Integrating Natural Language and Formal Analysis for Legal Documents

Shaun Azzopardi,* Albert Gatt,† Gordon Pace*

*Department of Computer Science
Faculty of ICT
University of Malta
shaun.azzopardi@um.edu.mt
gordon.pace@um.edu.mt

†Institute of Linguistics
University of Malta
alber.gatt@um.edu.mt

Abstract

Although much research has gone into natural language legal document analysis, practical solutions to support legal document drafting and reasoning are still limited in number and functionality. However given the textual basis of law there is much potential for NLP techniques to aid in the context of drafting legal documents, in particular contracts. Furthermore, there is a body of work focusing on the formal semantics of norms and legal notions which has direct applications in analysis of such documents. In this paper we present our attempt to use several off-the-shelf NLP techniques to provide a more intelligent contract editing tool to lawyers. We exploit these techniques to extract information from contract clauses to allow intelligent browsing of the contract. We use this surface analysis to bridge the gap between the English text of a contract to its formal representation, which is then amenable to automated deduction, specifically it allows us to identify conflicts in the contract.

1. 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 (Gervás, 2013). One area of the humanities that has increasingly received more 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. Such tools already exist that provide some form of help to the contract drafter (e.g. (Gabbard et al., 2015)), however what is lacking is an actual analysis of the semantics of the text.

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 (Fenech et al., 2009; Gao and Singh, 2014). 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.

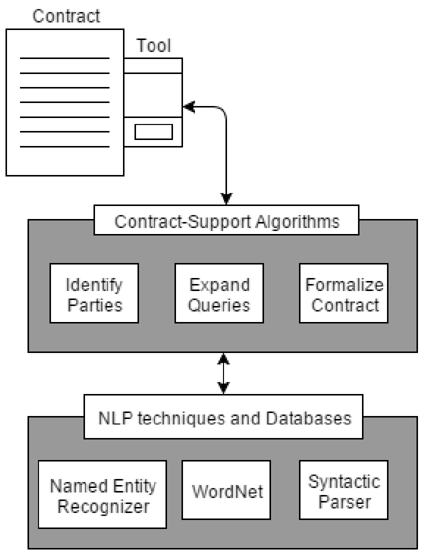

Figure 1: Three-layered (UI, Contract-Support, and NLP and Databases) tool architecture.

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.

2. 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 recompile the system.

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.

3. 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.

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.

4. 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 (Gao and Singh, 2014; Dragoni et al., 2015; Wyner and Peters, 2011).

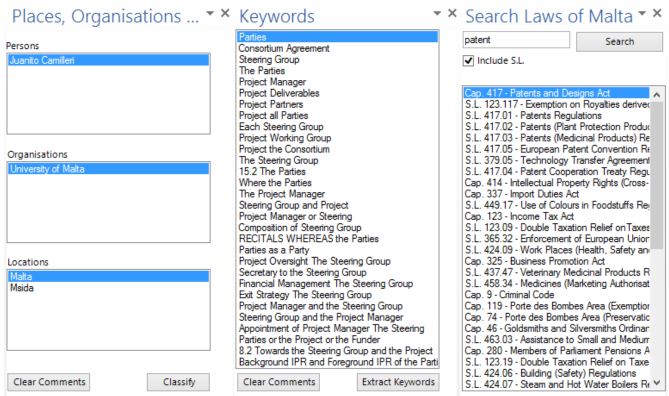

Figure 2: Task pane views of law search, keyword and named entity extraction.

Reasoning about contracts can be done by modelling these using deontic logic (Von Wright, 1999), 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 constructed a 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.

Definition 1 illustrates how a contract is defined in our logic, note how the deontic norms are modelled as predicates over an action α, labelled with the acting party p. Simple contract clauses can then either be an obligation (O), a permission (P), a prohibition (F) or a clause with the norm being unknown ($?_p(\alpha)$). These clauses can then also either be sequentially composed ($C \rhd C'$), reparated[1] ($C \blacktriangleright C'$), concurrently composed ($C \& C'$), or a conditioned on actions occurring ($[e]C$).

