Anonymization of Italian judgments

Gianmarco Pepi

January 18, 2022

Contents

1 Introduction

The goal of this project is to train a model for anonymization of text, in particular I want to build a model to extract the defendant's name from legal judgements.
In this article (link) they tested 4 Named Entity Recognition algorithms to anonymize commercial courts' decisions obtaining best results with Flair, bi-LSTM character-based model.
For this task I believe that the relationship between entities is fundamental because in a legal judgements I am not interested in the anonymization of all entities but only of the defendant entities, so I decided to use relation extraction to solve this task.
In section 2, I explained where I got the data from, the libraries I used to do the data scraping and how I built the scrapers.
Subsequently I analyzed and cleaned the dataset and before I could train the model I manually annotated the legal judgements.
In the 3rd section I built, trained and evaluated the model for relation extraction.
Finally, I created an interactive dashboard via the dash library.

2 Data Crawling

Unfortunately there is no labeled dataset to train the model I wanted to build so I first had to look for the institutional Websites that publish the legal judgements and I have identified two resources:

- **Giustizia Ammnistrativa:** On this site it is possible to download the administrative judgements in html format.
- **Corte Suprema di Cassazione:** As for the judgements of Corte di Cassazione you can download their pdf on this website

To get a large enough dataset I had to build two scrapers basically using Selenium and BeautifulSoup libraries.

2.1 Html Scraper

To obtain the text of a judgements starting from a single link I used 2 python libraries in combination:

- **requests** A library to sent HTTP requests. Furthermore I used fake_user library to avoid being blocked.
- **BeautifulSoup** Used to extract text from HTML file returned from python requests.

For the automated extraction of multiple links I had to use selenium both because the link to the html of the judgements is saved on JavaScript elements and to be able to simulate human-like mouse movements.
I managed to download 948 legal judgements and I stored them on a csv file.

2.2 PDF scraper

Obtaining the judgements of the *Corte di Cassazione* was much more difficult. First of all because I can't use BeautifulSoup library to obtain text because that judgments are loaded as pdf file, so I had to download the judgment and use the PyPDF2 library to extract the text. To download the pdf files I used python request as done previously but changing the proxy cyclically to avoid stopping after 50 requests of downloading.
Through the Selenium library I created the scraper by automatically changing the page as done previously and I obtained nearly 200 legal judgements.

3 Data Preparation

In this section I showed how I analyzed the data, how I cleaned them, the technology used for their annotation and finally their formatting to be transformed into a data set useful for training the next model.
While annotating I realized that using all the scraped data was not feasible so I decided to focus on the criminal judgements related to the *Corte di Cassazione*.
Reading this spaCy project I decided to train my model using spaCy-transformers so I have to transform my dataset in such a way as to be able to obtain training set,test set and validation set as spaCy binary files as shown Data transformation subsection.

3.1 Data Understanding

From data scraping I obtained data set with 2 columns, one containing the link to the related pdf judgements and the other containing the judgements text without any cleaning, so in this step I focused on text column. First of all from the histogram shown in fig 1 I noticed there are judgements too long with respect to the mean judgements length so I decided to eliminate judgements with more than 40 thousand characters. As for the cleaning of the text I did it after the annotation.

