NLP applied to judicial decisions parsing (Predilex Challenge)

Project Report

Ariane Alix
Department of Mathematics
École Normale Supérieure Paris-Saclay
94230 Cachan, France
ariane.alix@ens-paris-saclay.fr

1 Introduction

Our project focuses on the automatic extraction of data from "jurisprudences" of trials between a victim and its insurer. Those are public summaries of french trials containing the context of the accident or crime, the medical status of the victim at the moment of the accident and later on, and the decisions about financial compensation.

1.1 Objectives

The goal is to extract automatically the most relevant data of the texts: the gender of the victim, the date of the accident and the date of consolidation of their injuries.

1.2 Dataset

The dataset is composed of 1027 raw texts of judicial decisions. The first 770 texts are used for training since we have the gender, accident date and injury consolidation date for each of them. No ground truth is given for the 257 other texts; the prediction of those ones is used to rank us in the challenge.

2 Data cleaning

We apply a first simple pre-processing to make the raw texts usable:

- Reading in 'utf-8' to keep the accents that may be important in French,
- deletion of the empty lines,
- deletion of the dashes and series of dashes that appear in many sentences,
- deletion of end-of-line, tabs and other special characters,
- switching to lower cases,
- storing the full texts, and a decomposition in a list of sentences for reach text in easy to use Dataframes.

A extract of a cleaned jurisprudence text looks like:

```
'le : 12/11/2019  cour d'appel d'agen  chambre sociale  audience publique du 30 avril 2002  n° de rg: 01/00515  republique fran
caise  au nom du peuple francais  arret du 30 avril 2002 01/00515 yvon x... annie x... c/ groupe azur assurances iard caisse pr
imaire d'assurance maladie de lot et garonne yannick guguen ès qualités de mandataire liquidateur à la liquidation judiciaire d
e la société des transports morel arret n° cour d'appel d'agen chambre sociale prononcé à l'audience publique du trente avril d
eux mille deux par madame latrabe, conseiller, la cour d'appel d'agen, chambre sociale, dans l'affaire entre : monsieur yvon
x... né le 30 septembre 1955 à tonneins (47) y... du laurier 47300 bias rep/assistant : me pascale luguet (avocat au barreau
d'agen) madame annie x... y... du laurier 47300 bias rep/assistant : me pascale luguet (avocat au barreau d'agen) appelants d'u
n jugement du tribunal des affaires de sécurité sociale d'agen en date du 12 février 2001 d'une part, et : groupe azur assuranc
es iard 7 avenue marcel proust 28032 chartres cedex rep/assistant : me jean-loup bourdin (avocat au barreau d'agen) caisse prim
aire d'assurance maladie de lot et garonne 2, rue diderot 47914 agen cedex 9 représentée par melle z... (mandataire) munie d'un
pouvoir spécial maitre yannick guguen ès qualités de mandataire liquidateur à la liquidation judiciaire de la société des trans
ports morel 22 boulevard saint cyr 47300 villeneuve sur lot rep/assistant : me jean-loup bourdin (avocat au barreau d'agen) int
imee :  d'autre part,  direction regionale des affaires sanitaires et sociales aquitaine cité administrative bp 952 rue jules f
erry 33063 bordeaux cedex ni presente, ni representee partie intervenante : a rendu l'arrêt réputé contradictoire suivant.
```

3 Some definitions

3.1 N-grams

In the context of Natural Language Processing, an N-gram is a sequence of N items from a given sample of text. The items can be phonemes, syllables, letters or words according to the application. Using Latin numerical prefixes, an N-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram". [1]. In our case, we will consider N-grams of words extracted from the corpus, with N varying from 1 to 4.

3.2 Bag of Words representation

NLP is a major application field for machine learning algorithms. However, most of the machine learning algorithm expect numerical features vectors of fixed length as input. The Bag of Words representation aims to transform raw texts which are sequences of words of variable lengths, to exploitable numerical vectors.

To that end, we can apply the following pre-processing methods:

- Tokenize the words of the input strings, for instance using white-spaces and punctuation as token separators and one-hot-encode the words,
- count the number of occurrences of each token in each text,
- (optionally) weight the importance of tokens according to their occurrence frequencies in all the texts (TF-IDF method for example, see next definition).

