** and predict the labels on **test.tsv** as before.

3. **[25 points] Supervised FastText:** FastText is an extension of the word2vec model that uses character level information [link to tutorial]. Train a supervised fasttext classifier on the tweets from **train.tsv** and use it to predict the labels on **test.tsv**. You can use it either the Python library, or directly as a command line tool.

In a folder called **"task1"** have a file called **main.py** or **main.cpp** that will be run without any arguments from the command line (you can split your code into multiple files though). In all the sub-parts, you can use the default values for hyper-parameters for all models, or relatively smaller values. It is **NOT** required to get high performance out of your models in this Task.

Output the tweet ID and the predictions into three files for each of the sub-parts: **"RF.csv"**, **"SVM.csv"** and **"FT.csv"** respectively, as comma separated values with a header. Keep the generated files inside a folder called **"predictions"**. A sample of the file format is given at the end.

## Task 2: Any preferred model [15 points]

Submit any classification model other than the ones from Task 1 (your own or from the internet) that will predict the labels. Put all your codes insude a folder called **"task2"**. You need to save your predictions in the same format as Task 1, in a file called **"T2.csv"**

The model used for Task 2 should be clearly described in the file **"instructions.txt"**, including links to any code borrowed from the Internet, and how to run the submitted code (including any module dependencies). We should be able to replicate the exact submitted results using the description and the submitted code.

**Note:** You can **NOT** train your model on any external data other than **train.tsv**. To get an estimate of your performance, you can use cross validation [tutorial] on the train set, or take a random subset of the train set as a validation set.

**Evaluation Procedure:** A leaderboard shall be created based on the macro F1-score on the test dataset. You will be marked on a logarithmic scale based on your rank in the leaderboard.

To give you a reference to start with, we have tested a few very simple off the shelf ML models and calculated the macro F1 scores (given below). The text was preprocessed by lowercasing, removing punctuations and the lemmatizing words. Then they were converted to TF-IDF vectors excluding words with document frequency more than 90% and less than 5. The macro F1 scores are:

| Model $\geq$ | Macro F1 Score |
|---|---|
| Decision Tree | 0.6885 |
| Neural Net (2 layer MLP) | 0.7342 |
| AdaBoost Classifier | 0.7654 |
| SVM with RBF Kernel | 0.7734 |

# Submission Format

The prediction files should contain the header and 5000 more lines corresponding to each tweet in **test.tsv**, in the same order as given in **test.tsv**. Keep all the generated files inside a folder called **"predictions"**.

The first few lines will look like this:

```
id,hateful
397250289,0
557603687,0
51361623,1
947566416,1
...
```

The folder structure of the submission for a student with roll number 19CS60R00 should look as given below. Not following this structure will lead to penalty!

```
19CS60R00
  +--instructions.txt
  +--data
  |    +--train.tsv
  |    +--test.tsv
  +--predictions
  |    +--RF.csv
  |    +--SVM.csv
  |    +--FT.csv
  |    +--T2.csv
  +--task1
  |    +--main.py
  |    +--other files
  +--task2
  |    +--required files
```

*For any queries regarding the assignment, contact TA* **Soham Poddar** *(email id on course website).*