Modeling the graph of french law

Antonin PERONNET
Télécom Paris
antonin.peronnet@telecom-paris.fr

Nils Holzenberger
Télécom Paris
nils.holzenberger@telecom-paris.fr

Abstract

1 Introduction

In the last 5 years, there has been a fast increase of natural language processing research in the field of law. This research focuses in particular on classification, information extraction and information retrieval.

The question of how the law should be represented in a computer is crucial. First, this representation can impact the accuracy of all legal tasks. But second and perhaps most importantly, the way law is represented enforces its usage and evolution. Broad ways to represent the law includes pure text, graph-based approaches and formal rule-based approaches.

In a lot of domains, ontologies have emerged as one of the best way to respresent data. Ontologies can represent diverse knowledge, link information, be checked, and evolve with time. Ontologies for law have already been discussed, and graph-based approaches for NLP show promising results. That said, there have been very few attempts to represent law corpuses as ontologies. In particular, state law remains almost exclusively text-based.

Building any type of ontology involves Named Entity Recognition (NER). For the law, this task has some specificities that make it challenging.

Our contributions are as follows: 1) Defining ideal legal entities and discussing their properties 2) Creating a an approach for legal entity extraction based on grammatical tree similarity 3) Creating a AB-X testing benchmark for legal entity extraction in the case of french Law.

2 Background

As of may 2024, the french law is made up of approximatively 800k articles identified as "currently applicable". Among them, 160k (20%) is written in one of the 77 french codes.

Given this massive number of articles, any approach that involves information retrieval in the law needs to preprocess parts of the corpus in some ways.

The first and most straightforward way is language modeling. Some LLMs have been fined-tuned on the law and trained on specific tasks. As this approach works very well to improve legal reasonning [legal bench], this approach has important limitations. It requires massive data, and some vocabulary specific to a part of the law may not be understood by the model. LLMs still suffer a lot from halucinations, and leglal knowledge is no exception. Lastly, legal knowledge in LLMs cannot be edited, and updating them involves retraining the entire model.

Another approach is Retrieval Augmented Generation (RAG). For any legal task, an embedding key is calculated for each article, and then articles are retrieved based on this key. This can greatly reduce hallucinations, but embeddings have other problems. Embeddings can be sensitive to writing style, and are not interpretable.

To make information retrieval more accurate and interpretable, a well-known tool is ontology.

In an ontology, entities are linked by relations, and the resulting graph can be efficiently queried.

There is a wide variety of schemes proposed for legal ontologies, depending on the legal subdomain.

Knowledge about entity recognition is hard to transfer: what we mean by an "entity" or a "relation" is very specific to the field (medical, geography ...)

3 Legal Entity Recognition

Machine learning approaches have been developped for named entiy recognition, but they are based on labeled data that are not available for the law.

The law has a specificity: it is both **descriptive** and **normative**. Meaning that some entities are supposed to exist in the real world, and some of them are defined by the law.

Another specificity: noun verbs are often nested (see figure 2). In such cases, it is hard to know which entity sould be considered.

- concepts like "being mandatory", "being forbiden", "paying a fine" that are not named but very relevant.

We define a Legal Entity as a concept having the following properties. 1) it can be expressed unambiguously as a verb phrase or a noun phrase. We will refer to it as a "cannonical form" of the entity. 2) each time the entity appears in the text, it can be easily replaced by another entity 3) it can be used intuitively by a lawyer to find relevant articles.

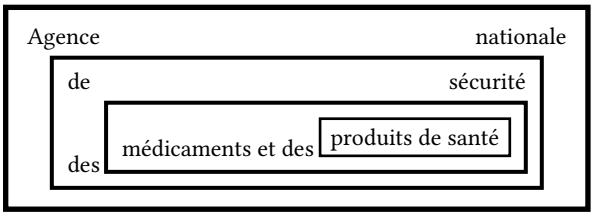


Figure 1: Nested structure of legal entities: "Agence nationale de sécurité des médicaments et des produits de santé"

Since the last point of the definition is very unformal and subjective, we define a measure of the information content of an entity.

Given a corpus D and a probabilistic model P_0, the information content of a collection of entities $(e_1, ..., e_n)$ is

$$H_m(e_1, ..., e_n) = \frac{1}{|D|} \sum_{t \in D} \log_2 \left(\frac{P_0(t)}{P_0(t \setminus e_1, ..., e_n)} \right)$$

Where $P_0(t \setminus e_1, ..., e_n)$ is the probability of t where the words refering to $(e_1, ..., e_n)$ are masked.

If we denote by I the set of indices of the words refering to $(e_1, ..., e_n)$ in the text, we can write

$$P_0(t \setminus e_1, ..., e_n) = P_0(\{t' \mid \forall i \notin I, t_i = t'_i\})$$

In other words, it is the probability of all observations compatible with the non-masked positions.

This measure reflects the information content of the set of selected entities: it measures the amount of information lost by a reader if we remove the words refering to the entities.

[1]

We can finally define the task of Legal Entity Recognition (LER).

Given a corpus D and an integer n, label each token position in the texts of D such that:

- all tokens with the same label relate indeed to the same entity
- the information content score is maximal

4 Preliminary results

To understand the behaviour of H_m, we can use the most simplistic probability model: the unigram model.

