INF442 : projet informatique 3
GDPR in practice: data anonymization

Contact: Adrien (adrien.ehrhardt@polytechnique.edu)

Conceptual design by Adrien Ehrhardt

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++ (in particular Sub-problem 1 (see Section 6.1). 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 6.1) and discuss them in your presentation.

You will need the data.zip directory.

2 Scenario

1. You are a Data Scientist in a private company. Suppose you have transcripts of phone calls with your after-sales department. Your team wants wants to sell this data to Amazonia, which will probably do some fancy Machine Learning to predict your users' behaviour, *e.g.* what they are going to buy next, to repay for the data they bought from you. The (legal, not moral) problem is that your data comes from many countries, which don't have the same legislation regarding private data, and all 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.

2. Profit!

3 Natural Language Processing: a very brief introduction

Consider for simplicity that your "samples" are raw text, *e.g.* the concatenation of all transcripts of phone calls with your after-sales department. Again, for simplicity, assume that this text is labelled at a suitable granularity, *e.g.* each word corresponds to a label: private / public data. Since you've been paying attention in INF442, you know this looks like supervised binary classification (we have training data; we have to classify between categories and not predict a number; and we have only two categories). **This is the input.**

You wish to sell this data and all subsequent transcripts after anonymization. Obviously, you cannot transfer private data. It is tiresome to manually annotate private / public data "forever" on new transcripts, so you wish to automize this task through Machine Learning. You will thus produce a model which will predict if each word is private or public. **This is the output.**

You know that Machine Learning algorithms deal with numerical values, *i.e.* linear regression will determine optimal coefficients for each (numerical) feature through some matrix calculus. Thus, you have to extract numerical features from each sample (*i.e.* each word).

In TDs 6 and 8, we aim at detecting spam emails and transform each email (think paragraph or collection of sentences) into a numerical vector: we use a dictionary of most common words and simply build a vector with 1s at indices where the corresponding words are present in the email, 0s otherwise. Since this step has been done "manually" by us (we had to choose a dictionary and how to use it), we call that **"feature extraction"**. Mathematically speaking, what we have done is project the data into some particular vector space. In this vector space, the irreducible error (called Bayes error) is fixed: we can only test all Machine Learning models that we know and hope that one of them will reach (in expectation) this error. If we choose the dictionary containing only "Viagra" for example, this irreducible error rate is the rate of spam emails not containing this word. We can probably do much better, not by changing the model, but by changing the vector space!

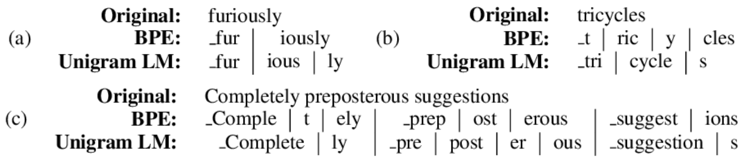
Subsequently, most recent efforts in Natural Language Processing can be classified as **"Representation Learning"** algorithms: they aim at learning the appropriate vector space to represent the data such that the data becomes more separable and the error rate is reduced. In this PI, several Subproblems are proposed. For the first and easiest subproblem, I provide the output of a popular representation learning algorithm such that each word is described by a numerical vector, and you only have to fit your favorite Machine Learning algorithm(s). For all other subproblems, you will have to think about the representation learning part of the overall process. The next section aims at describing this representation learning relatively briefly.

4 Step-by-step problem description

Dataset 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:

Representation

1. 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
BPE:	_fur | iously
Unigram LM:	_fur | ious | ly

(b)

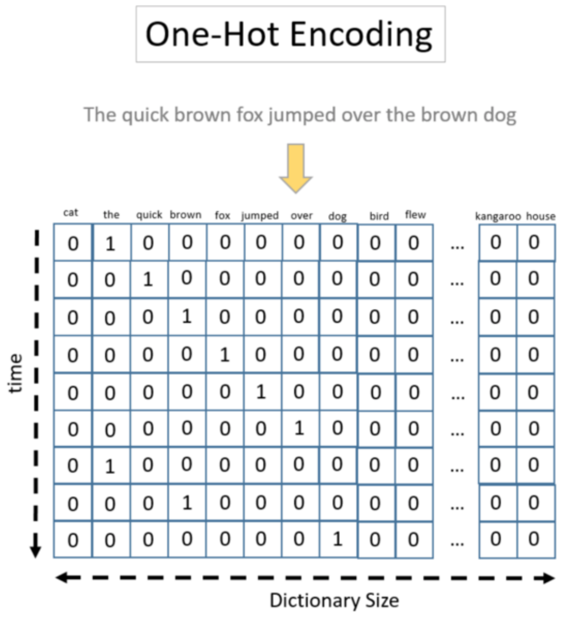
Original:	tricycles
BPE:	_t | ric | y | cles
Unigram LM:	_tri | cycle | s

