Making sense of legal texts

de Maat, E.

2 Parsing Structure

Text is comprised of words, which combine into sentences. Sentences combine into paragraphs, and paragraphs become chapters. Aided by punctuation and layout, most humans do not have any problem distinguishing this structure. For automated processing, though, the implicit rules that people use to recognise this need to made explicit. This is a fairly straightforward task, though laws have a somewhat more complex structure than most texts.

Most laws follow a common structure. Legislators have tried to create uniform documents in order to increase legibility of the laws. This has led to explicit rules or guidelines, in for example, Africa[8], the European Union[9] and the Netherlands[10]. These documents discuss a wide variety of information related to so-called legislative drafting techniques. They include topics such as structure, numbering, preferred word use and grammar and template sentences.

In various projects, in has been attempted to make the structure of laws explicit, by modelling it in XML. In the Netherlands, the most notable attempts are BWB DTD, the format used on the official website of the government, and MetaLex (Boer et al., 2002), which later merged with several initiatives from other jurisdictions, such as the Italian Norme-in-Rete (NIR) XML schema and the pan-African Akoma Ntoso schema, to form the European standard CEN/MetaLex (Boer et al., 2009). In these formats, structure elements are marked with a tag that denotes its level (e.g. *chapter*) or a more generic tag, such as *part* or *hierarchical container*.

These projects give us useful insights into the structure of laws, and also point us to the fact that there is a lot of variation. Many of such XML schemas were initially too strict, and had to be made more flexible in order to accommodate the many variations that occur. What they do not give us, is some insight in what the language is that is used to denote the structure. Here, the official guidelines help, but since they focus on what *should* be, they give limited information on what actually *is*.

In this chapter, we will discuss the structure of Dutch laws, based upon the information of the guidelines and the study of several laws. After that, we will discuss how we can use this information to perform the first step of the proposed modelling process, as depicted in figure 3. This first step should take as its input a legislative text without any metadata or annotations, and should result in an annotated text in which the structure has been explicitly marked. We will discuss how we can detect the structure based on the information found, and we will present a parser that accomplishes this task.

[8] Legislative Drafting Guidelines for persons involved in the drafting of legislation (2011)
[9] Joint Practical Guide of the European Parliament, the Council and the Commission for persons involved in the drafting of legislation within the Community institutions (2009)
[10] Aanwijzingen voor de regelgeving (2005)

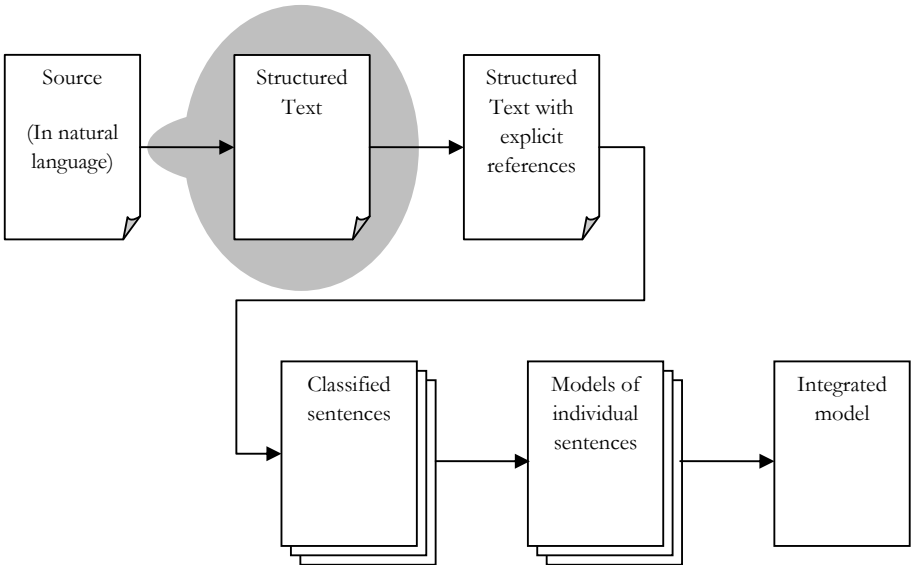

Figure 3: Parsing structure step

The interpretation of what a structure element is, and what should be recognised, follows closely the ideas behind MetaLex, expanded with insights from the official guidelines.

2.1 Structure of Dutch Laws

At top level, Dutch law (or any Dutch regulation) is divided into six parts:

1. title (*opschrift)*;
2. preamble (*aanhef)*;
3. body (*lichaam)*;
4. conclusion (*slotformulier)*;
5. signatures (*ondertekening)*;
6. appendices.

