COMP-9318 Final Project

Instructions:

1. This note book contains instructions for **COMP9318 Final-Project**.
2. You are required to complete your implementation in a file `submission.py` provided along with this notebook.
3. You are not allowed to print out unnecessary stuff. We will not consider any output printed out on the screen. All results should be returned in appropriate data structures returned by corresponding functions.
4. This notebook encompasses all the requisite details regarding the project. Detailed instructions including **CONSTRAINTS**, **FEEDBACK** and **EVALUATION** are provided in respective sections. In case of additional problem, you can post your query @ Piazza.
5. This project is **time-consuming**, so it is highly advised that you start working on this as early as possible.
6. You are allowed to use only the permitted libraries and modules (as mentioned in the **CONSTRAINTS** section). You should not import unnecessary modules/libraries, failing to import such modules at test time will lead to errors.
7. You are **NOT ALLOWED** to use dictionaries and/or external data resources for this project.
8. We will provide you **LIMITED FEEDBACK** for your submission (only **15** attempts allowed to each group). Instructions for the **FEEDBACK** and final submission are given in the **SUBMISSION** section.
9. For **Final Evaluation** we will be using a different dataset, so your final scores may vary.
10. Submission deadline for this assignment is **23:59:59 on 27-May, 2018**.
11. **Late Penalty: 10-% on day-1 and 20% on each subsequent day.**

Introduction:

In this Project, you are required to devise an algorithm/technique to fool a binary classifier named `target-classifier`. In this regard, you only have access to following information:

1. The `target-classifier` is a binary classifier classifying data to two categories, *i.e.*, **class-1** and **class-0**.
2. You have access to part of classifiers' training data, *i.e.*, a sample of 540 paragraphs. 180 for **class-1**, and 360 for **class-0**, provided in the files: `class-1.txt` and `class-0.txt` respectively.
3. The `target-classifier` belong to the SVM family.
4. The `target-classifier` allows **EXACTLY 20 DISTINCT** modifications in each test sample.
5. You are provided with a test sample of **200** paragraphs from **class-1** (in the file: `test_data.txt`). You can use these test samples to get feedback from the target classifier (**only 15 attempts** allowed to each group.).
6. **NOTE: You are not allowed to use the data `test_data.txt` for your model training (if any). VIOLATIONS in this regard will get ZERO score.**

-to-do:

- You are required to come up with an algorithm named `fool_classifier()` that makes best use of the above-mentioned information (**point 1-4**) to fool the `target-classifier`. By fooling the classifier we mean that your algorithm can help mis-classify a bunch of test instances (**point-5**) with minimal possible modifications (**EXACTLY 20 DISTINCT** modifications allowed to each test sample).
- **NOTE::** We put a **harsh limit** on the number of modifications allowed for each test instance. You are only allowed to modify each test sample by **EXACTLY 20 DISTINCT tokens (NO MORE NO LESS)**.
- **NOTE:: ADDING** or **DELETING** one word at a time is **ONE** modification. Replacement will be considered as **TWO** modifications (*i.e.*, **Deletion** followed by **Insertion**).

Constraints

Your implementation `submission.py` should comply with following constraints.