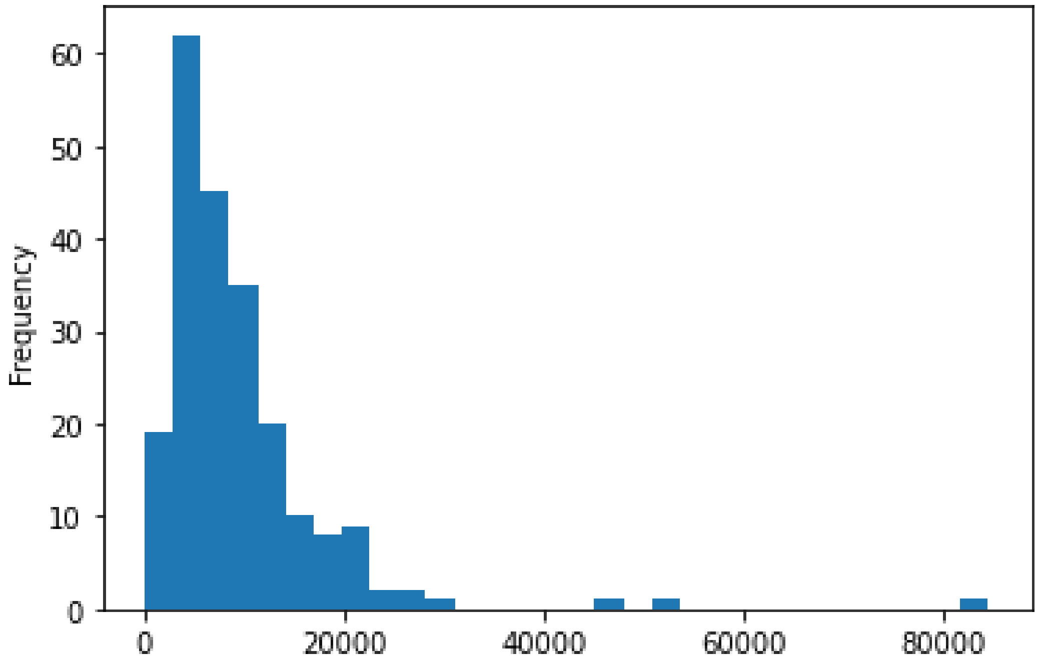


Figure 1: Distribution of the length of the judgements

Although from the previous histogram it is not clear there are sentences with less than 200 characters which represent a default pdf for the sentences currently obscured by the court so I have also deleted these sentences since they do not contain data to be anonymized.
Finally I created a new column called subtext that contains a portion of 500 characters of the entire judgement containing all the information useful for our model because usually after the string '1.' there is information on the accused, the judge and the lawyer.

3.2 Data Annotation

I decided to annotate entities and their relation with this Web Tool because it is open source but I hadn't considered the importance of the output format in fact annotation tools such as *prodigy* are much better integrated with the spaCy library. In data Transformation subsection I explained how I had to convert the output of this web annotation tool before I could convert it to binary spaCy file.
However at this stage I annotated the entities that represented the defendants, the lawyer and the persons or institution that issued the judgement against the defendant. Then I annotated the relations between this entities choosing from 'DIFENDE' and 'GIUDICA'. I did this by dividing the dataset into 4 samples and then merging the outputs. The output of this annotation tool is a json file with three properties:

- *SentText* containing the text of input
- *EntityMentions* containing a list of entities
- *RelationMentions* containing a list of dictionary each of which represents a relationship

Subsequently I read json files as pandas dataframe and I merged them eliminating those that I was unable to annotate due to lack of information in the input obtaining a data set of 171 rows. Before saving the dataset I cleaned the judgements by eliminating legislative terms and abbreviations that, due to the presence of dots, could cause a wrong contextualization by the Bert Embedding.

3.3 Data Transformation

From the tutorial contained in this project I found the code to convert the json file into a spaCy binary file but as mentioned in the previous section the output of the annotation tool I used was not in the right format so I created a function namely *create_dict* in DataTransformation notebook that convert a row of dataframe in a dictionary that will form the json file to be converted into spaCy binary file. In this function I combined other functions to tokenize the sentences and to take the entities that are closest in the text.
I splitted the data set into training set, test set and validation and then I converted them into spaCy binary files.

4 Building the Model

As said before I built a custom trainable relation extraction component from this spaCy project in which two different pipelines are built and runned separately:

- **rel_tok2vec pipeline:** ["tok2vec", "relation_extractor"]
- **rel_trf pipeline:** ["transformer", "relation_extractor"]

Both pipelines are implemented through the spaCy library and the difference between the two pipelines is the first component representing an Embedding layer. For the *tok2vec* component there are already Italian pipelines optimized for tok2vec and I chose *it_core_news_lg* pipeline.
For the *transformer* component there are still no Italian transformer pipeline so I left the pipeline on English namely *en_core_web_trf* and I changed the transformer model and used a multilingual language model namely *xlm-roberta-base.*
For the *transformer* component the architecture used is *TransformerModel.v1* taken from the *spaCy-transformers* package. This package provides spaCy model pipelines that wrap

Hugging Face's transformers package, so you can use them in spaCy.
The second component for each pipeline take in input the list of tokens representing the Document embedding in the vector space. Then a *Pooling* layer will be applied to summarize the token vectors into entity vectors and finally, we need method that generates pairs of entities that we want to classify as being related or not. This method takes any two entities from the same document, as long as they are within a maximum distance (in number of tokens) of each other. I changed maximum distance value by setting it to 75, which is the maximum length of the judgements I scraped.
I changed these parameters as explained so far by simply cloning this repository and changing the two configuration files inside the Configs folder.