In the rest of the paper, we call vectorization the general process of turning our collection of texts, sentences or subsentences into numerical feature vectors. With this strategy, the input strings are described by word occurrences but ignore the information about the position of the words in the text.

3.3 TF-IDF

Let us define $\mathbf{df}_w$ the document frequency which is the number of texts in the corpus that contain the word w. Then $\mathbf{idf}_w = \log(\frac{N}{\mathbf{df}_w})$ is the Inverse Document Frequency with N the total number of texts.

Let us define $\mathbf{tf}_{w,t}$ the term frequency of word w in text t. It is simply equal to the number of occurrences of w in t.

Finally, Term Frequency-Inverse Document Frequency is a weighting scheme that assign to each word w in each text t a weight computed as:

$$\mathbf{tf} - \mathbf{idf}_{w,t} = \mathbf{tf}_{w,t} \times \mathbf{idf}_w$$

As a consequence, the weight is higher when w occurs many times in text t but within a small number of documents otherwise, which gives it a high discriminating power; lower when the term occurs only a few times in text t, or appears in many texts and is thus less discriminating.[2]

4 Gender prediction

4.1 Methodology

4.1.1 Features

We decided to look for keywords and key subsentences that would help discriminate between female and male victim. To that end, we vectorized the whole texts of the corpus using a TF-IDF method and considering N-grams with N varying from 1 to 4.

We then computed chi-squared ($\chi 2$) stats between the features (vectorization of texts) of the training corpus, and the ground truth victim gender. The scores obtained enabled us to select some N-grams with the highest values for the test chi-squared statistic relative to the classes, which means that they are the most relevant for the classification problem at hand.

Using the N-grams chosen at the previous step, we computed a table counting the proportions for each text of the occurrences of the female vs male version of the sequence (proportion of "née" vs "né" or "subi par madame" vs "subi par monsieur" etc.).
We then replaced the missing values (when a sequence is not in a text), by the average of the proportion in female texts and the proportion in male texts to avoid bias. For example, the average proportion of "subi par madame" vs "subi par monsieur" is 0.977 in texts about female victims and 0.074 in texts about male victims; we therefore replaced the missing values for that N-gram by 0.5255.

Example of results: elle/il née/né madame/monsieur subi par madame/monsieur madame/monsieur a été victime victime madame/monsieur verser à madame/monsieur

Text	elle/il	née/né	mme/m.	subi par mme/m.	mme/m. a été victime	victime mme/m.	verser à mme/m.
0	0.296875	0.166667	0.312500	0.000000	0.000000	0.53125	0.500000
1	0.180124	0.333333	0.121622	0.000000	0.476786	0.00000	0.483493
2	0.527027	0.222222	0.842105	0.525568	0.476786	0.53125	0.483493
3	0.306667	0.150000	0.333333	0.525568	0.476786	0.53125	0.483493
4	0.239130	0.315789	0.500000	0.525568	0.476786	0.53125	0.483493

Table 1: Extract of the data used for gender prediction

4.1.2 Classifier

The classifier chosen for the particular task of predicting the gender of the victim is a Support Vector Machine (SVM) with a Stochastic Gradient Descent (SGD). We used cross-validation with 5 folds and a GridSearch to look for the parameters giving the highest accuracy.

The best parameters found were:

- Hinge loss: assuming the true labels in *y* are encoded with +1 and -1 and *w* is the predicted score of the decision function, the loss is $L_{\text{Hinge}}(y, w) = \max\{1 - wy, 0\}$,
- L_1 penalty: regularization in norm L_1 with a 10^{-4} regularization coefficient,
- less than 1000 iterations.

4.2 Results

This method achieved a **99.74%** average accuracy on the 5 folds, which is particularly good since the classes are balanced (hence the precision and recall are good too).

