Suggesting Model Fragments for Sentences in Dutch Law

EMILE DE MAAT, RADBoud WINKELS


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 sources of law, we follow a number of steps, as shown in Fig. 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 sources of law 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 within the source to
other sources of law, using a parser based on patterns for references\(^1\). This structure and reference information is stored in CEN/MetaLex XML\(^2\).

\[\text{Fig. 1 – Steps in automatic modelling of legal texts}\]

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 automatic recognition of certain patterns in the text\(^3\).

For several types of sentences, these patterns, together with some added features, are sufficient to extract all information needed to create a model of the sentence. This is usually the case with sentences that are about the law itself, instead of the subject matter of the law. These sentences are discussed in Section 2. Other sentences, such as obligations, do focus on the subject matter, and can vary wildly. Simple patterns will not suffice to deal with these sentences, and to extract information from these types of sentences, we


\(^2\) See http://www.metalex.eu/.

use a Dutch grammar parser. These sentences are discussed in Section 3. In both cases, we model the sentences using a frame-like representation. These look somewhat like the frames presented by van Kralingen⁴, but these were more legally oriented and had a fixed number of slots, while our structures are more dynamic and language oriented.

2. Sentences Dealing with the Law

There are several types of sentences that are about the law itself. These are sentences that change the text of a law or set the enactment date of a law. Compared to sentences which deal with the subject of the law (which could be anything: oil tankers, accountancy, book prices, etc.) they have very limited variation. In general, they follow clear patterns, from which it is easy to extract the information we need.

2.1. Scope Declaration

A scope declaration always contains one relevant piece of information: the scope. This scope is identified by a reference to the relevant text. If we assume that the reference parser has successfully identified all references during pre-processing, then the scope is easily found: it is the reference at the start of the sentence, or more easily, the one reference that appears in the sentence.

Law of May 20th, 2010 (Stb. 2010/200), article I, introduction

The Penal Code is modified as follows:

<table>
<thead>
<tr>
<th>Scope</th>
<th>Penal Code</th>
</tr>
</thead>
</table>

If a scope is set for an article or subpart, then all changes made within that article or subpart refer to that scope.

2.2. Repeal

A repeal removes an entire law, or part of that law. Other than the knowledge that we are dealing with a repeal, only one bit of information needs to

be extracted: the reference to the text that is to be repealed. This reference is directly at the beginning of the main sentence (though it is also the only reference).

*Law of June 12th, 2008 (Stb. 2008/229), article I, sub B*

Article 4 is repealed.

| Repeal | Target | Article 4 |

Alternatively, only part of the text is repealed, i.e. a number of words are deleted. In that case, we need to extract the location where the text to be modified is located, and the text that should be removed. The location will be a reference at the start of the sentence, following the word In (though again, it will be the only reference in the sentence). The text to be removed will be marked with double angle quotes.

*Law of May 20th, 2010 (Stb. 2010/205), article III*

In article 438, sub 1, 2nd item, of the Penal Code, «profession or appointment,» is repealed.

| Repeal | Target | Penal Code, article 438, sub 1, 2nd item |
|        | Text   | profession or appointment, |

2.3. Insertion

An insertion inserts some new text into the document. There are several different scenarios for insertion. The easiest situation exists when an entire new (numbered) element is inserted. In this case, sentence will be something like this:

*Law of May 13th, 2004 (Stb. 2004/220), article I, sub A*

After article 7:1 a new article is inserted in Section 7.1, reading: ...

We need to model two pieces of information. Firstly, the text to be inserted, which can be found after the colon. Then, we need to know where to insert it. The (part of the) document is denoted by a reference that is preceded by in (in this case: in Section 7.1). If the insertion has been preceded by a scope declaration, then this reference may be incomplete, and needs to
be combined with the scope declaration to get a complete reference. The location is given by another reference, preceded by either *before* or *after*. If the target is entirely defined by the scope definition, it will be missing from the sentence. Similarly, if the location is the end of the document, no reference for the position will be given. Instead, the text is said to be appended instead of inserted. For example, the following sentence is missing both references:

*Law of May 20th, 2010 (Stb. 2010/200), article I, sub A, sub 2*

A paragraph is appended, reading: ...