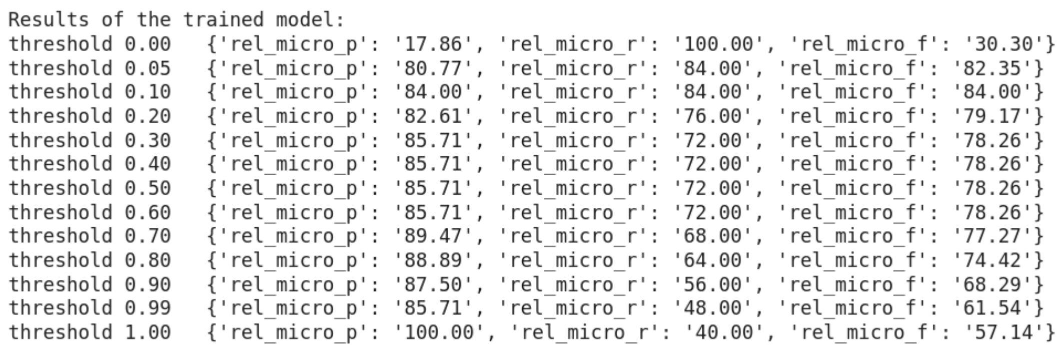
4.1 Models Training and Evaluation

After changing the configuration file parameters and adding the Data folder with our training set, test set and validation set in binary format, we can proceed to train the models into the two pipelines and evaluate them. Simply running following cells of code respectively for *rel_tok2vec* and *rel_trf* pipelines :

```
!spacy project run train_cpu
!spacy project run evaluate

!spacy project run train_gpu
!spacy project run evaluate
```

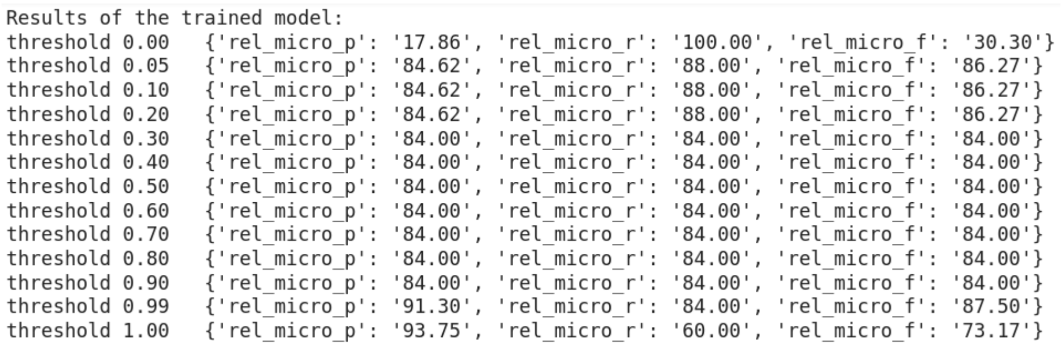
The model of the second pipeline is trained using CUDA GPUs because is based on transformers. The evaluation metrics for the two pipelines are *precision, recall and f1-score* and in fig 2 and fig 3 these metrics are shown as the threshold changes. The threshold parameter represent the threshold above which a prediction is seen as True.



```
Results of the trained model:
threshold 0.00   {'rel_micro_p': '17.86', 'rel_micro_r': '100.00', 'rel_micro_f': '30.30'}
threshold 0.05   {'rel_micro_p': '80.77', 'rel_micro_r': '84.00', 'rel_micro_f': '82.35'}
threshold 0.10   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.20   {'rel_micro_p': '82.61', 'rel_micro_r': '76.00', 'rel_micro_f': '79.17'}
threshold 0.30   {'rel_micro_p': '85.71', 'rel_micro_r': '72.00', 'rel_micro_f': '78.26'}
threshold 0.40   {'rel_micro_p': '85.71', 'rel_micro_r': '72.00', 'rel_micro_f': '78.26'}
threshold 0.50   {'rel_micro_p': '85.71', 'rel_micro_r': '72.00', 'rel_micro_f': '78.26'}
threshold 0.60   {'rel_micro_p': '85.71', 'rel_micro_r': '72.00', 'rel_micro_f': '78.26'}
threshold 0.70   {'rel_micro_p': '89.47', 'rel_micro_r': '68.00', 'rel_micro_f': '77.27'}
threshold 0.80   {'rel_micro_p': '88.89', 'rel_micro_r': '64.00', 'rel_micro_f': '74.42'}
threshold 0.90   {'rel_micro_p': '87.50', 'rel_micro_r': '56.00', 'rel_micro_f': '68.29'}
threshold 0.99   {'rel_micro_p': '85.71', 'rel_micro_r': '48.00', 'rel_micro_f': '61.54'}
threshold 1.00   {'rel_micro_p': '100.00', 'rel_micro_r': '40.00', 'rel_micro_f': '57.14'}
```