Transforming the decision scores to probabilities and using a 0.5 decision threshold, we got the following Receiver Operating Characteristic (ROC) curve of female vs male and n.c.:

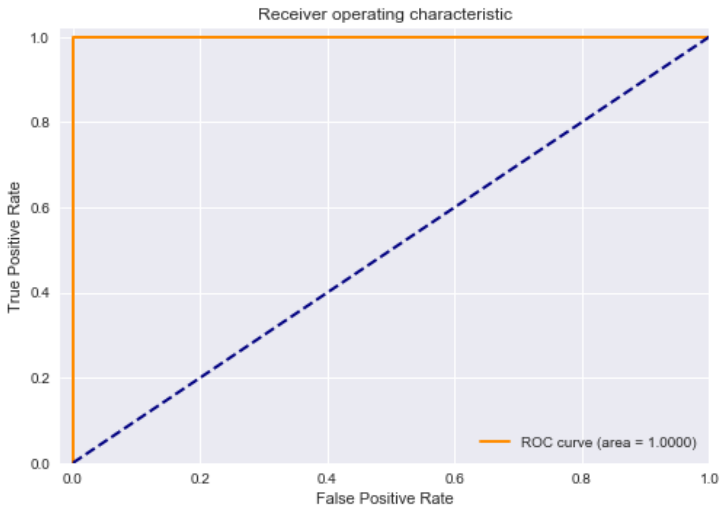

Figure 1: ROC curve of 'female' class against the rest ('male' or n.c.)
The Area Under the Curve (AUC) is rounded to 1.

5 Accident and consolidation dates prediction

The problems of the accident and consolidation dates extraction were resolved independently but with the same method; therefore in the rest of the section 'accident date' will be used to explain the principle but could be replaced by 'consolidation date'.

In that part we separate the problem in three sub-problems:

- Part 1 : Extract all the dates of the texts,
- Part 2 : Identify strings that contain an accident date,
- Part 3 : choose the most probable accident date given the previous data.

For the second part we tried two different methods: with a SVM and using Neural Networks; we will detail the implementation in the following subsections.

5.1 Part 1 - Extraction of all the dates

We used Regular expression (Regex) matching methods to extract the dates in all their potential format. The principle of Regex is to find specific patterns in the text and return the strings found or their positions. In our case, we first split the text in its sentences to search sentence by sentence. Then we looked at specific patterns, for example: *'nn/nn/nnnn'* or *'nn llllll nn'* where the *n* are numbers and *l* letters.

With all the patterns we chose, we could find a total of 27997 dates in the 770 texts of the training corpus. Only 7 out of the 770 accident dates were not found among the extracted dates, and 14 out of the consolidation dates (when not n.c or n.a.).

We also stored the subsentences (size 300 characters centered around the date) from which we extracted the dates. In those sentences we removed the date and replaced it by the word 'date' to avoid a bias due to the presence of a date in the sentence.

5.2 Part 2 - Methodology 1: SVM classifier

We remind that here, the classifier aims to predict for all extracted subsentences with a date if they contain the accident date.

5.2.1 Features

An SVM algorithm cannot take raw text as input nor perform vectorization itself. Therefore we need to perform a vectorization before feeding the obtained numerical features into the SVM. As in the beginning of the gender prediction part, we used a TF-IDF vectorizer. As opposed to the gender prediction though, we do not build an intermediary table with a few keywords but we feed the entire features vectors directly into the classifier which allows more complexity and flexibility.

5.2.2 Classifier

Since the TF-IDF can take some parameters as input, we search for the best parameters jointly with the SVM. To that end (using *scikit-learn*), we use a a GridSearch along with a Pipeline object that allow to optimize both the TF-IDF vectorizer parameters and the SVM ones when it is fed the Vectorizer features results.
As for the gender prediction, we use 5 folds of cross-validation to avoid overfitting.

For the accident date, the best results were obtained with :

- For the TF-IDF:
  - No removing of stopwords,
  - regularization with a norm L_2,
  - N-grams with N from 1 to 4.
- For the SVM:
  - Squared Hinge loss,
  - 'elasticnet' penalty: corresponds to a mixed $0.1L_1 + 0.9L_2$ norm regularization with a 5.10^{-5} regularization coefficient,
  - 'balanced' weights to give more importance to the class 'contains an accident date' which is far smaller,
  - less than 1000 iterations.

