Suggesting Model Fragments for Sentences in Dutch Law

Emile de Maat*, Radboud Winkels*
*Leibniz Center for Law, University of Amsterdam, {demaat|winkels}@uva.nl

Abstract. A main issue in the field of artificial intelligence and law is the translation of source of law that are written in natural language into formal models of law. This article describes a step in that transformation: the creation of models for individual sentences in a source of law. The approach uses a natural language parse to analyse the sentence, and then translates the resulting parse tree to a formal model, using both generic and law-specific attributes.

Keywords: Automated Modelling, Natural Language Processing

1. Introduction

A main issue in the field of artificial intelligence and law is the transformation of sources of law that are written in natural language (and therefore rather informal) into formal models of law that computers can reason with. This is a time and effort consuming process, error prone and different knowledge engineers will arrive at different models for the same sources of law. Moreover, these models should be closely linked to the original sources (and at the right level of detail, i.e. isomorphic) since these sources tend to change over time and maintenance of the models is a serious problem. This calls for tools and a method for supporting this modelling process and increasing inter-coder reliability.
We have been researching a method to create isomorphic models semi-automatically, focusing on (Dutch) laws. This article presents a next step in this creation process.

1.1. General Approach

In order to achieve (semi-)automatic modelling of legal sources, we follow a number of steps, as shown in figure 1. The process starts with the source document, written in natural language (Dutch). Currently, we focus on laws, though we hope to expand to other types of legal sources later on. We first make the structure of the document explicit, by marking up the different parts, such as chapters, paragraphs and sentences, and assigning identifiers to each part. We then proceed to mark all references to other legal sources that are contained in the text, using a parser based on patterns for references (see (de Maat, 2006)). This structure and reference information is stored in CEN/MetaLex XML[1].

[1] See http://www.metalex.eu/

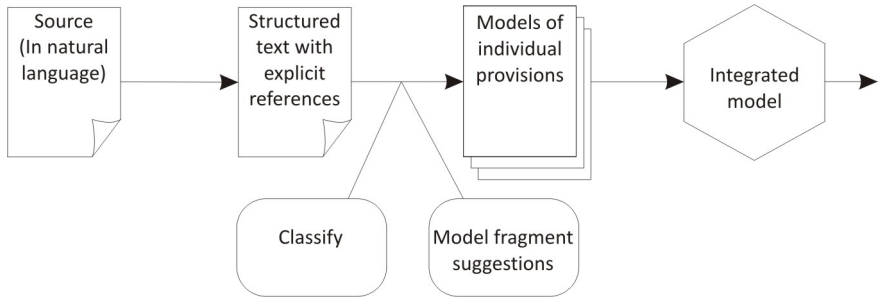

Figure 1. Steps in automatic modelling of legal texts.

The next step is to create models for each individual statement in the text. In most cases, each sentence in Dutch law forms a complete statement (though possibly part of a bigger construct), so we are, in fact, creating a model for each sentence in the text. In the last step, these individual models are integrated with each other to come to a complete model. In order to create the models, we start by classifying each sentence in the text as a specific provision, such as a definition, a duty, or a modification of an earlier law. In total, we recognise ten different main categories. As with the references, this is done by recognising certain patterns in the text (de Maat, 2008).
For several types of statements, such as modifications and setting the enactment date or citation title, recognising the pattern and classifying the sentence is also nearly sufficient for creating a model of the sentence. For example:

Aliens Act 2000
This law is referred to as: Aliens Act 2000.

This sentence is classified by the pattern "is referred to as", which splits the sentence in two parts: a reference (recognised by the reference parse) to "this law" and a citation title. This is all the information that is needed to represent the meaning of this sentence[2]. More elaborate sentences, that contain terms relating to the subject matter that the law is about, require more detailed analysis[3]. A natural language parser can provide such a more detailed analysis. This paper describes our initial experiences while using a natural language parser to enhance the input for our modeller. For this research, we have used the Alpino parser for Dutch (Bouma, 2001) to parse the sentences. The Alpino parser assigns a dependency structure to the sentence. These structures are described by Bouma et al:

[2] As said, this also holds true for sentences containing modifications to other legal sources. However, for such sentences, analysis of the modified text is needed to determine the full impact (not meaning) of such a sentence.