<table>
<thead>
<tr>
<th>Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

If a modification is made within a paragraph, it is not possible to point to the position using a reference (as sentences and words are not numbered). In these cases, instead of a reference, quoted text is used to describe the location:

*Law of May 20th, 2010 (Stb. 2010/200), article VII*

In article 30, sub 6, of the Motorised Vehicles Liability Insurance Act, a sentence is inserted after the sentence «...», reading: ...

The quoted text is marked using double angle quotation marks.

<table>
<thead>
<tr>
<th>Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
</tr>
<tr>
<td>Motorised Vehicles Liability Insurance Act, article 30, sub 6</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Position</td>
</tr>
<tr>
<td>After</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

2.4. Replacement

There are two main patterns of replacement, which yield a slightly different frame. The first pattern deals with the replacement of text within a given structure part. The text to be replaced is always marked with double angle quotation marks «». The replacing text is always at the end, preceded by a colon. The location of the replacement is at the beginning of the sentence, a reference following the word *In*. As with the other changes, this should be the only reference in the main sentence (disregarding any references that appear inside the quoted text).
Law of May 20th, 2010 (Stb. 2010/200), article IV, sub I
In article 43a, «a conviction» is replaced by: an earlier conviction.

<table>
<thead>
<tr>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old Text</td>
</tr>
<tr>
<td>New Text</td>
</tr>
</tbody>
</table>

The second type of replacement replaces an entire structure element, and does therefore not include a text to replace. The replacing text is again found after the colon, and the location is given as a reference after In at the beginning of the sentence. The location may be missing, which is usually the case if it has been set earlier with a scope declaration.

Law of May 20th, 2010 (Stb. 2010/205), article IV, sub I
Article 31, sub 2, will read:
2. ...

<table>
<thead>
<tr>
<th>Replacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
</tr>
<tr>
<td>New Text</td>
</tr>
</tbody>
</table>

2.5. Renumbering

Renumbering changes the index of parts of a text. Though it is very uncommon for articles and document parts above the level of articles to be renumbered, paragraphs and list items are frequently renumbered. The sentence will list the part(s) to renumber and the new numbering. A reference to the part(s) to renumber can be found at the start of the sentence, while the new index can be found after are renumbered to.

Modern Migration Policy Act
The articles 2a to 2u are renumbered to the articles 2i to 2cc.

<table>
<thead>
<tr>
<th>Renumbering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
</tr>
<tr>
<td>New numbering</td>
</tr>
</tbody>
</table>

2.6. Enactment Date

The basic sentence that sets an enactment date will include two pieces of information: the document (or parts of document) to be enacted, and the date on which it is to be enacted.
Fuel Taxes Environment Tariff Act 1991, article IV, sub 1, first sentence
This law is enacted starting on January 1st, 1991.

<table>
<thead>
<tr>
<th>Enactment Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
</tr>
<tr>
<td>Date</td>
</tr>
</tbody>
</table>

It is rather uncommon for a law to specify the exact date on which it will be enacted. More commonly, it will defer the decision to a royal decree:

Exception Situations Coordinating Act, article 11
This law is enacted on a date to be set by Royal Decree.

Or it will relate it to the publication date:

Law of July 7th, 2010 (Stb. 2010/305), article 9
This law is enacted starting on the day after the date of publication of the Bulletin of Acts and Decrees in which she is included.

Both constructions follow fixed text patterns, which can be detected instead of a date. In addition to these straightforward enactment date setting sentences, more complicated constructions exist which include exceptions for which the date is set separately.

2.7. Citation Title

A sentence setting a citation title contains two pieces of information: the law for which the citation title is set, and the actual citation title.