(c)

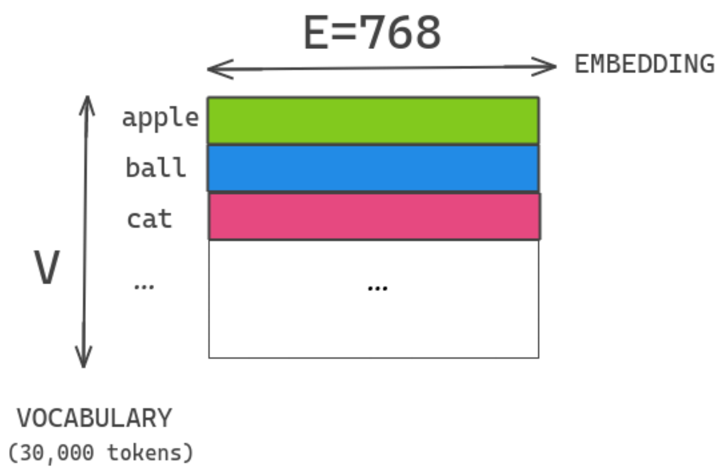
Original:	Completely preposterous suggestions
BPE:	_Comple | t | ely | _prep | ost | erous | _suggest | ions
Unigram LM:	_Complete | ly | _pre | post | er | ous | _suggestion | s

2. 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 step 1. is very effective in reducing the size of the dictionary, which, in a way, encodes a degree of complexity of the problem (by controlling the dimension of the resulting vector space);

3. 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;



4. 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. Recently, GPT (in particular Chat-GPT) has had a lot of (media) attention. This network "captures" the meaning of the sentence and embeds each of its tokens in a 768-dimensional vector space. Think of this as a transformation $f : \{0,1\}^s \to \mathbb{R}^{768}$ where $\tilde{x} = f(x)$ is the *BERT representation* of the token x.



At this stage, we have obtained an *a priori* suitable representation of the data. 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.

Supervised binary classification Now begins the "standard" Machine Learning phase: 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 we have to anonymize, we first have to convert the labels to PERS (say, class '1') vs rest (O + MISC + LOC, say, class '0'). Note that I kept the original ones for Sub-problem 6.3. These labels are in the `true_labels.*` files (also numpy format). I also provide the CSV sample.

Reproduce this process (if necessary) 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_pi8_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_pi8_environment.yml`. You should now have a conda environment named `inf442_pi8`.

Alternatively, you'll find a `requirements.txt` file with which you can install dependencies through `pip install -r requirements.txt\verb`. You'll also find a `Pipfile\verb` with which you can install a virtual environment through `pipenv install\verb`.

To add this environment to the Python kernels available to Jupyter Notebook, do `python -m ipykernel install --user --name inf442_pi8 --display-name "Python (INF442)"` then open `Anonymization.ipynb`, go to "Change Kernel" > "Python (INF442)" and you should be ready to go.

5 Resources

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

1. For C++:
   - BERT-NER in C++
   - PyTorch C++ API
2. For Python:
   - transformers
   - 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;
   - BERT: this blog post.
   - (Chat)GPT: the original paper.

6 Subproblems

6.1 Subproblem 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, paragraphs, documents, ... You have to reuse (this will be relatively straightforward) and / or (re)implement any Machine Learning algorithm seen during the course that is suited to the analysis you want to perform on the sanitized data.

6.2 Subproblem 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.

6.3 Subproblem 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 6.1), *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?

6.4 Subproblem 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.

7 Instructions

Complete the first subproblem and at least one other.

Present your analysis: do not focus too much about the technical aspects of your method(s) (in particular the aspects already explained in this document), bring the overall reasoning, the results, and possible actions (*e.g.* to get better performance, go faster, change perspective) forward.

8 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.