Figure 2: Evaluation metrics of rel_tok2vec pipeline

The results of the evaluation show that the model based on transformers is better and therefore I decided to use only the second model for the visualization in the next section.
Finished training and evaluation both pipelines will go to save the best model as an Italian pipeline optimized for relation extraction. To apply this pipeline to any textual document I have to load two Italian pipelines with the *spacy.load()* command which are my custom model and the Italian pipeline *it_core_news_lg*, giving the text document as input to this last pipeline and passing its output to my custom model.



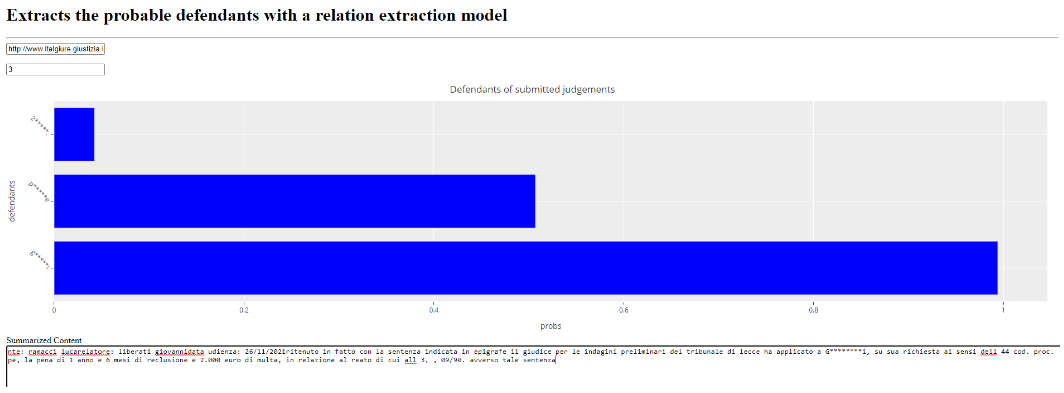
```
Results of the trained model:
threshold 0.00   {'rel_micro_p': '17.86', 'rel_micro_r': '100.00', 'rel_micro_f': '30.30'}
threshold 0.05   {'rel_micro_p': '84.62', 'rel_micro_r': '88.00', 'rel_micro_f': '86.27'}
threshold 0.10   {'rel_micro_p': '84.62', 'rel_micro_r': '88.00', 'rel_micro_f': '86.27'}
threshold 0.20   {'rel_micro_p': '84.62', 'rel_micro_r': '88.00', 'rel_micro_f': '86.27'}
threshold 0.30   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.40   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.50   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.60   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.70   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.80   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.90   {'rel_micro_p': '84.00', 'rel_micro_r': '84.00', 'rel_micro_f': '84.00'}
threshold 0.99   {'rel_micro_p': '91.30', 'rel_micro_r': '84.00', 'rel_micro_f': '87.50'}
threshold 1.00   {'rel_micro_p': '93.75', 'rel_micro_r': '60.00', 'rel_micro_f': '73.17'}
```

Figure 3: Evaluation metrics of rel_trf pipeline

5 Visualization

For the visualization part I decided to create an interactive dashboard using the Dash library. Graphically it is presented as a very simple dashboard but I have tried to show how the custom model I have obtained works.
In the dashboard there are two text areas for inputs, one to paste the link to the pdf of the judgement you want to anonymize and the other to enter the number of possible defendants k with the highest probability. As shown in the fig 4 below there are 2 outputs:

- **Bar plot:** with the k most eligible defendants and their probability
- **Text area:** a piece of sentence with the anonymization of only the most eligible defendant



Extracts the probable defendants with a relation extraction model

http://www.italgiure.giustizia.i

3

Summarized Content

nte: ramacci lucarelatore: liberati giovannidata udienza: 26/11/2021ritenuto in fatto con la sentenza indicata in epigrafe il giudice per le indagini preliminari del tribunale di lecce ha applicato a G********i, su sua richiesta ai sensi dell 44 cod. proc. pe, la pena di 1 anno e 6 mesi di reclusione e 2.000 euro di multa, in relazione al reato di cui all 3, , 09/90. avverso tale sentenza

Figure 4: Dashboard to anonymize

For the need to use the GPU I had to run the dash app on Colab so I had to use the ngrok library to create a tunnel to connect my local host to Colab server.