For the consolidation date, the best results were obtained with :

- For the TF-IDF:
  - No removing of stopwords,
  - regularization with a norm L_2,
  - N-grams with N from 1 to 4.
- For the SVM:
  - Squared Hinge loss,
  - 'elasticnet' penalty: corresponds to $0.2L_1 0.8L_2$ norm regularization with a 10^{-4} regularization coefficient,
  - 'balanced' weights to give more importance to the class 'contains an accident date' which is far smaller.

5.2.3 Results

For the accident dates, we obtained an average accuracy of 96.52% on the 5 folds on the training dataset. The obtained ROC curve (Area Under the Curve of 0.985) and the confusion matrix for a probability threshold of 0.5 are plotted in Figure 2:

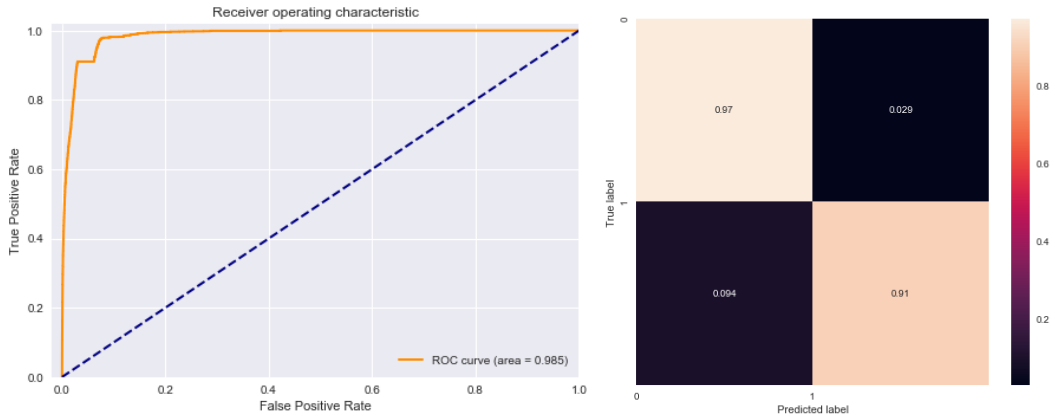

Figure 2: ROC curve (AUC of 0.985) and confusion matrix when predicting if the subsentences contain the accident date

For the consolidation dates, we obtained an average accuracy of 97.50% on the 5 folds on the training dataset. The obtained ROC curve (Area Under the Curve of 0.990) and the confusion matrix for a probability threshold of 0.5 are plotted in Figure 3:

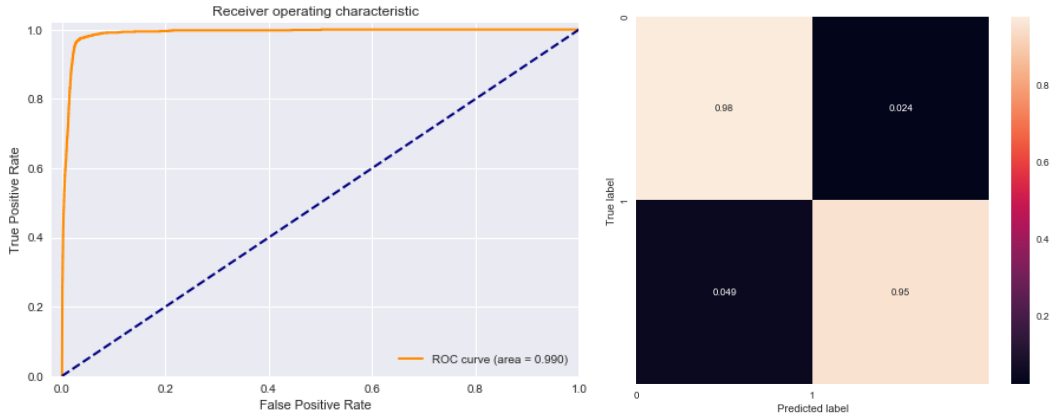

Figure 3: ROC curve (AUC of 0.990) and confusion matrix when predicting if the subsentences contain the consolidation date

5.3 Part 2 - Methodology 2: Neural Network classifier

5.3.1 Features

In the case of Neural Networks, the Embeddings and therefore vectorization of the text toward numerical features is included in the architecture of the model. Credit for the idea of the model goes to [4].

