Assignment 3 - Authorship Attribution

In this assignment you will be using various algorithms for text classification, performing an authorship attribution task on Donald Trump's tweets.
You will have to submit a comprehensive report, the accompanying code and classification output obtained on a test set.

This assignment could be done in pairs!
Submission deadline: 23:59, Dec 19 2021

Objectives:

1. Learn to work with the sklearn library.
2. Learn to work with the PyTorch library.
3. Learn to design and interpret experiments in NLP (authorship recognition,classification).
4. Understand the differences between the various algorithmic frameworks and their application to different types of data.

Classification: Who Controls this Account

Politicians, as well as other public figures, usually have assistants and staffers that manage most of their social media presence. However, like many other norm defying actions, Donald Trump, the 45th President of the United States is taking pride in his untamed use of Twitter. At times, during the presidential campaign, it was hypothesized (pdf) that Donald Trump is being kept away from his Twitter account in order to avoid unnecessary PR calamities. Trump's tweets are not explicitly labeled (Hillary Clinton, for example, used to sign tweets composed by her by an addition of '-H' at the end of the tweet while unsigned tweets were posted by her staffers). It is known, however, that Trump was using an android phone[1] while the staffers were most likely to use an iPhone. Luckily, the device information is part of the data available via the Twitter API, hence the device used can be used as an authorship label.

In this task you will be using a number of supervised machine learning classifiers in order to validate the hypothesis about Trump tweeting habits.

[1] Trump switched to a secured iPhone in April 2017, hence, building an accurate authorship model on older data can be used for authorship attribution of newer tweets.

Algorithms:

You should use Python's nltk, sklearn and Pytorch packages/libraries for preprocessing, training and testing your classifiers (these packages are well documented and usage examples are part of the documentation).

You should use:

1. sklearn.linear_model.LogisticRegression
2. sklearn.svm.SVC (use both linear and nonlinear kernels!)
3. You should use the PyTorch library to build a neural classifier with an architecture of your choice (with at least one hidden layer) to achieve the classification.
4. You should use the PyTorch library to build a Recurrent Neural Network classifier (dimensions, number of layers and other design details are up to you).
5. A fifth classifier of choice (neural or not). You are encouraged to experiment with classifiers that allow combining different types of features (e.g. number of capitalized words, time of tweeting, etc.)

You are encouraged to use sklearn's cross validation module. Think about the evaluation measures you use.

Data:

A small dataset of a couple of thousands tweets from Trump's account posted between early 2015 and mid 2017 can be found in tweets.tsv, available to download here.
The file is in a tab separated format, each tweet in a new line. The fields in the file correspond to:
<tweet id> <user handle> <tweet text> <time stamp> <device>
While the data is already cleaned and filtered, there is still some degree of freedom you will have to take care of. Specifically:

1. The **handle** field: the handle field can take one of the following three user names: realDonaldTrump (this is Trump's account), POTUS (stands for President of the United States, this is the official presidential account, thus not Trump before the election) and PressSec - the official twitter account of the president's Press Secretary.
2. The **device** field: the device field can take various values ranging from 'android', iphone', instagram (will appear as 'Instagram'), a web client (will appear as 'Twitter Web Client') among other possibilities.
3. The format of the timestamp field is '%Y-%m-%d %H:%M:%S' you can use the *datetime* module and the *strftime()* and *strptime()* functions to parse and process timestamps.

An unlabeled test set with 390 tweets is available here. This file lacks the <tweet id> and <device> fields.

Results

The submitted results file should have a single, space separated line containing only zeros and ones (integers) denoting the predicted class (0 for Trump, 1 for a staffer). The order of the labels MUST correspond to the tweet order in the testset.

Report

You should limit your report to 4 pages (font size 11p, 1.5 space btw. lines), not including references, and you are encouraged to use tables and/or figures along with all necessary details and some analysis. Using the ACL format (overleaf template) is encouraged but not required (latex and MS Word templates are also available). You can submit your report in Hebrew if you like.
(While the purpose of this assignment is not to test your scientific writing - you are expected to produce a coherent report that is readable and grammatically correct; we expect this whether you submit an English or a Hebrew report and we will take up to 10 points for bad writing).

Your report should include a detailed list of models and data assumptions and should indicate the differences between the different algorithms and the various settings as well as a detailed comparison of the results. It should also include **your insights and conclusions** as learnt from the data. Specifically you should address the following:

1. What data/features were finally used - how the corpus was preprocessed and filtered.
2. What is the data representation (input) for each algorithm.
3. What are the settings used for each algorithm?
4. Comparison between algorithms and settings.
5. If there are significant performance differences between algorithms/settings - why do you think that is.
6. You should specify the model and the exact parameters that yielded the best results on the test set (as submitted in the results file).

Extra: can you verify the claim that Trump was kept away from his Twitter account during the campaign?

Submission guidelines:

1. You should submit one tar-ball **tar.gz** file with all relevant code and reports. The file should be named <id1>_<id2>.tar.gz (for joint work, or <id>.tar.gz if you submit without a partner). The tar-ball should at least include the following files:
    a. Report: the main requirement of this assignment is a report file explaining your use of the algorithms and describing the results - comparing results of different algorithms and different configurations of parameters. This file should be named <id1>_<id2>.pdf (or <id>.pdf).

b. You should submit a results file called <id1>_<id2>.txt (or <id>.txt) corresponding to the test set. This file should hold the results of your best performing model. Format specification above.
c. Code files should be well documented with clear usage examples. Everything should be executed through a driver you provide (can contain all the code) called ex**3**_<id1>_<id2>.py. The driver shouldn't expect any parameters. You can assume that all relevant files are in the current directory.
d. Your best performing trained model (as a pickle file or any format you like, as long as your `load_best_model()` (see below) can load it.
e. **The driver should support (at least) the following functions** (they shouldn't expect parameters, you could use defaults or hardcoded filenames):
    i. `def load_best_model()`, returning your best performing model that was saved as part of the submission bundle..
    ii. `def train_best_model()`, training a classifier from scratch (should be the same classifier and parameters returned by `load_best_model()`. Of course, the final model could be slightly different than the one returned by `load_best_model()`, due to randomization issues. This function call training on the data file you received. You could assume it is in the current directory. It should trigger the preprocessing and the whole pipeline.
    iii. `def predict(m, fn)`, **this function does expect parameters**. m is the trained model and fn is the full path to a file in the same format as the test set (see above). `predict(m, fn)` returns a list of 0s and 1s, corresponding to the lines in the specified file.
f. If you opt to use word embeddings that are not part of the standard PyTorch or NLTK libraries you should add the embeddings file to the zip. The embeddings file should contain only the vectors for the vocabulary in the data (that is - it shouldn't be larger than **6**MB). It should be automatically loaded by the relevant model.

2. The input for the code files should be the appropriate raw input files provided above. Preprocessing should be done as part of the execution.
3. You should use Python 3.X for the coding part.
4. You could assume we have the pandas, sklearn and PyTorch libraries installed so they can be imported. Specifically, you could use (and assume we have installed): torchtext, torchnlp
5. While we are not focused on code optimization, we do expect your code to run in a reasonable time. That is - if your code runs for a couple of hours it may suggest something is wrong.