INF442 : projet informatique 9
GDPR in practice: data anonymization

Adrien Ehrhardt

March 6, 2021

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 5.1) and discuss them in your presentation.

You will need the Data.zip 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 your product. Your team wants wants to sell this data to Amazonia, which will do some fancy machine learning to predict your users' behaviour, *e.g.* what they are going to buy next. The (legal, not moral) problem is that your data comes from many countries, which don't have the same legislation regarding private data, and your users probably did not agree to give away their private data to Amazonia. Your mission, should you choose to accept it, is to anonymize all these texts to remove any personal data before handing them out to Amazonia.

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 5.4.

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 :

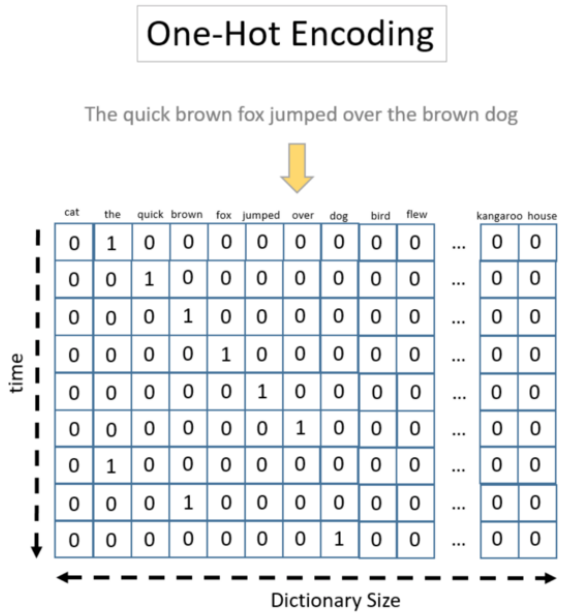
- The words are tokenized: for now, think of it as some fancy stemming or lemmatization, *i.e.* we separate the root of each word ("playing" becomes "play" and "#ing" where # usually represents a word separation into several tokens). Think of each token as an observation, or a row of a matrix, *e.g.* a flower in the famous *iris* dataset;

(a)	**Original:**	furiously	(b)	**Original:**	tricycles
	BPE:	_fur | iously		**BPE:**	_t | ric | y | cles
	Unigram LM:	_fur | ious | ly		**Unigram LM:**	_tri | cycle | s

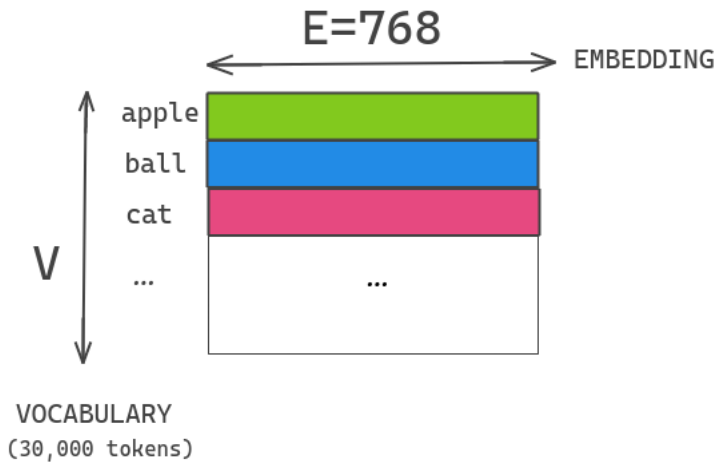
(c)	**Original:**	Completely preposterous suggestions
	BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
	Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

- The previous step enables us to construct a dictionary: the list (of, say, length s) of all unique tokens in the document. Think of this dictionary as the Larousse where each word would be uniquely indexed. Some info about the dictionary used here can be found here. Note that the previous step is very effective in reducing the size of the dictionary, which, in a way, encodes a degree of complexity of the problem;

- Each token x is converted to a dummy vector which size s is the same as the dictionary, where a '1' indicates the presence of that token and '0' its absence, *i.e.* each token is now represented by a flipped "bit" at the location of its index in the dictionary and $s - 1$ 0s;



- All tokens, now represented as dummy vectors $(x_i)_1^n$ where n denotes the number of tokens in the document, 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. Think of this as a transformation $f : \{0,1\}^s \to \mathbb{R}^{1024}$ where $\tilde{x} = f(x)$ is the *BERT representation* of the token x.



The files `representation.*` (numpy format) are the concatenation of the representation of all tokens. Since this is too big to be transferred as CSV files, a sample of 10,000 and 2,000 tokens for train and testa / testb respectively is provided.

We want to predict if a token is associated to a personal entity, and should thus be removed from the text. To this end, all tokens are paired with their 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'). Note that I kept the original ones for Sub-problem 5.3.

These labels are in the `true_labels.*` files (also numpy format). I also provide the CSV sample.

These files were obtained by running the code in `Anonymization.ipynb` which is a Jupyter Notebook. If you'd like to run it as well, you'll find a YAML file named `inf442_pi9_environment.yml` which contains the versions of the packages used in the Python virtual environment that ran the code in the notebook. To install such an environment, assuming you have Anaconda, do `conda env create -f inf442_pi9_environment.yml`. You should now have a conda environment named `inf442_pi9`. To add this environment to the Python kernels available to Jupyter Notebook, do `python -m ipykernel install --user --name inf442_pi9 --display-name "Python (INF442)"`, then open `Anonymization.ipynb`, go to "Change Kernel" > "Python (INF442)" and you should be ready to go.

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 among the ones identified in Section 5.

4 Resources

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

1. For C++:
   - spacy-cpp
   - BERT-NER in C++
   - PyTorch C++ API
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 blog post;
   - Attention: this already-linked blog post;
   - 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 5.1), *i.e.* not just for PERS vs rest. 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.