Nobility Act
This law may be cited as Nobility Act.

The law concerned is indicated with a reference at the start of the sentence. Usually, this is this law, but sometimes, the citation title of some other law is changed. The new citation title follows the text may be cited as, and is sometimes enclosed in double quotes.

<table>
<thead>
<tr>
<th>Citation Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
</tr>
<tr>
<td>Citation Title</td>
</tr>
</tbody>
</table>
2.8. Application Provision

An application provision refers to a text that does (or does not) apply in a given context (usually the article in which the application provision). For the most basic application provisions, there is only one piece of information to be extracted: a reference to the text that does or does not apply. This reference can be found at the beginning of the sentence:

*Personal Data Act*

Article 11, sub 3, applies accordingly.

| Application | Target | Article 11, sub 3 |

Many application provisions do come with additional restrictions, which often relate to the subject matter of the law.

*Partnership Taxes Act 1969*

This article does not apply to brands and logos produced by the taxable person and any similar assets.

If this is the case, then the core of the sentence is still modelled in the same manner, but the additional restrictions need to be processed in the manner presented in Section 3.

3. Sentences Dealing with Subject Matter

Sentences dealing with the subject matter of the law come in many shapes and varieties, and though there are some clear signal words, it is difficult to predict what features a sentence contains, and to extract those features. Biagioli et al.\(^5\) have made a system in which they create frames for provisions in Italian laws. These frames do not cover all elements of each sentence. For example, for an obligation, their approach attempts to fill the slots addressee, action and third-party. We wish to achieve more detail, by splitting these slots into more parts (if applicable). We also want to ensure that all elements in the sentence are included in the model, even if they do not correspond

---

to a slot. To achieve this, we use a more generic approach. In this sense, our method comes closer to those of Sarwar Bajwa et al.⁶, who generate UML models from parse trees, McCarty⁷, who transforms parse trees to quasi-logical form, or Bos et al.⁸, 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:

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

Our method is a mix between these methods which include all elements from the sentence and Biagioli's frames, which assign a label to these elements. For the sentences in Section 2, we used patterns to identify the different elements; for the sentences here, which vary more widely, we use the Dutch grammar parser Alpino⁹ to divide the sentence into elements.

### 3.1. Normative Sentences

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. A number of these other elements are labelled according


to their semantic role (or thematic relation) in the sentence. The other elements are considered as generic modifiers. At the moment, we distinguish only the agent, patient and recipient of the action. Other researchers have already been working at classifiers to assign semantic roles, and we hope to adopt one of those in the future, but for the moment, we use two simple schemes for labelling them, one for active sentences, and one for passive sentences.

In an active sentence, we assume that:
- the subject is the agent of the action;
- the direct object is the patient of the action;
- the indirect object is the recipient of the 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 agent (*Our Minister*), the patient (*a warrant*) and the recipient (*the negligent person*). All these elements are distinguished by the Alpino parser (as subject, direct object and indirect object), 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.

<table>
<thead>
<tr>
<th>Obligation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
</tr>
<tr>
<td>Agent</td>
</tr>
<tr>
<td>Patient</td>
</tr>
<tr>
<td>Recipient</td>
</tr>
</tbody>
</table>

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.

The example above is an active sentence, but many sentences in Dutch law are phrased in the passive voice, such as this instruction:


An English translation is added to this report.

A sentence in the passive voice cannot be modelled in the same way as a regular sentence, as the subject of the sentence is not the agent, but the patient, and should be modelled as such. The parse of the sentence gives us an easy way to do this (see Fig. 2).