The title of a law is usually rather elaborate, listing the date the law was signed, as well as its purpose, such as:

Act of July 5th, 2006 on the merging of the municipalities of Medemblik, Noorder-Koggenland and Wognum, and on a modification of the borders of the municipality of Hoorn[11]

If the law has a short title, this short title is included in the title between parentheses:

Act of October 31st, 2002, on rules concerning the exploration of and the excavation of minerals and concerning activities related to mining (Mining Act)[12]

The title is followed by the preamble, which starts with greetings from the king or queen. Currently, under Queen Beatrix, the preamble starts with:

[11] Wet van 5 juli 2006 tot samenvoeging van de gemeenten Medemblik, Noorder-Koggenland en Wognum en een wijziging van de grenzen van de gemeente Hoorn

[12] Wet van 31 oktober 2002, houdende regels met betrekking tot het onderzoek naar en het winnen van delfstoffen en met betrekking tot met de mijnbouw verwante activiteiten (Mijnbouwwet)

We Beatrix, by the grace of God, Queen of the Netherlands, Princess of Orange-Nassau, etc., etc., etc.
To all who see or hear this, greeting! Let it be known:[13]

For the previous monarchs, the preamble is of course slightly different, but the general structure remains the same[14]. After these greetings, some considerations and motivations for the law are mentioned in a sentence starting with *Whereas We have considered.*

Whereas We have considered that it is desirable to replace the existing system of legal regulations concerning the exploration for and production of minerals by new regulations, which fulfil current requirements, and to lay down a number of rules concerning certain activities.

The preamble ends with the sentence:

We, having heard the Council of State, and in having consulted the States General, have approved and understood, and we now approve and understand as follows:[15]

After the preamble, the body of law follows. The body is usually the largest part of the law, and contains all the rules. As such, it is the most important part for this research. The body of a law is divided into articles[16], which may be grouped into several super levels. These levels are named (from smallest to largest):

1. Section (*paragraaf)*
2. Division (*afdeling*)
3. Title (*titel*)
4. Chapter (*hoofdstuk*)
5. Part (*deel*)
6. Book (*boek)*

This order (book being the largest part, section being the smallest part) has been established by the official guidelines[17]. However, laws and regulations do deviate from this order. For

---

[13] Wij Beatrix, bij de gratie Gods, Koningin der Nederlanden, Prinses van Oranje-Nassau, enz. enz. enz.
Allen, die deze zullen zien of horen lezen, saluut! doen te weten:

[14] There are two obvious changes to the first sentence of the preamble: first of all, instead of Beatrix, the name of the previous monarch is used, and secondly, if it is a King instead of a Queen, the titles will be King and Prince rather than Queen and Princess. William I, II and III also listed the title Grand Duke of Luxembourg.
The most different opening occurred during Emma's regency. For laws from this period, the first line of the preamble is replaced with:

In the name of Her Majesty Wilhelmina, by the grace of God, Queen of the Netherlands, Princess of Orange-Nassau, etc., etc., etc.
We Emma, Queen-Widow, Regentess of the Kingdom,

In naam van Hare Majesteit Wilhelmina, bij de gratie Gods, Koningin der Nederlanden, Prinses van Oranje-Nassau, enz., enz., enz.
Wij Emma, Koningin-Weduwe, Regentes van het Koninkrijk,

[15] Zo is het, dat Wij, de Raad van State gehoord, en met gemeen overleg der Staten-Generaal, hebben goedgevonden en verstaan, gelijk Wij goedvinden en verstaan bij deze:

[16] If the law is very short, it may consist of a single article.

[17] **Aanwijzing 97, vierde lid**

example, the Act on the Organisation of the Market in Health Care[18] has three levels, chapter, section and division, with chapter being the largest part and division being the smallest. The guidelines also state that the level of "book" should only be used in a code of law.

In addition to these "named" levels, some older laws use one additional level, below section (and above the articles). These levels only have a title (indicated with a sub-header); they have no category (like *section* or *chapter*) or index.

The articles themselves can consist of the following:

- text;
- several numbered subparagraphs;
- text followed by several numbered subparagraphs (This usually occurs in amending laws, where the text is a sentence setting the scope for the changes that are described in the subparagraphs)(see section 4.4.3).

The subparagraphs can consist of text and subparagraphs. Most regular laws will not have subparagraphs within subparagraphs, but it is rather common for amending laws.

Within the text of a paragraph or subparagraph, one more structure element can occur: the list. A list consists of an introduction, several list items and optionally a conclusion. Though the list forms a single sentence (i.e. the individual list items are not complete sentences), the list items are often referenced separately (usually in conjunction with the introduction and the conclusion). Thus, it is necessary to identify the list items.

The body of the law is followed by a single, fixed sentence[19]:

It is hereby ordered that this act will be published in the State Gazette, and that all ministries, authorities, bodies and officials shall ensure that this act is accurately implemented.[20]

This sentence is followed by the signatures of the reigning monarch and the ministers involved. After this sentence, there can be some appendices.

2.2 Numbering

With the exception of the level of the sub-header, the various parts of the regulations are numbered. The precise numbering used differs between regulations. The Guidelines advise to use Arabic numbering for all levels above article. Articles should also be numbered using Arabic numbers. This can either be done throughout the entire regulation or per chapter, with the chapter number added in front of the article number. If there is only one article, it is identified as *Only article* (as opposed to *Article 1*). List items are numbered *a*, *b*, *c*, etcetera, with

---

Bij een verdeling op meer dan twee niveaus worden de onderdelen in volgorde van omvang "deel", "hoofdstuk", "titel", " afdeling" en "paragraaf" genoemd, met dien verstande dat in ieder geval de aanduidingen "hoofdstuk " en "paragraaf" worden gebruikt.