[3] This applies to norms, definitions and many application provisions. Earlier research (de Maat, 2008) suggests that these comprise about 64% of the sentences encountered.

Dependency structures make explicit the dependency relations between constituents in a sentence. Each non-terminal node in a dependency structure consists of a head-daughter and a list of non-head daughters, whose dependency relation to the head is marked.

The dependency structure can be stored as an XML file, which is the format we use as input for our modeller.

2. Creating Model Fragments

Our approach is similar to that published in (Biagioli, 2005), where Italian laws are modelled. However, Biagioli et al. aim for fairly rough frames; for example, for an obligation, their approach attempts to fill the slots addressee, action and third-party. We hope to achieve some more detail, splitting up these fields in more parts. In this sense, our method comes closer to those of (Sarwar Bajwa, 2009), who generates UML models from parse trees, (McCarty, 2007), who transforms parse trees to quasi-logical form, or (Bos, 2004), who translate parse trees to first order logic statements. Both these methods map individual words to model elements. An example by Bos et al:

The school-board hearing at which she was dismissed was crowded with students and teachers.

This results in the following first-order logic statement:

$$\exists a((school-board(a) \wedge hearing(a)) \wedge \exists b(female(b) \wedge \exists c(dismiss(c) \wedge (patient(c,b) \wedge (at(a,c) \wedge \exists d(crowd(d) \wedge (patient(d,a) \wedge (\exists e(student(e) \wedge with(d,e)) \wedge \exists f(teacher(f) \wedge with(d,f))) \wedge event(d)))))))))$$

We wish to mix these approaches. For normative sentences, this means that we see each normative sentence as describing a situation that is allowed or disallowed. We consider the main verb of a sentence as the action that is allowed or disallowed, with the other elements being modifiers or properties of that action. For example:

Our Minister issues a warrant to the negligent person.

The main verb of this sentence is *to issue*, so that is considered the action. Properties of this action are the subject (*Our Minister*), the direct object (*a warrant*) and the indirect object (*the negligent person*). All these elements are distinguished by the Alpino parser, allowing us to extract them for our model. Within Dutch law, this sentence format expresses an obligation, so the action as a whole is classified as an obligation.

Obligation	
Action	issue
Subject	Our Minister
Direct Object	warrant
Indirect Object	negligent person

The articles (*the*, *a*) are left out of the model, though they are stored internally, as they are of importance during later integration of the model; *the negligent person* often is a reference to an earlier sentence, whereas *a negligent person* is not.

Further detail can be added by splitting of adjectives and relative clauses from the noun they modify. For example, *negligent person* has two properties: being a person and being negligent. Splitting adjectives from nouns is not always desirable; it is preferable to leave multiword expressions intact. *European Union* is not any union that is also European; *Our Minister of Finance* is not any minister that is also ours, and of finance[4]. Instead, these are references to concepts that have been defined elsewhere: the common sense domain, the juridical domain or elsewhere in this law. Common multiword expressions are recognised by the Alpino parser; juridical domain or law-dependent expressions need be filtered out separately.

Relative clauses are more complex then adjectives, as they contain a complete new sentence. In this case, we repeat the procedure for the main sentence, identifying the main action and all properties of that action. For example:

Our Minister issues a warrant to the person that neglected his duties.

This sentence would yield a frame like[5]:

Obligation		
Action	issue	
Subject	Our Minister	
Direct Object	warrant	
Indirect Object	person	
	subjectOf	
	Action	neglect
	Direct Object	his duties

[4] In Dutch laws, *Our Minister of Finance* is a reference to the (Dutch) Minister of Finance. No more detailed model is needed, as no derivations need to be made.