Let's suppose P is unigram and we are only allowed to select words as entities.

$P(s) = \prod_i P(w_i)$

Let $I = \{e_1, ..., e_n\}$ be the candidate set of entities.

$$\log_2 \left(\frac{P(t)}{P(t \\ e_1, ..., e_n)} \right) = \log_2 \left(\frac{\prod_i P(w_i)}{\prod_{w_i \notin I} P(w_i)} \right) = - \sum_{w_i \in I} \log_2 P(w_i)$$

$$H_{m(e_1,...,e_n)} = \sum_{t \in D} \sum_{w_i \in I} \log_2 P(w_i) = \sum_j \#(e_j) \log_2 \left(\frac{1}{P(e_j)} \right)$$

But as soon as we chose a slightly more complicated probability model (even bigram), the computation becomes intractable.

For example, someone might try to mask a candidate in a sentence and use a LLM to compute the probability, but candidates are not independent: if we have 2 potential candidates e_1 and e_2 such that e_2 appears whenever e_1 appear, then $H_{m(e_1,e_2)} \approx H_{m(e_1)}$

5 Entity extraction algorithm

Our algorithm consists of 3 phases: 1) selecting occurences and computing their scores 2) clustering the occurences into candidate entities 3) optimizing

5.1 Extracting candidates

The first challenge consist in finding potential entities.

Following the criteria 1) and 2) of our definition, we want to identify self-contained noun and verb phrases.

For this, we will use a standard NLP pipeline with part of speech (POS) tagging and dependency parsing.

A candidate occurence is any token in the text marked with a POS that is either NOUN or VERB, along with its entire subtree. In addition, to respect criteria 2), we filter out occurences that have too many nouns and verbs[2]. We chose the threshold of 7 nouns and verbs, because it is considered as the maximal number of items someone can hold in working memory (criteria 3)

5.2 Subtree probability

Ideally, we would like to approximate P_0 in the lawyer's head. But to be able to approach the optimization problem, we chose a simpler model instead.

Instead, we will make the hypothesis that most of the meaning of an entity is captured by its syntaxic structure.

We will use dependency parsing to represent each one of the sentences in our corpus.

Given a sentence $w_0, ..., w_n$ with $m + 1$ nodes, the dependency parsing of s is a rooted tree T with root at positino r

We will make the following simplifying assumptions:

- The arity of each node in the tree is IID according to some given distribution $P_{\text{arity}(\cdot)}$, independent from the words
- The probability of the word of a given node only depends on its parent according to $P_{\text{word}(\cdot \mid \cdot)}$

Thus, for a tree T with node arities a_i and arcs $\{i \to j\}$ (parent to child),

$$P(T) = P_{\text{word}(w_r)} \prod_{i \to j} P_{\text{arity}(a_i)} P_{\text{word}(i \mid j)}$$

In a similar way as the unigram model, we can derive H_m (see appendix). D is now a set of sentence trees.

$$H_{m(e_1,...,e_n)} = \sum_{T \in D} \sum_{i \to j} \log_2 \left(P_{\text{arity}(a_i)} P_{\text{word}(i \mid j)} \right)$$

5.3 Optimization algorithm

5.4 Merging

TODO

5.5 Additional step

6 Creation of the benchmark

6.1 Motivation

- we suppose there is a ground truth
  - most people can identify when 2 articles have an entity in common
- possible to create a benchmark
- challenges:
  - no dataset for this specific task
  - discussion of other data sources:

[1] Of course, the measure depends on P_0. Ideally, we would like P_0 to be as close as possible to the probability in the lawyer head.

[2] This is necessary, otherwise the algorithm will favor large chunks of text because they are surprising

    - retrieval (not really relevant data available)
    - judgement prediction (very noisy, cite each other and not just the law)
    - link prediction (limited)
  - 2 different approaches can put different labels on the same entities: we chose a graph similarity approach

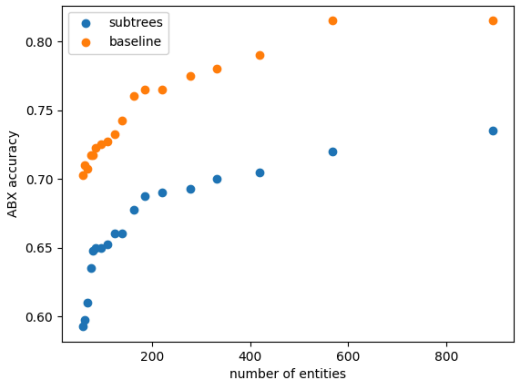
6.2 Citation network

- seen as a citation network (multiple papers on this)
- can be (somewhat) explained by a combination of entities in common
- sometimes, entities in common without citations, or reverse
- possible to transform the bipartite graph into a weighted graph
  - paper https://www-complexnetworks.lip6.fr/~magnien/Publis/32PredictionDynak/article.pdf
  - information lost
  - important question: can 2 very different bipartite graphs produce the same weighted graph ? probability ?

7 Results

7.1 Baseline

7.2 ABX



8 Discussion

- our approach does not penalize very large entities.

9 Appendices

- hyperparameters
- data preprocessing

10 Supplementary material

- extracted entities