MIT Computational Law Report

The Linguistic Etch A Sketch

Megan Ma

Published on: May 14, 2021

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)

First and foremost, welcome to the column! Whether you deliberately sought this out or stumbled upon it, I am thrilled to bring you into this humble space where we explore issues of core linguistics in computational law. By way of brief introduction, my name is Megan. I am one of the editors of the MIT Computational Law Report and will be your primary guide through this adventure.

I will open by clarifying that the column is not focused on computational linguistics. In fact, one of the aims is to reconsider and rethink the application of computational linguistics in the translation of law to code. The rise in LegalTech and broader movement to increase accessibility to legal services have provoked scholars and practitioners alike to determine whether the law is indeed computable. The purported benefits of making legal processes quantifiable[1] have been widely discussed. Importantly, it has been recognized that law can function as an algorithm.[2] With monumental advances in natural language processing (NLP) and natural language understanding (NLU), these techniques have made noticeable impact in the legal industry. From legal document automation, contract and predictive analytics to word embeddings and transformer models for legal research, LegalTech has further cemented the irrefutable bond between law and language.

Often regarded as a technical language itself, to state that law has language at its core is, in effect, an understatement. Since the 1960s, the structural dynamics between law and language have been explored at length, and in particular, a focus on the arcane language of written legal texts.[3] However, it was not until the 1980s that language was understood as the medium through which the law does its work. Notable scholars like Brenda Danet focused on the "strategic significance of alternative ways of naming and categorizing objects and actions"[4] that separate legal from every day (ordinary) language. This begs the question:

> ***Why is there a distinct legal language and how does this affect migrating legal processes from analog to digital?***

The intention of the column is then to explore the parameters and limits of legal expression. Moreover, this column will investigate the mediums of communicating and representing legal knowledge. As opposed to computational linguistics, the column reflects on computation *and* language. Returning to the essential pillars of syntax, semantics, and pragmatics, we consider how natural language is shaped and deconstructed. The hope is that this deeper analysis of how natural language is treated

will demonstrate the ways that legal language can be reconciled with computational methods.

In the coming series, we will venture into the rabbit holes of linguistics and the computational techniques regarded as their parallels. Starting from syntax, we will introduce core tenets of sentence structure, diving into generative grammars, constituents, and dependency trees. We then progress to meaning, specifically how meaning is formed. Semantics views the meaning of sentences as sets of worlds that share the same truth conditions. Pragmatics, on the other hand, factors the context inferred and the accounts of "additional meaning." While the former is built on propositional calculus and predicate logic, the latter is built on reference, presupposition, and discursive performance. In short, semantics is predominantly context-independent while pragmatics is context-dependent.

On the other hand, we consider their counterparts in programming; beginning with regular expressions and context-free grammars designed for syntax. We then advance into attribute grammars, often used to provide context-sensitivity when defining the semantics of a programming language. Perhaps the most exciting chapters will reflect on abstraction and object-orientation used to classify and bridge concepts within the language. From the fundamentals, the column will turn to knowledge representation, ontology, and complexity. Equally, we will analyze literary mechanisms, such as metaphor and analogy, used to situate and contextualize meaning.

Ultimately, the column hopes to unpack whether the continued focus on syntax and semantics in computational linguistics has removed an essential component of legal language – the role of context in interpretation and, in effect, a "pragmatics" in code. This has evident implications in the translation of natural to machine-readable language. Understanding the "Law of Interpretation," or better, how to reason with legal texts is one of the most fundamental and oldest questions in legal practice. Should machine-readable language fail to account for context, this may run the risk of conceptual slippage, and inadvertently, change the functional character of the law.

We hope that this column will engage with notions of computation in an unconventional framing. Notably, it seeks to debunk systemic assumptions on incongruency between law and computation by redirecting the focus to the unique medium of language. I look forward to your future comments and your participation on this journey!

For a sneak peek of what's to come, I'd encourage you to take a glance at some of these readings:

> Mark C. Marino, *Critical Code Studies*, (MIT Press 2020).
>
> Betty J. Birner, *Language and Meaning* (Routledge 2018).
>
> Peter Goodrich, *Legal Discourse: Studies in Linguistics, Rhetoric and Legal Analysis* (Palgrave Macmillan 1987).
>
> James Boyd White, *The Legal Imagination* (The University of Chicago Press, 1985).

Footnotes

1. Daniel W. Linna Jr., The Future of Law and Computational Technologies: Two Sides of the Same Coin, MIT Computational Law Report (2019). ↩

2. Alex "Sandy" Pentland, A Perspective on Legal Algorithms, MIT Computational Law Report (2019). ↩

3. David Mellinkoff, The Language of the Law (1963). ↩

4. Brenda Danet, Language in Legal Process, 14 L. & Soc. Rev. 445 (1980) ↩