[5] For the moment we use a frame-like representation. These look somewhat like the frames presented by (van Kralingen, 1995), but these were more legally oriented and had a fixed number of slots, while our structures are more dynamic and language oriented

2.1. Filtering out signal words

The sentences we showed above are examples of normative sentences that do not use signal words; only the desired situation is described, and it is left implicit that this is an obligation. Other sentences in the law use signal words to make the kind of norm explicit, such as:

The buyer is obliged to pay the price.[6]

This sentence uses *is obliged* to make it clear that this is an obligation. Other examples of signal words are *must*, *may* and *is allowed*. These sentences require a different approach than the sentences without signal words. If we were to use the same approach, the result would be something like:

Obligation	
Action	is obliged to pay
Subject	buyer
Direct Object	price

This is not a desirable outcome, as the action that this norm deals with is pay rather than is obliged to pay. When modelling these sentences, these signal words should not be included in the model of the situation (their meaning is translated into whether the situation is allowed or disallowed). Ideally, after we have categorised the sentence (based on the signal words), we would like to transform the sentence to a sentence without signal words, like:

The buyer pays the price.

We could then model that sentence to come to a correct frame. Simply leaving out the signal words may lead to errors, since the role of the other words might need to shift as well. However, the parse of the sentence actually contains this "transformed sentence" that we want to model. This is shown in figure 2.

Beneath the body node, we find exactly the sentence that we are looking for. Alpino assigns this dependency structure to any sentence that follows this pattern. This makes it easy to filter out the signal words by simply focusing on the part of the parse tree that contains the transformed sentence. For each pattern we use for classification, it seems possible to define a part of the parse tree that should be ignored in order to come up with a correct model.

2.2. Passive voice

Many sentences in Dutch law are phrased in the passive voice, such as this instruction:

[6] Dutch Civil Code, BW7, article 6 sub 1.

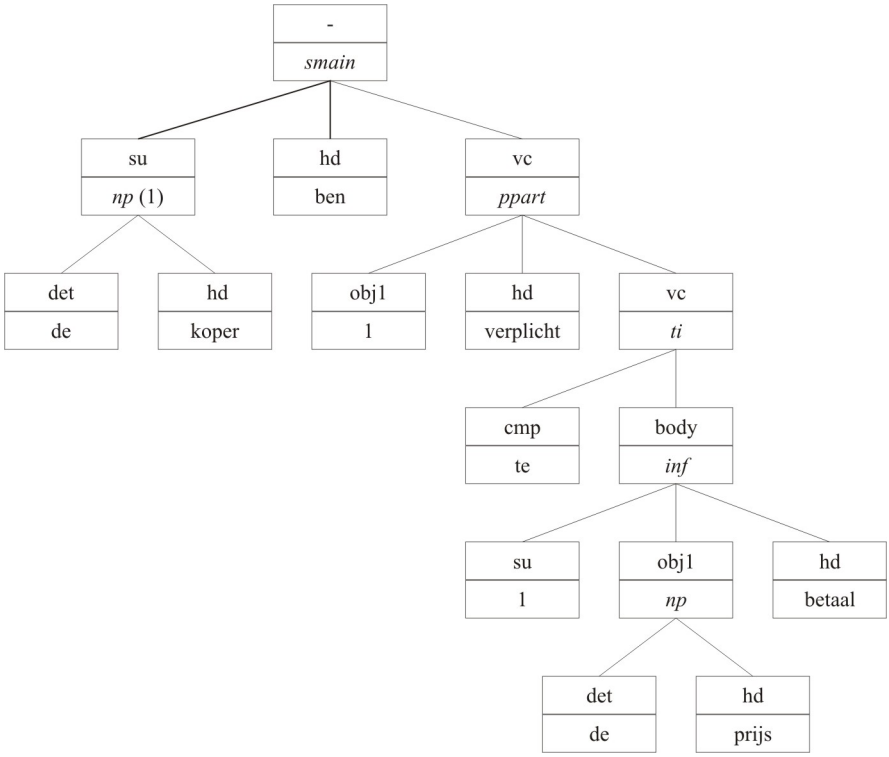

Figure 2. Alpino parse tree (with reduced information) for "The buyer is obliged to pay the price" (in Dutch).