[18] Wet marktordening gezondheidszorg

[19] There exists one variation on this sentence, for those laws that apply not only in the Netherlands, but also in the Netherlands Antilles and Aruba, in which case the sentence starts with:
It is hereby ordered that this act will be published in the State Gazette, the Official Journal of the Netherlands Antilles and the Aruba Gazette

[20] Lasten en bevelen dat deze in het Staatsblad zal worden geplaatst en dat alle ministeries, autoriteiten, colleges en ambtenaren wie zulks aangaat, aan de nauwkeurige uitvoering de hand zullen houden.

sub-lists numbered *1°*, *2°*, *3°* etcetera. Change laws deviate from this numbering, and use capitalised Roman numerals for articles and capitalised letters for subparagraphs.

Many regulations do not conform to the Guidelines (many have been written before the guidelines were published). Hence, you may find structure elements that use a different numbering than presented above, and you may also find numbering using ordinals and lower case Roman numerals.

In addition, when pre-pending a chapter's number to an article's number, there are a several different styles being used:

- Separate using a full stop: *Article 3.15* (Income Tax Act 2001[21])
- Separate using a colon: *Article 3:15* (General Administrative Act[22])
- Separate using a space: *Article C 15* (Electoral Act[23])

When a law is expanded after its initial publication, newly inserted parts also need a number. Paragraphs and list items are often (but not always) simply renumbered. For other levels, this is not usually the case. Instead, the new part is given a number consisting of the number of the last (regularly numbered) previous part, with some suffix to distinguish it. For example, a new article inserted between articles *15* and *16* will often get assigned the number *15a*.

Common suffixes include:

- Letters: Used with Roman and Arabic numbers: *Article 212m* (Bankruptcy Act[24])
- Arabic numbers: Used with letters: *Article 1, item x1* (Higher Education and Scientific Research Act[25]) or *Article 1, item s.1.* (Medication Act[26])
- Latin numerical adverbs: *bis, ter, quater, quinqes, sexies, septies, octies*: *Article 12ter* (Military Officials Act[27])

These suffixes can be combined if a new part is inserted between two parts that already have a suffix in their number. For example, in the Penal Code, between articles *77h* and *77i*, an article *77h bis* has been inserted. Likewise, between articles *77w* and *77x*, the articles *77wa*, *77wb*, *77wc* and *77wd* have been added.

Of course, these suffixes can also be used in combination with combined chapter/article numbers, leading to more complex numbers like *article 9.30a* (Higher Education and Scientific Research Act).

2.3 Automated Detection

In order to detect the structure of a law, we will have to detect all the divisions described above. We will discuss the different "layers" separately, resulting in four steps:

---

[21] Wet inkomstenbelasting 2001
[22] Algemene wet bestuursrecht
[23] Kieswet
[24] Faillissementswet
[25] Wet op het hoger onderwijs en wetenschappelijk onderzoek
[26] Geneesmiddelenwet
[27] Militaire Ambtenarenwet 1931

1. separate the document into its basic parts (title, preamble, body, conclusion, signatures and appendices);
2. separate the body of the document into its different levels (chapter, etc.) and articles;
3. separate articles in numbered paragraphs, when appropriate;
4. split article and paragraph text into sentences and lists.

2.3.1 Separating the Different Parts of a Law

Separating the different parts of the document is an easy task, due to the fixed formulas that are used for the preamble and the conclusion. These fixed sentences can easily be found in the text, and with that, the entire preamble has been found (being the first sentence, the last sentence and all text in between). This procedure can even be simplified by identifying the preamble as the piece of text starting with the first occurrence of *We* and ending with the first occurrence of *Let it be known:*[28]. This of course assumes that *We* never occurs in the title of the document, and *Let it be known:* does not occur in the preamble other than at the end.

As the conclusion starts with a fixed sentence (see above), the start of the conclusion can be found in a similar manner. A third part of the law that can be easily identified are the appendices, which all have a header starting with *appendix* (*bijlage* in Dutch). The detection of headers is discussed in more detail below.

So, with these fixed elements, the preamble, conclusion and the appendices can be identified. This also gives us enough information to establish the boundaries for the other parts:

- the title is the first part of the document, before the preamble;
- the body is the part between the preamble and the conclusion[29];
- the signatures are the part between the conclusion and the appendices.

This approach can also be applied to other Dutch regulations. The overall structure is the same, and fixed formulas are used in those documents as well, though the precise text differs from those used in laws.

2.4 Splitting the Body of a Document in Parts and Articles

As mentioned above, the body of a document is split into several parts. These parts identify themselves by means of a header. Some examples of headers:

Article 1

Title 18. Absence, disappearance and determination of death in certain situations[30]

Title 1.1 Definitions and scope[31]

---

[28] With the exception of Emma's laws, this approach will cover all different first lines of the preamble, as they all start with *We*. A separate rule will still be needed to identify the preamble of the laws drafted during Emma's regency.