1. You should implement your methodology using `Python3`.
2. You should implement your code in the function `fool_classifier()` in the file `submission.py`.
3. You are only allowed to use pre-defined class `strategy()` defined in the file: `helper.py` in order to train your models (if any).
4. You **should not** do any pre-processing on the data. We have already pre-processed the data for you.
5. You are supposed to implement your algorithm using **scikit-learn (version=0.19.1)**. We will **NOT** accept implementations using other Libraries.
6. You are **not supposed to augment** the data using external/additional resources. You are only allowed to use the partial training data provided to you (*i.e.*, `class-1.txt` and `class-0.txt`).
7. You are **not** allowed to use the test samples (*i.e.*, `test_data.txt`) for model training and/or inference building. You can only use this data for testing, *i.e.*, calculating success %-age (as described in the **EVALUATION** section.). **VIOLATIONS IN THIS REGARD WILL GET ZERO SCORE**.
8. You are **not** allowed to hard code the ground truth and any other information into your implementation `submission.py`.
9. Considering the **RUNNING TIME**, your implementation is supposed to read the test data file (*i.e.*, `test_data.txt` with 200 test samples), process it and write the modified file (`modified_data.txt`) within **12 Minutes**.
10. Each modified test sample in the modified file (`modified_data.txt`) should not differ from the original test sample corresponding to the file (`test_data.txt`) by more than 20 tokens.
11. **NOTE::** Inserting or Deleting a word is **ONE** modification. Replacement will be considered as **TWO** modifications (*i.e.*, deletion followed by insertion).

Submission Instructions:

- Please read these instructions **VERY CAREFULLY**.

FEEDBACK:

- For this project, we will provide real-time feed-back on a test data (*i.e.*, the file `test_data.txt` containing **200** test cases).
- Each group is allowed to avail only **15 attempts in TOTAL**, so use your attempts **WISELY**.
- We will only provide **ACCUMULATIVE FEEDBACK** (*i.e.*, how many modified test samples out of **200** were classified as Class-0). We **WILL NOT** provide detailed feedback for individual test cases.
- For the feedback, you are required to submit the modified text file (*i.e.*, `modified_data.txt`) via the submission portal: http://kg.cse.unsw.edu.au:8318/project/ (http://kg.cse.unsw.edu.au:8318/project/) (using Group name and Group password).
- **NOTE::** Please make sure that the modified text file is generated by your program `fool_classifier()` and it obeys the modification constraints. We have provided a function named: `check_data()` in the class: `strategy()` to check whether the modified file: `modified_data.txt` obeys the constraints.
- Your algorithm should modify each test sample in `test_data.txt` by **EXACTLY 20 DISTINCT TOKENS**.

Final Submission:

1. For final submission, you need to submit:
   - Your code in the file `submission.py`
   - A report (`report.pdf`) outlining your approach for this project.
2. We will release the detailed instructions for the final submission submission via Piazza.

Implementation Details

1. In the file `submission.py`, you are required to implement a function named: `fool_classifier()` that reads a text file named: `test_data.txt` from Present Working Directory(PWD), and writes out the modified text file: `modified_data.txt` in the same directory.
2. We have provided the implementation of **strategy** class in a seperate file `helper.py`. You are supposed to use this class for your model training (if any) and inference building.
3. **Detailed description of input and/or output parts is given below:**

Input:

- The function `fool_classifier()` reads a text files named `test_data.txt` having almost (500-1500) test samples. Each line in the input file corresponds to a single test sample.
- **Note:** We will also provide the partial training data ((i) `class-0.txt` and (ii) `class-1.txt`) in the test environment. You can access this data using the class: `strategy()`.

Output:

- You are supposed to write down the modified file named `modified_data.txt` in the same directory, and in the same format as that of the `test_data.txt`. In addition, your program is supposed to return the instance of the `strategy` class defined in `helper.py`.

- **Note:** Please make sure that the file: `modified_data.txt` is generated by your code, and it follows the **MODIFICATION RESTRICTIONS (ADD** and/or **DELETE EXACTLY 20 DISTINCT TOKENS)**. In case of **ERRORS**, we will **NOT** allow more feedback attempts.