5.3.2 Classifier

Our architecture is composed of an Embedding layer, a Long Short-Term Memory (LSTM) network and a final Dense layer with an activation function to transform the output to probabilities. An example of the global structure is summarized below in Figure 4. Here the embedding is different from the vectorization we used with the SVM: we do not construct a Bag Of Words representation so that it keeps the information of the words positions.

The particularity of the LSTM is that it is a Recurrent Neural Network (RNN) composed of many layers. It allows the LSTM to comprehend the sequences of words, by keeping track of the previous words as context.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 64)          3200000
_________________________________________________________________
lstm (LSTM)                  (None, 64)                33024
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 3,233,089
Trainable params: 3,233,089
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Structure of the model

Cross-validation being time consuming with Neural Networks, we split the training texts randomly in 80% for the training of the model and 20% for the validation.

The best results for both the accident and consolidation date predictions were reached with:

- An RMSprop optimizer of learning rate 0.0015,
- an embedding size of 32,
- 128 neurons in the LSTM,
- a batch size of 64,
- 20 epochs.

5.3.3 Results

The accident date prediction model reached a 97.40% accuracy on the training set and 96.77% on the validation set. The obtained ROC curve (Area Under the Curve of 0.988) and the confusion matrix for a probability threshold of 0.5 are plotted in Figure 5:

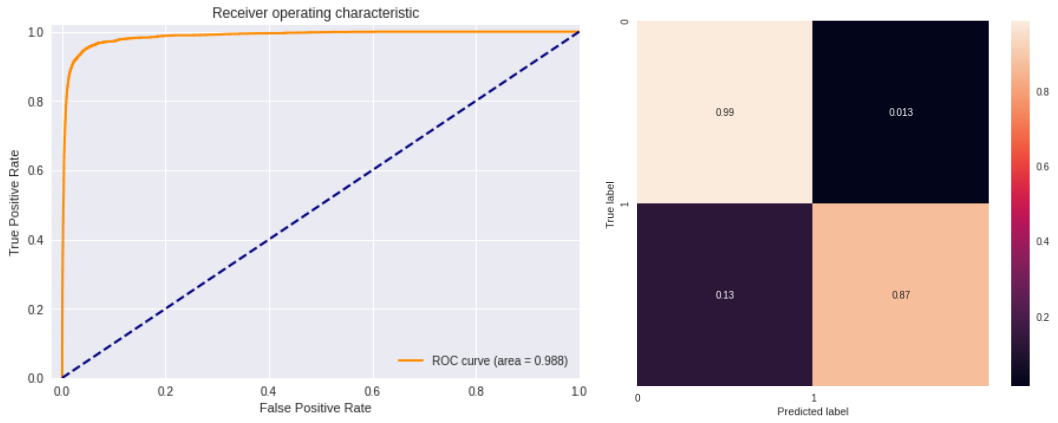

Figure 5: ROC curve (AUC of 0.988) and confusion matrix when predicting if the subsentences contain the accident date

The consolidation date prediction model reached a 98.06% accuracy on the training set and 97.93% on the validation set.The obtained ROC curve (Area Under the Curve of 0.962) and the confusion matrix for a probability threshold of 0.5 are plotted in Figure 6:

