Integrating Natural Language and Formal Analysis for Legal Documents

Shaun Azzopardi
Department of Computer Science
University of Malta
Email: shaun.azzopardi@um.edu.mt

Albert Gatt
Institute of Linguistics
University of Malta
Email: albert.gatt@um.edu.mt

Gordon Pace
Department of Computer Science
University of Malta
Email: gordon.pace@um.edu.mt

I. Introduction

Many fields of inquiry that have traditionally fallen under humanistic studies have benefited from the development of language technologies. For example, computational linguists have investigated several aspects of literary text and, more generally, "creativity", yielding important insights into the mechanisms underlying the creation of such texts [1]. One area of the humanities that has received far less attention in the NLP community is the field of legal studies.

The legal profession is wide and varied, however a good amount of legal work involves the drafting of documents, specifically contracts. Thus contracts are themselves linguistic artefacts and amenable to NLP techniques such as keyword and named entity extraction. Tools leveraging such NLP techniques to complement technical knowledge, have the potential to do for the drafting of contracts what intelligent design solutions have done for architects and engineers.

Contracts have drawn the attention of researchers interested in the formalisation of some of their features, such as norms, rights and obligations, often using some form of deontic logic [2], [3]. Such formalisations constitute an abstraction of core aspects of the content of a contract, supporting reasoning and detection of errors and/or conflicts. However there remains a gap between the linguistic "surface" of a contract document, and its underlying structure and semantics. As a result, software tools that genuinely support contract editing, by providing on-demand analysis and reasoning of the linguistic content of a contract, remain absent in the field.

The present paper seeks to address this gap. In particular, we describe work on an intelligent contract editor which (a) exploits well-understood NLP techniques to extract information from the text as it is being drafted, using this (b) to enable intelligent browsing of contract clauses, and (c) to automatically construct a partial formal representation that supports automated reasoning about the core elements of the contract.

II. Architecture

In designing the tool we were motivated by the need to have an architecture that is extensible, given the relatively frequent updates that NLP tools/models tend to get, and the different approaches one can take to the same NLP problem. Thus, as shown in Figure 1, we physically separate the architecture in three modules that encapsulate the UI, the contract-support algorithms, and the off-the-shelf NLP tools and databases. In each module we similarly take a component-based approach, keeping each algorithm separate, using dependency injection to enable exchange of these components without the need to change and re-compile the system.

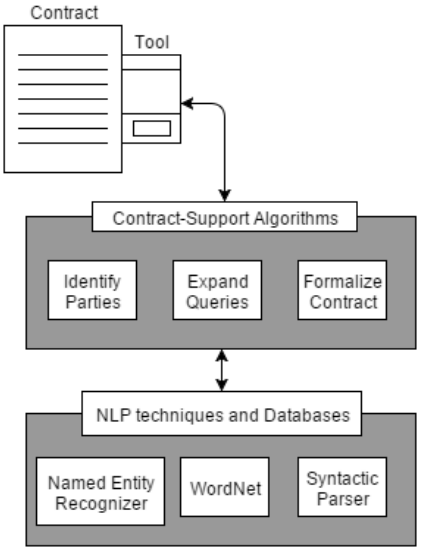

Fig. 1. Three-layered (UI, Contract-Support, and NLP and Databases) tool architecture.

Our tool involves the use of off-the-shelf NLP tools, such as a dependency parser and a part-of-speech tagger, of which there are multiple implementations. Moreover most are not developed in C#, our language of choice. To overcome this problem our architecture exploits loosely coupled modules and C# wrappers for the off-the-shelf tools to enable interfacing with out tool.

III. Information Extraction

Our contract-editing tool is implemented as an add-in to Microsoft Word, allowing analysis of the contract side to side with contract editing. For this we exploit several information extraction algorithms, information which we then associate with each contract clause as a set features.

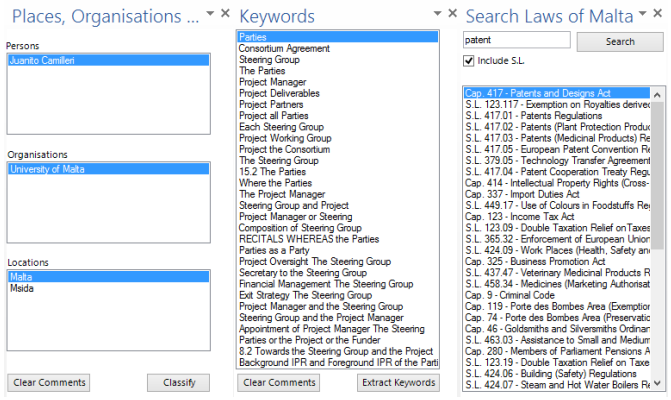

Fig. 2. Task pane views of law search, keyword and named entity extraction.