[29] Alternatively, the start of the body can be identified by the first header of an article or chapter, etc. that appears in the text. This may be useful for dealing with documents from which the pre-amble has been removed, and for dealing with the books from the Civil Code, which are often dealt with as if they were separate laws.

[30] Burgerlijk Wetboek Boek 1: Titel 18. Afwezigheid, vermissing en vaststelling van overlijden in bepaalde gevallen

§ 1. Definitions[32]

First division. On the declaration of bankruptcy[33]

As you can see, there are three elements in such a header:

- a category label, such as *Titel, Afdeling* or *§*;
- an index, such as *1* or *1.1* (as discussed in section 2.1);
- optionally, a title.

So, in order to detect a header, we can search for patterns consisting of a category followed by an index followed by any other words (which form the title). Such a pattern can be written down in a grammar or regular expression (see appendix A) for an example of this.

Since a header always appears on a new line, we can also include the newline tokens in our patterns, including a newline both before and after the header. This will prevent several false positives, as otherwise we would often detect a header where in fact there was only a reference to a certain part.

Still, it is not guaranteed that we will not find false positives. With this general structure, we will still find any line in the document that happens to start with a category label followed by an index[34]. There are two types of false positives we may encounter:

1. Sentences that happen to start with a category label followed by an index , such as:
   Article 12 does not apply.
   With the suggested regular expressions, this sentence could be classified as a header with category *article*, index *12* and title *does not apply*.
2. Lines that start midway a sentence because newlines have been applied to control the flow of the text:
   … as meant in↵
   article 12 …↵
   As above, this would result into a heading denoting article 12.

Some of these errors can be avoided by adding the information that a header never ends with a full stop or a colon, whereas regular sentences do. This will reduce the number of type #1 false positives, as any sentences that end on the same line will have a full stop.

Though in general the number of false positives for headers is very small (as is confirmed in our experiment described in section 2.8), we can further reduce it by making a number of assumptions.

Errors of type #2 can be avoided if it is assumed that a header is always preceded by a blank line. As there will not be a blank line between two lines of the same sentence, this will prevent a label followed by an index inside a sentence to be classified. It may also prevent errors of

---

[31] Algemene wet bestuursrecht: Titel 1.1 Definities en reikwijdte
[32] Drank- en Horecawet: § 1. Begripsbepalingen
[33] Faillissementswet, titel 1: Eerste afdeling. Van de faillietverklaring
[34] In the remainder of this section, I'll discuss the issues in terms of a category label followed by an index, though all of it also applies to an ordinal followed by a category label.

type #1, as this will limit the chance of such an error occurring to the first sentence of a paragraph.

Another way to avoid errors of type #2 is to assume that a header always starts with capital letters, while a reference within a sentence does not.

These assumptions are likely to hold, unless the input text is derived from a source that had more layout information. For example, an HTML or Word document may have had the headers marked-up, using that information to use increased line spacing (rather than a blank line) to visually separate the headers. If such a source is stripped of its layout information, the result can be a document in which the headers are not preceded by blank lines.

Another way to avoid false positives is to run a post-processor afterwards to eliminate false positives by looking at the numbering. A heading for article 12 appearing between articles 17 and 18 is unlikely to be correct, especially if another article 12 header has been found before.

The regular expressions presented above, together with their improvements, are sufficient to detect most common headers. However, there are some headers they cannot deal with, all related to titles appearing without a category/index pair.

As mentioned earlier, some of the older laws include a level marked by a sub-header, which consists of only a title. For example, the first articles in the Disownment Act[35] are grouped under the title *General provisions*[36]. Where most (possible) headers can be recognised because they start with a category/index pair, such a clue is missing for these headers. These lines can be distinguished from regular sentences in the text because, like other headers, they do not end in a full stop. However, not every line that does not end with a full stop is a title; it can also be a sentence that is continued on the next line. In order to recognise these titles, we need to look at the context. An incomplete sentence will be followed by the remainder of the sentence, and eventually a full stop. A title will be followed by a blank line or an article header.

2.5 Splitting Articles

In order to split articles into sentences, existing tools can be used such as the ANNIE Sentence Splitter (Maynard et al., 2002). Such tools split text into sentences based on punctuation, capital letters and whitespace. However, next to sentences, articles can also contain two other elements: subparagraphs and lists.

Unlike chapters, articles, etc. subparagraphs do not have a full heading. Instead, they only have an index, followed by a full stop[37], with the contents of the subparagraph following directly. The line preceding the index is either the article heading, a scope declaration (ending with a colon)(see 4.2.1) or the last sentence of the previous subparagraph (ending in a full stop). Subparagraphs can be numbered using Arabic numbers (used in regular laws and for sub-subparagraphs in amending laws) or capital letters (in amending laws). Like headings, these

[35] Onteigeningswet
[36] Algemeene bepalingen
[37] There exists a variation in which the index and the full stop are enclosed in square brackets, e.g. *[1.]* This is often used in older laws (those published around 1900).

indices can be detected using patterns. The start of a new subparagraph denotes the end of the next subparagraph.

