Data Challenge

- Martin Cepeda
- Romain Egelé
- Cédric Javault

Files

First you need to unzip the submitted files.

We submitted the following files :

- `pipeline-feature-engineering-and-model-selection.ipynb`: was use to test the classifiers and build the features
- `pipeline-model-tuning.ipynb` : uses all the features with the sophisticated neural network.
- `dh_load_data.py, dh_problem.py, dh_run.py, dh-analytics-hps.ipynb`: for the hyperparameter search.

Install

Create a python virtual environmemt and run:

```
pip install -r requirements.txt
```

Run

To completely run our work from scratch, you need to first run `pipeline-feature-engineering-and-model-selection.ipynb`. Place the training and testing data in a `data` folder located at the same level than the notebooks and scritps. To execute, every feature embedding computation switch all variables containing `override_` to `True`.

1. Execute all the lines in 'A. Start': WARNING IN SECTION A.2., the paths must be checked with your local implementation
    - training.txt must be here: train_file = './data/training.txt'
    - testing.txt must be here: test_file = './data/testing.txt'
    - The text of the web sites must be here: texts_path_abs = './data/node_information/text' et
2. Execute the lines in 'B.1. Create vector X (Node 1 - Node 2) and y (Linked ?) on training.txt'
3. DO NOT EXECUTE B.2. (Didn't work)
4. Execute the lines in 'B.3. Build UMAP on the Adjency matrix and save it (this might take a few minutes -> we built a code to save it and reload it)' or D.1 to reload
5. Execute the lines in 'B.4. Jacquart coeff'

6. Execute the lines in 'B.5. Build Node2Vec embedding' (this takes quite 30 minutes -> we built a code to save it and reload it)' or D.1 to reload
7. Execute the lines in 'C.1. Very simple feature : size of the document'
8. Execute the lines in 'C.2. Preprocessing : deal with the language issue' (this take 10 minutes -> we built a code to save it and reload it)
9. DO NOT EXECUTE C.3. (Didn't work)
10. DO NOT EXECUTE C.4. (Not needed)
11. Section 'C.5. Latent Semantic Indexing : compute the distance matrix between all doc'
12. Execute the lines in 'C.5.1. Import and loads'
13. Execute the lines in 'C.5.2. Compute LSI matrix of all the documents, not taking their language into account' (this take more then 1 hour - We saved the matrix but it's a 4Go file)
14. DO NOT EXECUTE C.5.3. (Didn't work)
15. DO NOT EXECUTE C.5.4. (Not needed)
16. Execute 'C.6. Build Word2Vec embedding' (Warning -> over 20 hours of computation -> we built a code to save it and reload it) / or D.1 to reload
17. Execute D.1. To reload Umap Node2Vec and Word2Vec if you choose not to recompute them (see below)
18. DO NOT EXECUTE D.2.1. (Not needed)
19. Execute 'D.2.2. Better model with information about the text in the files'
20. Execute 'D.2.3. Save the 7 last colums for Romain's Notebook'
21. Execute 'D.3. 'Prepare the testing set the same way' You are done with "Pipeline Cédric" (the rest of the code is about the different classifiers but not about our best model)

Recap of the features that you can choose to reload instead of rebuilding : In the A.3. Parameters section, there is always two parameters for each feature B.3 UMAP (file 'umap_with_nc_equal3.model') - Set overwrite_umap to True to compute it - Set it to False and load_umap to True to reload precomputed work B.5 Node2Vec (file 'node2vec.model') - Set overwrite_nodes to True to compute it - Set it to False and load_nodes to True to reload precomputed work C.2 Language (file 'languages.txt') - Set language_overwrite to True to compute it - Set it to False and language_load to True to reload precomputed work C.6 Word2Vec (file 'word2vec_300.kv') - Set overwrite_text to True to compute it - Set it to False and load_text to True to reload precomputed work

Then you can run the `pipeline-model-tuning.ipynb` to have the UMAP tuning, and neural network regular training as well as adversarial training.

In order to execute the hyperparameter search do:

```
python -m deephyper.search.hps.ambs --n-jobs -1 --max-evals 10000 --
problem dh_problem.py --run dh_run.py
```

A `result.csv` file will be outputed. Having this file you can execute the final notebook `dh-analytics-hps.ipynb`.