Mapping tax law to executable computer code

Nils Holzenberger (Télécom Paris) & Peter Fratrič (HEC Paris)

1 Context

This M2 research internship will take place under the supervision of Peter Fratrič and Nils Holzenberger, in the DIG team, at Télécom Paris, 19 place Marguerite Perey, 91120 Palaiseau. The intended duration is 6 months, with a start date between February and May 2024. The project will be carried out in the context of an ongoing research collaboration with David Restrepo Amariles at HEC Paris (Fratric et al., 2024).

The CodeReCivil project aims to discover tax loopholes (Blair-Stanek et al., 2022) by simulating a legal and economic environment, and letting agents interact with the environment and possibly one another, guided by reward maximization (Fratric et al., 2024). This exploration leads to the discovery of new schemes of behavior, including previously unknown tax loopholes. The ability to automatically turn a set of laws and regulations into a framework of computer-executable rules, as described here, is a cornerstone of the CodeReCivil approach.

We are looking for an excellent student currently in a Master's program, with (1) a solid background in Natural Language Processing (NLP) and Machine Learning, and (2) good foundations in formal languages.

2 Summary

Legislation can be viewed as a body of prescriptive rules expressed in natural language. Early research on legal artificial intelligence has shown that laws can be expressed in computer-executable form (McCarty, 1976). This makes it possible to automate case-based reasoning (Popp and Schlink, 1974), computing taxation of stock (Hellawell, 1980) or income (Sherman, 1987), reasoning about civil cases (Satoh et al., 2010), and determining rights (Bench-Capon et al., 1987; Sergot et al., 1986), *inter alia*. This approach is still relevant, with the ongoing effort to translate French tax law into the Catala programming language (Merigoux et al., 2021), and the law as code movement (Mohun and Roberts, 2020) and computable contracts (Servantez et al., 2023).

The main bottleneck of this approach is the difficulty of translating law into code. In the case of the Catala language, programmers must work jointly with lawyers, in a pair-programmming setting. This makes large-scale translation expensive and slow. In human-authored translations of laws into computer code, including in the case of Catala, the mapping between the language of the law and the computer code is complex, making automatic approaches fail (Morgenstern, 2014; Pertierra et al., 2017; Vu and Nguyen, 2018).

In this project, we seek to automate the mapping of law to a computer-executable language. A straightforward approach is to use code already written in the CodeReCivil project to bootstrap a code generation model, using a Large Language Model (LLM). If that proves to be limited, the project could be extended to design a meaning representation able to capture the semantic content of tax and business law, while being closely aligned with the language of the law. This will result in a representation that can be (1) used by automatic reasoning engines, and (2) generated automatically from the legal text.

3 Objectives and expected results

The goal is to efficiently represent the semantics of tax law, to automatically map legal text to computer-executable code. To develop and evaluate the proposed approach, we will rely on the SARA dataset, a benchmark for reasoning with tax law and legal cases (Holzenberger et al., 2020). The SARA dataset contains legal cases, with which to test how well the proposed formalization enables automatic legal reasoning.

The first step of this project will be to refine the proposed mapping. This will be done by manually translating a subset of the laws in the SARA dataset, and identifying necessary design choices.

The second step is to explore how well current NLP techniques can translate legal language into the proposed representation. This is a problem of code generation (Chen et al., 2021) and semantic parsing (Pertierra et al., 2017). We will recast the SARA dataset as a code generation benchmark, by treating its laws as code specification, and legal cases as unit test cases, providing us with a way of measuring execution accuracy of the generated code. We will tackle this code generation problem using state-of-art methods involving Large Language Models: Codex (Chen et al., 2021) and AlphaCode (Li et al., 2022), LORA fine-tuning (Hu et al., 2022), and constrained decoding (Shin et al., 2021). The PI's research team has extensive expertise in those areas.

Finally, we will try out the approach on real-world laws and regulations. We expect this internship to lead to a publication in an NLP conference. With the practical and scientific significance of automatically translating legal text into computer code, we expect this project to uncover multiple key challenges. These could form the basis for a PhD thesis.

Bibliographic References

Bench-Capon, T. J. M., G. O. Robinson, T. Routen, and M. J. Sergot, 1987: Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *ICAIL '87*.

Blair-Stanek, A., N. Holzenberger, and B. Van Durme, 2022: Shelter check: Proactively finding tax minimization strategies via ai. *Tax Notes Federal, Dec*, 12.

Chen, M., J. Tworek, H. Jun, et al., 2021: Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Fratric, P., N. Holzenberger, and D. Restrepo Amariles, 2024: Rules2lab: from prolog knowledge-base, to learning agents, to norm engineering. In *EUMAS 2024*.

Hellawell, R., 1980: A computer program for legal planning and analysis: Taxation of stock redemptions. *Columbia Law Review*, 80(7), 1363–1398.

Holzenberger, N., A. Blair-Stanek, and B. Van Durme, 2020: A dataset for statutory reasoning in tax law entailment and question answering. In *NLLP Workshop at KDD 2020*.

Hu, E. J., Y. Shen, P. Wallis, et al., 2022: Lora: Low-rank adaptation of large language models. In *ICLR 2022*.

Li, Y., D. H. Choi, J. Chung, et al., 2022: Competition-level code generation with alphacode. *CoRR*, abs/2203.07814.

McCarty, L. T., 1976: Reflections on taxman: An experiment in artificial intelligence and legal reasoning. *Harv. L. Rev.*.

Merigoux, D., N. Chataing, and J. Protzenko, 2021: Catala: a programming language for the law. *Proceedings of the ACM on Programming Languages*, 5(ICFP), 1–29.

Mohun, J. and A. Roberts, 2020: Cracking the code: Rulemaking for humans and machines.

Morgenstern, L., 2014: Toward automated international law compliance monitoring (tailcm). Tech. rep., LEIDOS HOLDINGS INC RESTON VA.

Pertierra, M. A., S. Lawsky, E. Hemberg, and U. O'Reilly, 2017: Towards formalizing statute law as default logic through automatic semantic parsing. In *ASAIL @ ICAIL 2017*.

Popp, W. G. and B. Schlink, 1974: Judith, a computer program to advise lawyers in reasoning a case. *Jurimetrics J.*.

Satoh, K., K. Asai, T. Kogawa, et al., 2010: Proleg: an implementation of the presupposed ultimate fact theory of japanese civil code by prolog technology. In *JSAI International Symposium on Artificial Intelligence*.

Sergot, M. J., F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory, 1986: The british nationality act as a logic program. *Communications of the ACM*, 29(5), 370–386.

Servantez, S., N. Lipka, A. F. Siu, et al., 2023: Computable contracts by extracting obligation logic graphs. *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*.

Sherman, D. M., 1987: A prolog model of the income tax act of canada. In *ICAIL 1987*.

Shin, R., C. H. Lin, S. Thomson, et al., 2021: Constrained language models yield few-shot semantic parsers. In *EMNLP 2021*.

Vu, T. S. and L. M. Nguyen, 2018: An empirical evaluation of AMR parsing for legal documents. In *JSAI-isAI @ JURISIN 2018*.