List items are similar. They are numbered using lowercase letters (*a*, *b*, *c*, etc.), with sub-lists numbered *1°*, *2°*, *3°* etc. The line preceding the index is a blank line or a line ending in a colon or a semicolon or a semicolon followed by *and* or *or*. If list items have been detected, the introduction of the list (and optionally, the conclusion of the list) should also be marked. The conclusion will start at the end of the last complete sentence (detected using the sentence splitter), and end at the start of the first list item. If the last list item ends with a semicolon, then the text that follows, until the start of the next sentence, is the conclusion[38]. The items of a list and its sub-list(s) should be separated based upon the numbering used.

2.6 Recognising Quoted Text to be Added Elsewhere

Sentences that introduce new text in an existing law (insertions and replacements) contain this new text, which may include headings and indices, such as in the example below:

Act of December 17th, 2009 (Stb. 2010, 8), article I, sub D[39]
After chapter 8, a chapter is inserted, reading:

CHAPTER 8A. PARTICIPANT AND PARENT REPRESENTATION; NATIONAL REPRESENTATION DISPUTES COMMITTEE

TITLE 1. GENERAL PROVISIONS

Article 8a.1.1 Definitions
In this chapter is understood by:
...

This new text is not explicitly marked, and the headers contained in the new text can disrupt the parsing of the document[40]. Because of that, it is important to detect new text.

The new text starts after the colon. However, not every piece of text that follows a colon is new text. A colon can also indicate the start of a list, the start of several subparagraphs or the start of the actual definition in a defining sentence. Since a colon in these cases is followed by list items, subparagraphs or normal text (i.e. not a heading or index), the presence of a header for an article or higher-level structure element is a clear indication that it is the start of new

[38] If a sub-list ends in a semicolon, it may be followed by the next item in the main list instead of a conclusion.
[39] **Wet van 17 december 2009 (Stb. 2010, 8), artikel I, lid D**
Na hoofdstuk 8 wordt een hoofdstuk ingevoegd, luidende:

HOOFDSTUK 8A. MEDEZEGGENSCHAP VAN DEELNEMERS EN OUDERS; LANDELIJKE GESCHILLENCOMMISSIE MEDEZEGGENSCHAP

TITEL 1. ALGEMENE BEPALINGEN

Artikel 8a.1.1. Begripsbepalingen
In dit hoofdstuk wordt verstaan onder:
[40] Many sentences will also quote existing text. However, these quotes are always marked (using double angle quotation marks), and do not lead to complications in the way new text does.

text. If a colon is followed by an index for a list item or subparagraph, then there are some heuristics that may apply:

- If there is only one list item or subparagraph, then the text is more likely to be new text.
- If the first index is not *1* (or *a* or *I*, etc.), then the text is more likely to be new text.

However, these heuristics are insufficient to distinguish between all cases. A text replacing the first two list items will have the same structure as a list with two list items (i.e. two list items, one with index *a.* and one with index *b.*) and cannot be distinguished based on punctuation and headers. The same is true for text that does not contain headers or indices[41]. A better way to determine whether the text following the colon is new text is to classify the sentence that precedes the colon, using the methods and classification described in chapter 4. If the sentence is a replacement or an insertion, the text following the colon must be new text; otherwise, it is not.

Now that we know where a new text starts, the next step is to determine where it ends. There is no clear demarcation of the end of new text. Often, only one structure element (e.g. one article or one list item, etc.) is inserted, but even if we assume that the new text contains only one element, the end can be difficult to determine. For example, a subparagraph may contain new text containing an article with subparagraphs. The article ends at the beginning of the subparagraph following the original subparagraph, but based on structure alone, it is not always possible to distinguish between that following subparagraph and a subparagraph that belongs to the (new) article. In order to make the distinction, the index of the different elements needs to be taken into account. First of all, the format of the index can already give sufficient information to make the distinction. In an amending law, the articles and subparagraphs use a different index format than in a normal law[42]. Thus, when a amending law modifies a normal law, new articles and subparagraphs can be distinguished from regular text because of the format they use. However, there are a lot of situations that aren't covered by this: regular laws modifying regular laws, modification of different structure levels than articles and subparagraphs and even sometimes amending laws modifying amending laws. Here, we can apply a different heuristic: new text will always have contiguous numbering. So, as soon as an index does not follow the previous one, that marks the end of the new text. Of course, with this approach, if the indices in the new text by accident precede the index of the following regular text, the end of the new text is still not detected correctly. This will occur rarely, but if so desired, there are two more pieces of information that can be used to reduce the number of errors[43]:

---

[41] New text that does not contain indices or headings will not disrupt the structure recognition of the entire document if not detected, but to remain consistent, it is desirable to mark all new text as such.