An English translation is added to this report.[7]

A sentence in the passive voice cannot be modelled in the same way as a regular sentence, as the subject of the sentence is actually the direct object, and should be modelled as such. Again, the parse of the sentence gives us an easy way to do this:

The verb clause (vc) of the sentence holds the sentence in active voice, with the subject re-cast in the role of object. By modelling the verb clause instead of the sentence as a whole, we get the correct model, with the correct object, and without the auxiliary verb. If the actual subject is present in the sentence (for example, if the sentence would read*An English translation is added to this report by the organiser*), then this prepositional object is not re-cast in the role of object in the tree. We will have to detect its presence by scanning for signal words like *by*. As this does not always indicate a subject, this will be one of the cases were human validation is necessary.

[7] Law for the protection of Antarctica, article 33, sub 3

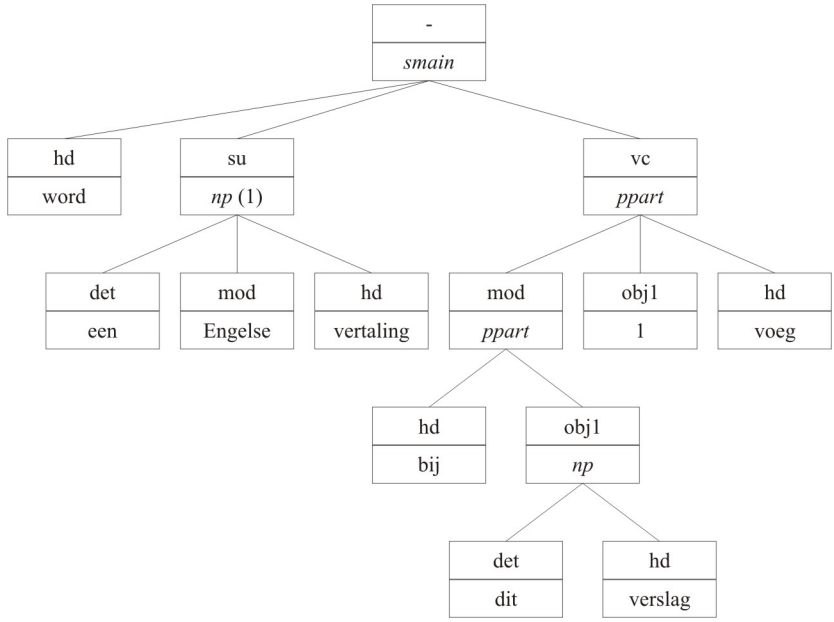

Figure 3. Alpino parse tree (with reduced information) for 'An English translation is added to this report" (in Dutch).

2.3. LISTS

Lists are also recognised by the Alpino parser, and can therefore easily be added to our models as the union or intersection of the different list items, depending on the conjunction used. However, though the conjunction *and* suggests an intersection, it often expresses a union instead. For example:

Advances and duties are paid in cash.

In this sentence, it is the union of *advances*and *duties*that is meant. Our current approach is to translate *and*with a union if it appears in a relative clause, and with an intersection otherwise.

2.4. NEGATION

Negative sentences should also be recognised, and modelled as the "positive" sentence, with the additional notion that it is inverted. This can usually be done by not including certain signal words as element in the model, but by inverting the model if it is encountered. The most common signal word is *not*. If it is encountered, it is not added to the frame, but instead, the containing element is marked as inverted. The determiner*no* is another example of a signal word for negation. However, it can affect more than its containing element. For example:

No bodies are interred on a closed cemetery.

This is an obligation, and the direct object of this sentence is *no bodies*. However, if we apply the negation simply to the object, i.e. the object is "not a body", it would imply the obligation of to bury something that is not a body on the cemetery. Instead, we need to apply the negation to the entire sentence: On is obliged not to bury bodies at a closed cemetery.

3. Experiences

At this moment, we do not have a fully automated process to create the models, and have not yet tested this method on a large body of sentences. Instead, random sentences have been selected, parsed using Alpino and then fed into our modeller.

There is a clear difference between the computer generated models and those created by a human expert with regard to the granularity of the model. Our method will create models with model elements that represent one word from the original sentence, whereas a human expert is more likely to include some sentence fragments as a whole. For example, one Dutch law defines an alcoholic drink as *the drink that, at a temperature of twenty degrees Celsius, consists of alcohol for fifteen or more volume percents, with the exception of wine*. Our algorithm will dissect this sentence, whereas most human modellers will leave the first subordinate sentence intact and add it to the model as a single attribute (most likely abbreviated to *alcohol by volume*). A more detailed model seems not necessarily wrong, but quite possibly over-the-top and inconvenient for many applications.