**Fig. 2 – Alpino parse tree (with reduced information)**
_for “An English translation is added to this report” (in Dutch)_

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 agent 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 the agent, this will be one of the cases were human validation is necessary. 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\(^\text{12}\). 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 than 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 yields a frame like:

<table>
<thead>
<tr>
<th>Obligation</th>
<th>action: issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Our Minister</td>
</tr>
<tr>
<td>Patient</td>
<td>warrant</td>
</tr>
<tr>
<td>Recipient</td>
<td>person</td>
</tr>
<tr>
<td>subjectOf</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>neglect</td>
</tr>
<tr>
<td>Direct Object</td>
<td>his duties</td>
</tr>
</tbody>
</table>

3.1.1. 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\(^\text{13}\).

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 sen-

\(^{12}\) 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.

\(^{13}\) Dutch Civil Code, BW7, article 6 sub 1.
sence require a different approach than the sentences without signal words. If we were to use the same approach, the result would be something like:

<table>
<thead>
<tr>
<th>Obligation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
</tr>
<tr>
<td>Agent</td>
</tr>
<tr>
<td>Patient</td>
</tr>
</tbody>
</table>

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 Fig. 3.

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.

3.1.2. 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.
3.1.3. 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: One is obliged not to bury bodies at a closed cemetery.

3.1.4. Explicit Exceptions

Sometimes, a normative sentence in a Dutch law includes a prefix to denote that it is an exception to some other rule, like:

In exception to article 12, ...

Alternatively, some sentence start with a prefix to denote that it is not an exception, like:

Without prejudice to article 12, ...

These prefixes differ from other elements in the sentence in the sense that they do not describe the situation that is allowed or obliged, but instead tell us something about how this rule interacts with some other rule. Hence, this element should not be added to the frame describing the rule.

3.2. Definitions and Deeming Provisions

Definitions and deeming provisions attach a meaning to a specific concept. At top level, a definition contains three elements: the definiendum and the definiens, and, optionally, a scope declaration stating for which sources of law this definition applies. For example:

Medication Act, article 1, introduction and item c

In this law and stipulations based upon it, it is understood by immunological drug: a vaccine, toxin, serum or allergen.

This definition has the scope this law and stipulations based upon it. The definiendum is immunological drug and the definiens is a vaccine, toxin, serum or allergen.

Like the sentences presented in Section 2, these top elements can easily be extracted by means of the pattern used to classify the sentence (in this case it is understood by) and some additional features. The scope, if present, will
follow the word in (and end at the text it is understood by). The definiens will follow the word by and end at the colon, and the defiendum will follow the colon. Thus, we can easily extract a top level frame:

<table>
<thead>
<tr>
<th>Definition</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>this law and the stipulations based upon it</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defiendum</th>
</tr>
</thead>
<tbody>
<tr>
<td>immunological drug</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definiens</th>
</tr>
</thead>
<tbody>
<tr>
<td>a vaccine, toxin, serum or allergen</td>
</tr>
</tbody>
</table>

However, modeling the definiens in this way is unsatisfactory, as it gives insufficient detail to use this model for practical purposes. To create a more useful model, we need to split up the definiens. To do so, we use the same methods used for concepts in normative sentences. This requires a parse tree; we can either parse the entire sentence or just the definiens (see Fig. 4).

![Fig. 4 – Alpino parse tree (with reduced information) for “a vaccine, toxin, serum or allergen” (in Dutch)](image)

4. 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.

5. 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, ef-
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. However, we have not yet arrived at the kind of models that we wish to achieve.

Firstly, 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.

Secondly, it is important to note that these sentence models are in fact partial models. They need to be combined with each other to come to complete models; an integration step is needed. For some situations, this integration is quite simple. If a term is used in an obligation, and that term has been defined in an earlier sentence, it is easy to combine the two. Similar, if there are explicit references, the two sentences are also easily linked. In other situations, the terms will not have a one-on-one match, such as the use of plural instead of singular, the use of group names or the use of verbs as a noun. Also, sometimes terms are not explicitly named in each sentence. For example, this is the case with ellipsis and relative pronouns.

Still, we are convinced that these even these partial models will be a useful in supporting human experts creating models, and feel that this approach will lead to useful tools.