[42] Normal laws use Arabic numbering for both articles and subparagraphs; change laws use capital Roman numerals for articles, and capital letters for subparagraphs.

[43] There are other pieces of information that seem helpful, such as:

- A law will not contain two articles (or chapter, etc.) with the same index twice. Likewise, an article will (usually) not contain two subparagraphs or two list items with the same index. So if an there are e.g. two articles with the same index, one of them must belong to new text.

- If it can be assumed that the law is contiguously numbered (i.e. no indexed structure elements have been repealed or inserted) then this information can be used to determine which parts of the text does not belong to the new text (e.g. if a law contains the articles 1 to 30, and the heading *article 12* occurs only once, then that article cannot belong to new text). Alternatively, one could go for the more relaxed assumption that if a structure element could belong to the regular text (i.e. it is not out of place there), it most likely belongs to the regular text and not to the new text.
- Insertions and replacements will often quantify the amount of elements that are inserted (e.g. *one article is inserted*). This can help in finding the end of the new text; if only one article is inserted, and the beginning of a new article is found, it is clear that that next article does not belong to the new text. However, this is no guaranteed method for finding the end of new text: if a subparagraph inserts an article, the next element may be another subparagraph and not an article.

2.7 Tables, Images and Other Content

Laws may contain tables, and could even contain images and other non-textual content. The way these are embedded in the document varies between document formats; thus, there is no generic way to detect them.

2.8 Experiment

The methods above for recognising the structure of a document have been (partially) implemented in a parser based upon the General Architecture for Text Engineering (GATE)(Maynard et al., 2002 and Cunningham, 2002).

GATE is an extensive, open-source tool for text processing. Among several other things, it allows the processing of text by means of the JAVA Annotations Pattern Engine (JAPE) (Cunningham, 2000). JAPE allows us to apply regular expressions to both strings and existing annotations (instead of just strings, as is common for regular expressions) and then apply new annotations.

We are using a pipeline of four GATE processing resources to recognise the starts and ends of different structure elements in a law.

First of all, we run a tokeniser, which splits the text into tokens such as words, numbers and punctuation. For this purpose, we use the GATE Unicode Tokeniser, which is part of the GATE library.

In general, this default tokeniser works fine for Dutch laws, though it is unable to recognise indexes that consist of a mix of numbers and letters, such as *13a*, *1.1* or *2:1:1*. It sees these as separate tokens, so:

- *13a* is seen as the tokens "13" and "a"*;*
- *1.1* is seen as the tokens "1", "." and "1";

- The elements of the law will appear in order, so any element that appears to be "out of order" must belong to new text.

This information is superseded by the information that the new text starts at the colon and is contiguously numbered.

- *2:1:1* is seen as the tokens "2", ":", "1", ":" and "1".

Our next step is to apply a set of JAPE rules that recognise such structures and merge the separate tokens into one token, in order to simplify the steps that follow[44].

The next step is to separate the sentences in the document. For this, we also use an existing GATE component, the ANNIE Sentence Splitter. In order to prevent the sentence splitter from incorrectly concluding that a full stop following an abbreviation actual marks the end of the sentence, it has to be provided with a list of abbreviations. As few abbreviations are used in Dutch laws, this list could be rather short (see table 1). For future applications, the list could be expanded with a complete list of Dutch abbreviations. The version of the sentence splitter that we used was case sensitive, thus the list did also contain variations for those abbreviations which were sometimes written with a capital letter and sometimes without. Furthermore, the sentence splitter would not recognise an abbreviation when it was preceded by a parenthesis. To circumvent this, the list of abbreviations also contained each abbreviation with an opening parenthesis pre-prended (e.g. not only *Trb.* was in the list, but *(Trb.* as well).

Abbreviation	Stands for
Enz.	Enzovoort
No.	Numero
N°.	Numero
Nr.	Nummer
Stb.	Staatsblad
Stcrt.	Staatscourant
Trb.	Tractatenblad

Table 1: Dutch abbreviations encountered in Dutch legislation

After running the sentence splitter, another series of JAPE rules is applied to recognise the headers in the document, the standard phrases which begin and end the preamble and the conclusion, and the indexes of subparts and list items.

After performing these four steps in GATE, the result is converted to XML using a postprocessor written in regular JAVA code rather than JAPE rules. This code copies most of the GATE annotations (such as the headings) and adds containers based on these headings. For example, for an article in the law that consists of two sentences, the GATE process will have detected:

- the header of the article, split in category and index;
- the end of both sentences;
- the header of the next article.

The postprocessor will generate XML based upon this information, in which it will also mark the complete header of the article as well as the category and index. It will mark the sentences, and it will mark the article as a whole (as derived from the start of this article and the start of the next article). The containers are added by a postprocessor instead of a JAPE transducer because of the varying order in which levels occur. For example, if a law follows the official guidelines, then a chapter is at a lower level than a division, so the start of a division will mark the end of a chapter (but the start of a chapter does not mark the end of a division). However, some laws deviate from these guidelines, so in order to determine the hierarchy, it is necessary

[44] Rather than using the default tokeniser and then correcting the errors, we could also have made a specialised tokeniser, which will probably have a better performance. For this experiment, we went with the current scenario as it was simpler to execute.

to keep track of the order in which the headings appear in the law, which cannot be done using JAPE regular expressions.