Definition 1. *A contract C, where α is an action label and p is a party label, is defined as follows:*

$$C := O_p(\alpha) \mid P_p(\alpha) \mid F_p(\alpha) \mid C \rhd C \mid C \blacktriangleright C \mid C \& C \mid [e]C$$
$$e := \alpha \mid 0 \mid 1 \mid e.e \mid e + e \mid e \& e$$

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')**, shallowly, as in Figure 3. Note how this structures the sentence such that it separates the party **(NP** $\rightarrow$ **the passenger)**, the norm **(MD** $\rightarrow$ **should)**, and the action **(VP'** $\rightarrow$ **check in)** into different sub-trees.

[1] A reparation clause C' for a contract C comes into effect if and after C is violated.

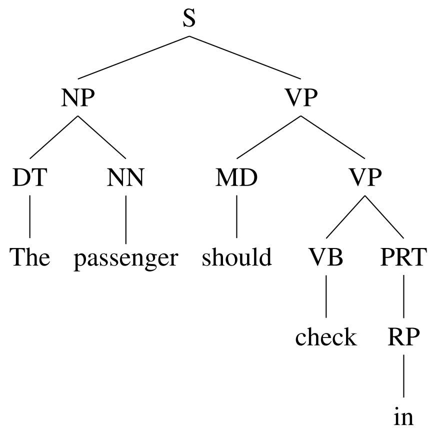

Figure 3: Parse tree of a normative sentence.

To extract these from a sentence we define a number of pattern-matching expressions using Tregex (Levy and Andrew, 2006), which allows us to separately grab exactly the relevant features of a normative sentence. Thus with an appropriate expression we can get the formal counterpart of the clause, i.e. $O_{\textit{passenger}}(\textit{checkIn})$, indicating an obligation on the passenger to check in. We have defined several such expressions that correspond to a certain parse tree structure (along with the presence of a norm specifier like *should*, or *permitted to*). Although it is not clear whether such constructions always correspond to a normative sentence (e.g. "The receptionist should have been here" does not specify an obligation, although it can be seen to imply a perceived one), in the limited context of contracts this is more likely.

This approach is however limited by the number of expressions defined (and their quality). Another issue is that some sentences may not have correlates in our logic (e.g. a distinction is made between state-based and action-based clauses, between which their is no one-to-one correlation (Hage, 2001)).

With this formal representation we can detect conflicts automatically, through an appropriate trace semantics (e.g. $O_p(a)$ is satisfied if a is done, while $C \rhd C'$ is satisfied if C is satisfied, after which C' applies and is satisfied). We generate an automaton that is liable to conflict analysis at states, using the method delineated in (Fenech et al., 2009). While we axiomatise conflicts in Definition 2, from (Gordon J. Pace, 2012).

Definition 2. *Two contracts are said to be in conflict if there is no trace that satisfies both at the same time. The conflict relation is denoted by* $\maltese$*, so that that* C *and* C' *are conflicting is denoted by* $C\maltese C'$*. Note also that we denote two mutually exclusive actions as* $a \bowtie a'$*.*

Contract	TP	TN	FP	FN	Precision	Recall	F_1	$F_{0.5}$
C14	14	33	5	2	0.739	0.875		
C21	9	170	56	0	0.139	1		
C41	16	61	9	0	0.64	1		
C69	12	37	4	0	0.75	1		
C199	5	37	18	10	0.217	0.333		
Results					0.497	0.842	0.625	0.541

Figure 4: Formalizing norms evaluation.

Axioms:

$$\vdash P_p(a)\maltese F_p(a) \tag{1}$$

$$\vdash O_p(a)\maltese F_p(a) \tag{2}$$

$$a \bowtie a' \vdash O_p(a)\maltese O_p(a') \tag{3}$$

$$a \bowtie a' \vdash O_p(a)\maltese P_p(a') \tag{4}$$

$$C\maltese C' \vdash C'\maltese C \tag{5}$$

$$C\maltese C' \wedge\ C' \equiv C'' \vdash C\maltese C'' \tag{6}$$

5. Evaluation

We evaluated our research in two ways: (1) by testing our English to deontic logic translation on a random selection of contracts from the Australian Contract Corpus (Curtotti and McCreath, 2011); and (2) simple feedback from notaries after a few days use.

As our gold-standard we selected five contracts from the corpus (of varying length), and then hand-tagging each clause with a suitable representation in our logic (where possible). Clauses were also tagged as *normative* or not, and as *formalizable* (in our logic) or not.

The results from this are shown in Figure 4, where: True positives those clauses that can be formalized and have been formalized correctly. False positives are clauses that are cannot be formalized but have been formalized (incorrectly). True negatives are clauses that cannot be formalized and where not attempted. While false negatives are clauses that can be formalized and where not formalized.