In [1]:
```
# We have provided these implementations in the file helper.py, provide
## Please do not change these functions.
###################
class countcalls(object):
    __instances = {}
    def __init__(self, f):
        self.__f = f
        self.__numcalls = 0
        countcalls.__instances[f] = self
    def __call__(self, *args, **kwargs):
        self.__numcalls += 1
        return self.__f(*args, **kwargs)
    @staticmethod
    def count(f):
        return countcalls.__instances[f].__numcalls
    @staticmethod
    def counts():
        res = sum(countcalls.count(f) for f in countcalls.__instances)
```

```
        for f in countcalls.__instances:
            countcalls.__instances[f].__numcalls = 0
        return res

## Strategy() class provided in helper.py to facilitate the implementa
class strategy:
    ## Read in the required training data...
    def __init__(self):
        with open('class-0.txt','r') as class0:
            class_0=[line.strip().split(' ') for line in class0]
        with open('class-1.txt','r') as class1:
            class_1=[line.strip().split(' ') for line in class1]
        self.class0=class_0
        self.class1=class_1

    @countcalls
    def train_svm(parameters, x_train, y_train):
        ## Populate the parameters...
        gamma=parameters['gamma']
        C=parameters['C']
        kernel=parameters['kernel']
        degree=parameters['degree']
        coef0=parameters['coef0']
        ## Train the classifier...
        clf = svm.SVC(kernel=kernel, C=C, gamma=gamma, degree=degree, c
        assert x_train.shape[0] <=541 and x_train.shape[1] <= 5720
        clf.fit(x_train, y_train)
        return clf

    ## Function to check the Modification Limits...(You can modify EXAC
    def check_data(self, original_file, modified_file):
        with open(original_file, 'r') as infile:
            data=[line.strip().split(' ') for line in infile]
        Original={}
        for idx in range(len(data)):
            Original[idx] = data[idx]

        with open(modified_file, 'r') as infile:
            data=[line.strip().split(' ') for line in infile]
        Modified={}
        for idx in range(len(data)):
            Modified[idx] = data[idx]

        for k in sorted(Original.keys()):
            record=set(Original[k])
            sample=set(Modified[k])
            assert len((set(record)-set(sample)) | (set(sample)-set(rec
        return True
```

In [2]:
```
import helper
def fool_classifier(test_data): ## Please do not change the function de
    ## Read the test data file, i.e., 'test_data.txt' from Present Wori


    ## You are supposed to use pre-defined class: 'strategy()' in the i
    #  and modifications limit checking
    strategy_instance=helper.strategy()
    parameters={}


    ##.................................#
    #
    #
    #
    ## Your implementation goes here....#
    #
    #
    #
    ##.................................#


    ## Write out the modified file, i.e., 'modified_data.txt' in Prese


    ## You can check that the modified text is within the modification
    modified_data = "./modified_data.txt"
    assert strategy_instance.check_data(test_data, modified_data)
    return strategy_instance ## NOTE: You are required to return the i
```


NOTE:

1. **You are required to return the instance of the class: `strategy()`, *e.g.*, `strategy_instance` in the above cell.**
2. **You are supposed to write out the file `modified_data.txt` in the same directory, and in the same format as that of `test_data.txt`**

How we test your code

In [9]:
```
import helper
import submission as submission
test_data='./test_data.txt'
strategy_instance = submission.fool_classifier(test_data)

########
#
# Testing Script.......
#
#
########

print('Success %-age = {}-%'.format(result))
```

```
Success %-age = 89.5-%
```

EVALUATION:

1. For evaluation, we will consider a bunch of test paragraphs having:
   - Approximately 500-1500 test samples for class-1, with each line corresponding to a distinct test sample.The input test file will follow the same format as that of `test_data.txt`.
   - We will consider the success rate of your algorithm for final evaluation. By success rate we mean %-age of samples miss-classified by the `target-classifier` (*i.e.*, instances of `class-1` classified as `class-0` after 20 distinct modifications).

Example:

1. Consider 200 test-samples (classified as **class-1** by the `target-classifier`).
2. For-Example, after modifying each test sample by (**20 DISTINCT TOKENS**) the `target-classifier` mis-classifies **100** test samples (*i.e.*, 100 test samples are classified as **class-0** then your **success %-age** is:
3. **success %-age** = (100) x 100/200 = **50%**

In [ ]:
```
```