INF442 : projet informatique 9
GDPR in practice: data anonymization

Adrien Ehrhardt

February 29, 2020

1 Software

Contrary to the TDs, you are allowed to use any software you'd like. It could even be a good idea to mix them, *e.g.* by performing the ETL (Extract, Transform and Load) tasks with a high level language (probably Python for this specific *Projet Informatique*), and the computationally heavier stuff in C++. Again, it's all up to you, as long as it is an equivalent of 500 lines of C++ code (at your appreciation).

Bonus points if you're able to mix them in the same file / package / library (see *e.g.* `Cython`).

In this particular *Projet Informatique*, training and inference times are of particular importance: include them in your benchmarks (see Section 4.1) and discuss them in your presentation.

You will need this zipped directory (also available here).

2 Scenario

You are a Data Scientist in an industry that deals with private data. Suppose you have transcripts from interactions people had with their Google Home product. Your team wants to do some fancy machine learning to predict your users' behaviour, *e.g.* what they are going to buy next. The problem is your data comes from many countries, who don't have the same legislation regarding private data, and this data comes from subsidiaries so your users did not agree to pass their data on to the holding company. Your mission, should you choose to accept it, is to anonymize all these texts to remove any personal data before handing them out to the rest of the team.

3 Problem description

First, you will have to transform the raw data to a format suitable to your subsequent analysis. **Example:** the raw text has to be tokenized into "tuples" consisting of a word and a label. Here, a simple, preprocessed dataset is handed to you, but your learning algorithm might benefit from other data sources. Besides, all research papers in the NLP field illustrate their method on several datasets, *e.g.* all these. If you're brave enough, you can also try other languages! See Section 4.4.

The dataset I provide is the classical CONLL 2003 dataset (it can also be found here). The files `*.testa` and `*.testb` should normally be used for testing, whereas the `*.train` file as a train dataset. These files were pre-processed so that :

- The words are tokenized: for now, think as it as some fancy stemming or lemmatization, *i.e.* we keep only the root of each word ("playing" becomes "play");
- The previous step enables us to construct a dictionary: the list of all unique tokens in the document;
- These tokens are converted to a dummy vector which size is the same as the dictionary, where a '1' indicates the presence of that token and '0' its absence;
- All tokens, now represented as dummy vectors, go through some very big neural network. In this *Projet Informatique*, we first focus on BERT, which has 24 layers, 16 attention heads (you do not need to understand this concept to do the *Projet*), 340M parameters. This network "captures" the meaning of the sentence and embeds each of its tokens in a 1024-dimensional vector space. The files `representation.*` (numpy format) are the concatenation of the representation of all tokens. These files are associated with their `true_labels.*` (also numpy format) which is the concatenation of their NER tags.

Note that I kept the original Named-Entity-Recognition (NER) tags (see here for a quick explanation), such that the labels are of 4 kinds: O (nothing in particular), MISC, PERS (our tag of interest) and LOC (a location). Since you have to anonymize, you first have to convert the labels to PERS (say, class '1') vs rest (say, class '0').

Second, you will reuse and / or implement any algorithm seen during the course that is suited to the analysis you want to perform on the sanitized data.

Third, you will present your analysis: do not focus too much about the technical aspects of your method(s), bring the overall reasoning, the results, and possible actions forward.

You have to do at least 2 sub-problems as identified in Section **??**.

4 Resources

Aside from the provided Jupyter notebook, you might find the following resources interesting:

1. For C++:
   - spacy-cpp
   - BERT-NER in C++
2. For Python:
   - transformers (especially the NER examples which form the basis of my Jupyter Notebook)
   - spacy
   - simpletransformers
3. To get up-to-speed in the latest advancements in NLP/NER:
   - Neural networks: INF442!
   - RNNs: this blog post;
   - LSTMs: this blogpost;
   - Attention: this already-linked blogpost;
   - BERT: this blog post.

5 Sub-problems

5.1 Sub-problem 1 - easy (mandatory)

You have to anonymize the ConLL2003 dataset, already pre-processed so that each token (=sub-word) is represented as a numeric vector, one after another, without the notion of sentences. You can thus straightforwardly apply any supervised binary classification algorithm that you've seen in this course, preferably in C++.

5.2 Sub-problem 2 - hard (optional)

You have to build your own text-to-vector-to-label algorithm. It does not have to be as complicated as BERT; for instance, you might use REGEXP rules (such as `if first letter is capital than person`) which will be infinitely faster to infer and compare the complexity / performance trade-off that can be achieved. You can also test other pre-trained architectures (XLnet, distilbert, ...). If you have access to a GPU (you could also try Google Colab), you can even fine-tune these architectures on the anonymization task.

5.3 Sub-problem 3 - medium (optional)

You have to build and compare classifiers for the original Named Entity Recognition (NER) problem in ConLL2003 (you can reuse the algorithms used in Section 4.1). You will compare NER and anonymization accuracies. Is it easier to classify person vs rest or do you get approximately the same accuracies?

5.4 Sub-problem 4 - medium (optional)

You have to use other datasets, see for example this page for a list of readily-available datasets. Do you get the same accuracies? Why? In which way do these datasets differ?

The subject is purposely open-ended: do not get lost into details, choose sub-problems wisely with the time at your disposal.

6 Bonus

Although open-sourcing your work might not be part of your future daily job (highly job-dependent), it might be a good idea to "give back" to the community by making the result of your *Projet Informatique* publicly available, *e.g.* by sharing your analysis *via* a blog post, a Github repository, etc.