As can be seen the amounts of false positives is not negligible, especially with contracts **C21** and **C199**. Through an analysis of their text we conclude that these false positives occur mostly in the definitions section of these contracts. These are only tagged as normative (given the presence of possible norm specifiers), with translation mostly failing. Methods however exist to extract definitions automatically (e.g. (Curtotti et al., 2013)) which we should employ in a more mature version of the algorithm.

The tool as a whole was given to a number of lawyers for simple feedback on its utility for them. The feedback was mostly positively focused on the legal document and company search, and neutral from those who said they rarely consult with such already available online databases. The other features were not perceived as useful, although it was pointed out that they may be more applicable in the context of large contracts.

6. Discussion

Our tool thus effectively combines existing NLP tools and formal contract analysis algorithms, providing for a degree of automated analysis. However, there are other features that we did not consider that would make the tool more attractive to notaries, such as a templating system, easing the analysis of definitions (e.g. (Curtotti et al., 2013)), a versioning system (a work-in-progress), or a higher-level analysis of the components of a contract (e.g. (Gabbard et al., 2015)).

On the formal side our approach also has some limitations. A major one is the fact that we check for equality between actions simply by checking for string equality. A better measure of equality can be added to our algorithm by semantic similarity measures that use a lexical database to analyse the senses of a word (as done in (Aires et al., 2015)).

The logic used needs to be augmented with state-based norms as first-class entities, since these appear in contracts also although seemingly at a lesser incidence then action-based ones. An example of such a norm is "The passenger should be in possession of their passport during the whole trip", which we detect automatically by noting the use of "be"[2].

7. 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 deontic logic, 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.

8. References

Jo˜ao Paul Aires, Vera Lucia Strube de Lima, and Felipe Meneguzzi. 2015. Identifying potential conflicts between norms in contracts. In *18th International Workshop on Coordination, Organisations, Institutions and Norms (COIN 2015) @IJCAI*, July.

Shaun Azzopardi. 2015. Intelligent contract editing. Master's thesis, Department of Computer Science, University of Malta.

Michael Curtotti and Eric C. McCreath. 2011. A corpus of australian contract language: Description, profiling and analysis. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, ICAIL '11, pages 199–208, New York, NY, USA. ACM.

Michael Curtotti, Eric McCreath, and Srinivas Sridharan. 2013. Software tools for the visualization of definition networks in legal contracts. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, ICAIL '13, pages 192–196, New York, NY, USA. ACM.

Mauro Dragoni, Guido Governatori, and Serena Villata. 2015. Automated rules generation from natural language legal texts. In *Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts*, pages 1–6, San Diego, USA, June.

Stephen Fenech, Gordon J. Pace, and Gerardo Schneider. 2009. Automatic conflict detection on contracts. In *Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing*, ICTAC 2009, August.

Jason Gabbard, Jana Z. Sukkarieh, and Federico Silva. 2015. Writing and reviewing contracts: Don't you wish to save time, effort, and money? In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 229–230, New York, NY, USA. ACM.

Xibin Gao and Munindar P. Singh. 2014. Extracting normative relationships from business contracts. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS 2014, May.

Pablo Gervás. 2013. Story generator algorithms. In P. Hühn, editor, *The Living Handbook of Narratology*. Hamburg: Hamburg University.

Fernando Schapachnik Gordon J. Pace. 2012. Contracts for interacting two-party systems. In Anders P. Ravn Gordon J. Pace, editor, *Proceedings of Sixth Workshop on Formal Languages and Analysis of Contract-Oriented Software*, volume 94 of *EPTCS*, pages 21–30.

Jaap Hage. 2001. *Contrary to Duty Obligations - A Study in Legal Ontology*. IOS Press.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *5th International Conference on Language Resources and Evaluation*, LREC 2006.

Georg Henrik Von Wright. 1999. Deontic logic: A personal view. *Ratio Juris*, 12(1):26–38, March.

Adam Wyner and Wim Peters. 2011. On rule extraction from regulations. In Katie Atkinson, editor, *JURIX*, volume 235 of *Frontiers in Artificial Intelligence and Applications*, pages 113–122. IOS Press.

[2] There does not exist a single action a such that for this example we can construct a norm $O_p(a)$, i.e. a norm which is satisfied by the performance of a single action.