For our experiment, we have not implemented everything that has been described in this section. Most notably, the following is missing:

- recognition of the headers of appendices;
- recognition of quoted text;
- (full) distinction between the items of a list and its sub-list;
- recognition of the conclusion of a list.

The experiment was run on a dataset consisting of ten laws, randomly selected from a list of active non-modifying acts. These acts were downloaded from the portal of the Dutch government, wetten.nl. On wetten.nl, the acts are available in different formats: XML, RTF, HTML and ASCII. As the goal of this experiment is to see whether we can determine the structure in a document that does not yet have this structure marked, we used the acts in ASCII format rather than one of the other formats.

The acts from wetten.nl have line breaks dividing the texts in lines of approximately 80 characters. They contain a number of modifications compared to the official text. First of all, the header of a repealed article is still included in the text, excluding the title, followed by a note inside square brackets noting the fact that the article has been repealed, and when. Likewise, the header of a repealed chapter, department, etc. is also still included, excluding the title. These do not carry a note indicating their repealed status. A second modification is that any articles that contain modifications to other legislation have their content removed and replaced by a note inside square brackets noting the fact that the article includes changes to other legislation. Finally, some references have a number in square brackets added to them.

For the use in our experiment, these files were modified as follows. Each article in which the text had been replaced by a note (i.e. a repealed article or a modifying article) had the note converted to a sentence inside the article, by adding line breaks and a full stop, and removing the square brackets. The notes for references were removed, and the file was converted from ASCII to UTF-8 format.

By retaining the repealed and modifying articles, the test files still had the high-level structure of the complete laws, even though the articles themselves were rather simple. Table 2 shows the acts used, as well as their size measured in number of articles.

Act	Year	Articles			High level structure
		Total	Repealed	Modifying	
Civil Registry Fees Act *Wet rechten burgerlijke stand*	1879	7	0	0	-
Wrecks Act *Wrakkenwet*	1934	14	1	0	-
Act of June 27th, 1963 (Stb. 1963, 344) *Wet van 27 juni 1963 (Stb. 1963, 344)*	1963	9	0	1	-
Student Grant Act *Wet op de studiefinanciering*	1986	207	197	0	15 chapters, 21 titles
Act of February 1st , 1990 (Stb. 1990, 60) *Wet van 1 februari 1990 (Stb. 1990, 60)*	1990	3	1	0	-
Act on the Marking of Plastic Explosives *Wet inzake het merken van kneedspringstoffen*	1998	10	0	0	-
Act on the State Committee for International Private Law *Wet op de Staatscommissie voor het internationaal privaatrecht*	1998	6	0	0	-
Working Conditions Act *Arbeidsomstandighedenwet*	1999	70	22	0	8 chapters
Cableway Installations Act *Wet kabelbaaninstallaties*	2004	44	0	4	10 chapters
International Child Protection Implementation Act *Uitvoeringswet internationale kinderbescherming*	2006	32	0	3	11 chapters

Table 2: Structure parsing test set

These files were processed by the parser, and the results were checked by hand for errors. For all acts, the parser detected the high-level structure (introduction and conclusion, chapters, titles and articles) without errors. The errors that occurred all occurred within an article (and did not propagate outside of that article). Table 3 shows the number of articles that contain errors for each act: as an absolute number, as a percentage of the number of articles in that act and as a percentage of the real articles in that act (i.e. not counting the repealed and modifying articles with dummy content).

	Articles with Errors	% Correct	
		All articles	Real articles
Civil Registry Fees Act	1	86%	86%
Wrecks Act	0	100%	100%
Act of June 27th, 1963 (Stb. 1963, 344)	1	89%	88%
Student Grant Act	0	100%	100%
Act of February 1st , 1990 (Stb. 1990, 60)	0	100%	100%
Act on the Marking of Plastic Explosives	0	100%	100%
Act on the State Committee for International Private Law	0	100%	100%
Working Conditions Act	4	94%	93%
Cableway Installations Act	0	100%	100%
International Child Protection Implementation Act	0	100%	100%
Total	**7**	**98%**	**96%**

Table 3: Structure parsing errors

The assumption that the word *We* does not appear in the title and that therefore the first occurrence of *We* in the document may be seen as the start of the pre-amble (see section 2.3.1) held for these test files.

As was expected in section 2.4, the recognition of headers does not pose a problem. In fact, there were no errors with regards to the headers; all the errors found where related to lists. The Working Conditions Act contained five articles in which the parser had failed to properly process a list. In one article there was a sub-list that ends in a full stop followed by a

semicolon (i.e. ".;"). This pattern was not implemented in the parser, which concluded that the list had ended based on the full stop.

In another list, the introduction does not end with a colon. The parser was based on the assumption that each introduction does end with a colon, and failed to recognise the list.