We developed an algorithm that uses a mixture of regular expressions and named entity extraction to identify the parties to the contract. Due to the structure of contracts being dependent on the drafter, rather than some universal template, this is not always effective, thus we allow the user to specify these themselves to enable further analysis.

We use keyword and named entity extraction in this manner, such that each clause is labelled with the keywords specific to it and the named entities mentioned by it, along with the parties mentioned.

These sets are used to enable the user to browse the contract in question quickly, by highlighting the specific keyword, party, and/or entity in question. This can be useful for, for example, identifying clauses talking about a certain party or a specific concept (e.g. clauses involving a payment, or involving a specific party).

Drafting contracts can also require or benefit from consultation with a country's laws and/or other relevant legal documents. Thus, our tool also includes the capacity to search through the laws and other legal documents of Malta. A database containing information about companies is also included, to allow cross-referencing with official company details which are important to get right in a contract.

To improve on both the results of these searches and the clause browsing we employ query expansion, where we consider also the synonyms, and one edge away hypernyms and hyponyms of the query.

IV. Formal Analysis

Extracting sufficient information from a contract to support such reasoning remains an understudied problem. Indeed, most approaches to legal texts that apply NLP techniques tend to view the task as a form of information retrieval whose results are insufficient to support automated reasoning [3].

Reasoning about contracts can be done by modelling these using deontic logic [4], which views contracts as agreement between two or more parties, with norms (i.e. obligations, permissions, and prohibitions) and structures over these (e.g. sequential composition of these). To bridge the gap between a natural language contract and such a model we have defined a partial deontic logic that we can use to reason about a contract which is only partially known. This is important since this is intended to help during contract-drafting, when the contract is not complete, and also in the case where the translation algorithm is imperfect.

Our approach to translate English contracts into this formal representation uses syntactic parsing, where, for example, "The passenger should check in" (an obligation clause) would be of the following form S $\rightarrow$ NP (VP $\rightarrow$ MD VP'), from which we can extract the party (NP $\rightarrow$ the passenger), the norm (MD $\rightarrow$ should), and the action (VP' $\rightarrow$ check in). Thus, we can get the formal counterpart of the clause, i.e. $O_{passenger}(checkIn)$, indicating an obligation on the passenger to check in.

With this formal representation we can detect conflicts automatically [2], e.g. detecting that a prohibition and an obligation to do the same action cannot be satisfied since they are in conflict.

V. Conclusion

Professionals involved in contract-drafting have the potential to benefit from tools that employ NLP techniques that can automatically analyse the contract while it is being written. This is an area of the humanities where NLP tools have yet to make an impact.

We developed a tool as an add-in to MS Word that presents several features as task panes. These employ keyword and named entity extraction so as to facilitate the extraction of certain key words associated with each clause, to enable easier browsing of a contract depending on these keys.

We also employ a partial deontic logic we constructed, and syntactic parsing to automatically (partially) translate an English contract into a deontic logic model from which automated deductions can be made. Specifically conflicts between clauses can be detected.

The tool was tested by lawyers and notaries, getting overall positive feedback with suggestions for further work (e.g. including contract templates), with the law and company search being seen as the most useful, and automated deduction as promising.

References

[1] P. Gervs, "Story generator algorithms," in *The Living Handbook of Narratology*, P. Hhn, Ed. Hamburg: Hamburg University, 2013.

[2] S. Fenech, G. J. Pace, and G. Schneider, "Automatic conflict detection on contracts," in *Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing*, ser. ICTAC 2009, August 2009.

[3] X. Gao and M. P. Singh, "Extracting normative relationships from business contracts," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS 2014, May 2014.

[4] G. H. Von Wright, "Deontic logic: A personal view," *Ratio Juris*, vol. 12, no. 1, pp. 26–38, March 1999.

[5] S. Azzopardi, "Intelligent contract editing," Master's thesis, Department of Computer Science, University of Malta, 2015.