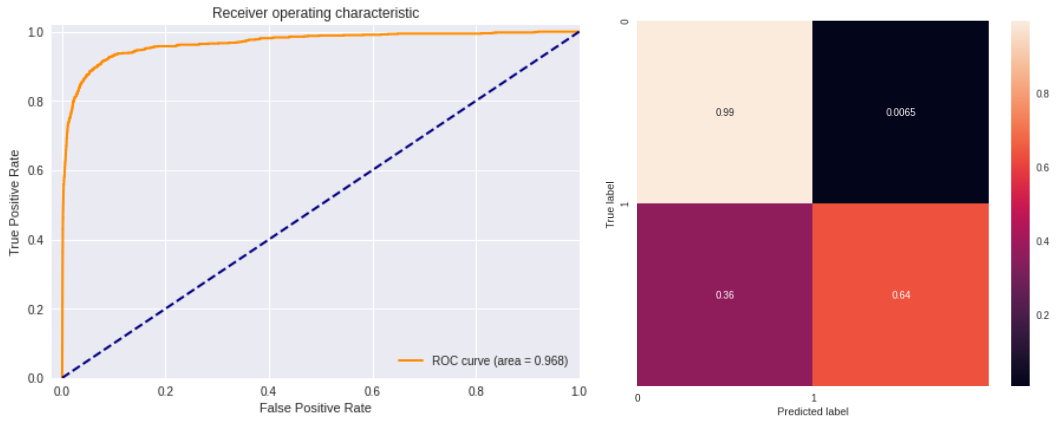

Figure 6: ROC curve (AUC of 0.962) and confusion matrix when predicting if the subsentences contain the consolidation date

In both cases we notice a high accuracy, but an Area Under the Curve not as performing as we could expect. This is due to the imbalanced classes and the fact that the loss of the model is computed without taking the imbalance into account. Therefore, it reached a high accuracy by performing on the 0 class, but it has not a really good recall ($\frac{\text{predicted positive}}{\text{true positives}}$), which we can see in the normalized confusion matrices.

5.4 Part 3 - Prediction of the dates

For each subsentence where we found a date, we use the models of Part 2 to predict if they contain an accident date. That way, the corresponding dates are the potential consolidation dates. For each text and each potential date, we sum the probabilities given to the subsentences they were extracted from. The final predicted accident date is the one with the highest score.

If no subsentence was predicted as containing the accident date, we return 'n.c.' or 'n.a'. The choice between 'n.c.' or 'n.a' is done thanks to an annex SVM classifier (AUC: 0.982) that predicts if the victim is dead, in which case the consolidation date is not applicable ('n.a.').

6 Final results and discussion

With the SVM classifier, the average accuracy for the 3 fields (gender, accident date, consolidation date) was of 88.01% on the training set and 80.21% on the test set.

With the LSTM-based Neural Network, the average accuracy for the 3 fields (gender, accident date, consolidation date) was of 90.13% on the training set; and could not be scored on the test set before the deadline. This approach gives better result that the SVM one, but is much more complex and requires more time and computational power to train. Depending on the time and resources available, the SVM method might be sufficient.

We think that we could achieve a better accuracy with more time to finetune the Neural Network. It might also be interesting to replace the LSTM model by a bi-LSTM (bidirectional) [7], which allows it to gain context for the words not only before but also after. Another potential improvement for the Neural Network is to include weights in the computation of the loss, to compensate for the imbalanced classes, which is not naturally done [6].
The prediction of 'n.a.' for the consolidation date could probably also be improved by predicting better if a victim is dead and by comparing this probability of death to the probabilities of the consolidation dates.

References

[1] n-gram, https://en.wikipedia.org/wiki/N-gram

[2] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. (2008)

[3] Susan Li, https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f (2018)

[4] Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, Long Wang, *An LSTM Approach to Short Text Sentiment Classification with Word Embeddings* [The 2018 Conference on Computational Linguistics and Speech Processing] (2018)

[5] Kim, Y., *Convolutional Neural Networks for Sentence Classification*. Proceedings of the Conference on Empirical Methods in Natural Language Processing, 1746–1751.http://arxiv.org/abs/1408.5882 (2014)

[6] Jason Brownlee, *Cost sensitive Neural Network for imbalancedclassification*, https://machinelearningmastery.com/cost-sensitive-neural-network-for-imbalanced-classification/ (2020)

[7] Shi, Xiaoming, Lu, Ran, *1ttention-Based Bidirectional Hierarchical LSTM Networks for Text Semantic Classification*, [IEEE 2019 10th International Conference on Information Technology in Medicine and Education (ITME)] (2019)