The method assigns rather broad categorisations to each object (it is either a direct, indirect or prepositional object), but does not yet assign a legal meaning to such an object. It may be a third party involved or the instrument. Perhaps this is not an obstacle; users dealing with a system based on such models are likely to recognise the roles from the context and language used, whereas a computer does not need this information for the derivations we currently want to make. For future projects, though, the information may be required, and some way to automatically recognise it is desired.

For the modelling of norms, we have been focussing on the sentences that represent an obligation, duty or right. For those sentences, the method seems adequate. However, for other types of sentences, such as delegation, we have not come to an acceptable approach yet. Dealing with these sentences will require first of all that we recognise them. Currently, our classifier distinguishes only between obligation/prohibition and right/permission. Several of the patterns used clearly indicate delegations, but we have not yet established whether these patterns cover all delegations in Dutch laws.

A minor problem with regard to the parses made by Alpino is that most often,

the correct parse is not the one preferred by Alpino, but second, third or fourth. If we make several suggestions (each suggestion based on a parse by Alpino), this means that it will often not be the first suggestion that is correct, which means more effort is needed by a human expert who is verifying the models.
We expect that by expanding the lexicon used by Alpino, and perhaps by recalibrating the disambiguation on a written legal corpus, these problems will disappear.

4. Conclusion

We have presented a next step towards a method and tools for supporting the semi-automatic modelling of sources of law, necessary for an efficient, effective, and more reliable and pragmatic use of knowledge technology in the legal domain. We were already able to reliably detect structure in sources of law, find and resolve references in and between them, and classify individual sentences. Now we are able to suggest formal model fragments for certain types of the classifications. Though we are convinced that these model fragments will be a useful in supporting human experts creating models, we do feel that the approach is still too general. A more elaborate method is needed to create appropriate model fragments for different subtypes of sentences. Some method to avoid too granular models is desirable as well.

References

Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S. and Soria, C. (2005), *Automatic semantics extraction in law documents*, in "Proceedings of the Tenth International Conference on Artificial Intelligence and Law (ICAIL '05)", ACM, New York, pp. 133-140.

Bos, J., Clark, C., Steedman, M., Curran, J.R. and Hockenmaier, J. (2004), *Wide-Coverage Semantic Representations from a CCG Parser*, in 'Proceedings of the 20th international conference on Computational Linguistics", pp. 1240-1246.

Bouma, G., Noord, G. van, and Malouf, R. (2001), *Alpino: Wide Coverage Computational Analysis of Dutch*, in Daelemans, W., Sima'an, K., Veenstra, J. and Zavrel, J. (Eds.), "Computational Linguistics in the Netherlands CLIN 2000. Selected Papers from the Eleventh CLIN Meeting.", Rodopi, Amsterdam, pp. 45-59.

Kralingen, R. W. van (1995), *Frame-based Conceptual Models of Statute Law*, PhD thesis, Kluwer Law International, The Hague.

Maat, E. de, Winkels, R. and Engers, T. van (2006), *Automated Detection of Reference Structures in Law*, in Engers, T.M. van (Ed.), "Legal Knowledge and Information Systems. Jurix 2006: The Nineteenth Annual Conference", IOS Press, Amsterdam, pp. 41-50.

Maat, E. de and Winkels, R. (2008), *Automatic Classification of Sentences in Dutch Laws*, in Francesconi, E., Sartor, G. and Tiscornia, D. (Eds.), "Legal Knowledge and Information Systems. Jurix 2008: The Twenty-First Annual Conference", IOS Press, Amsterdam, pp. 207-216.

McCarty, L.T. (2007), *Deep semantic interpretations of legal texts*, in "Proceedings of the 11th International Conference on Artificial Intelligence and Law", ACM Press, New York, pp. 217-224.

Sarwar Bajwa, I., Samad, A. and Mumtaz, S. (2009), *Object Oriented Software Modeling Using NLP Based Knowledge Extraction*, European Journal of Scientific Research, Vol. 35, No. 1, pp. 22-33.