In a third list, there is an abbreviation using dashes, in which the parser mistakenly identified one of the dashes as the start of a new item. This is in fact a bug in the parser, which does not check whether a new list item starts at a new line (but instead fully relies on the output of the sentence splitter to recognise the end of a sentence).

The fourth error occurred in a list where the last item ends in a comma instead of a semi-colon. The parser is based on the assumption that any item will end with either a semi-colon or a full stop, and missed the end of this item.

The errors in the other two laws were both also a list in which the introduction did not end in a colon. All three laws in which this error occurred did include other lists in which the introduction did end with a colon, so the use of the colon is not consistent within each law.

Bacci, Spinosa, Marchetti, and Battistoni (2009) have created a similar parser for Italian regulations. Their parser consists of two elements. For the preamble and the conclusion of a regulation, they have opted for a machine-learning approach, using Hidden Markov Models. This is because the patterns in the header and footer are not very precise. Multiple models are available; if the specific subtype (e.g. Act, Bill, Regional Act) of the document is known, a model specific for that subtype is used, if not a generic model is used. The second part of the parser targets the body of the document. It is a non-deterministic finite automaton that tries to parse the body of the document according to the restrictions set out by the NIR XML schema (see Biagioli, Francesconi, Spinosa & Taddei, 2003). It would seem that the preamble and conclusion of a Dutch law are both less informative and more structured than the preamble and conclusion of an Italian law. As shown in this chapter, the preamble and conclusion of a Dutch law consist mainly of fixed sentences, and are easily recognised. In addition, they have little content we wish to mark-up. Hence, for our parser for Dutch laws, we have no need of the fuzzy methods that Bacci et al. needed to handle Italian laws.

As far as the body is concerned, our methods do not differ much. A finite automata, as used by Bacci et al., has the advantage of knowing what is allowed at a certain point in the text, which makes it easier to interpret the text. However, few elements in Dutch laws are ambiguous in this respect; almost any header has only one interpretation. Using a finite automata might handle lists a bit better (as the index of a list is easily confused with that of a sub-list or a paragraph), but our method seems to perform fine on lists as well. On the other hand, finite automata have the disadvantage that they can get stuck if the input text does not conform to the prescribed structure. In this respect, our method is a bit more forgiving as it will start a new element whenever such a header is encountered, which means that an error is almost always contained within a paragraph or article.

Bacci et al. also indicate that in Italian law, like in Dutch law, the inclusion of quoted text makes the process of parsing the document much harder. The exact impact is somewhat

different, though. In Italian law, the entire quoted structure is marked with quotes, which means that they do not have problems determining where a quoted text starts and ends. However, the fact that the quoted text can start at any level of a legal document does increase difficulty, as does the fact that they need to determine whether a modification merely modifies text or also includes structure information. This problem does not occur in Dutch text as there is a difference in the markings of the two: quoted text consisting of a few words (no structure) is marked with double angle quotes, whereas bigger modifications are unmarked, but start on a new line.

2.9 Conclusions

In this chapter, we discussed the structure of Dutch laws and how the text of these laws gives us handles to automatically detect this structure. An experimental parser, implementing most of the ideas, performed very well, recognising the high level structure of law without errors, and correctly recognising the structure of 96% of the articles.

There are a few difficulties with automatic recognition of the structure of laws. Though most of them are solvable, they do increase the amount of effort required. First, there is the variation of indexes used, especially those indexes that contain spaces, and the occurrence of headers without a category. When a law has been drafted using the official guidelines, these difficulties should not occur. The guidelines specify what indexes to use, and do not allow headers without a category. This means that a structure recogniser for laws that are drafted under the guidelines can be simpler than presented above.

Two other issues are not covered by the guidelines. First of all, it is difficult to determine the end of a sub-list without checking the indexes. The guidelines specify that all elements of a list should end with a semicolon or a colon, with the exception of the last item. There is no specific rule for the last item of a sub-list, which sometimes ends with a semicolon (as it is the end of the item in the main list) and sometimes in a full stop (as it is the last item in the sub-list). Specifying a clear way to end a sub-list, such as a full stop followed by a semicolon, such as used in the Working Conditions Act (see section 2.8), will make it easier to make the distinction.

Another problem occurs with large blocks of quoted texts (i.e. quoted text containing entire structure elements rather than a few words). Determining the end of such texts is a complicated task, which would help if there was a clear marker indicating the end the quote. A disadvantage of this is that reliance on such a marker means that any errors introduced because the marker is missing will affect large parts of the document. In general, though, it should give better results with less effort. An alternative is to mark the beginning of each structure element that is part of the quoted text with a marker. This method is used in bills in the United States, which have an opening quote character at the start of each structure element, and a closing quote character at the end of the entire block. This method would also reduce the effort needed to recognise blocks of quoted text, and the impact of a missing marker would be less.

Though our experiment was aimed at Dutch laws, we expect that this method will also work for many other jurisdictions. Other Dutch regulations, such as royal decrees, also have to

follow the official guidelines, and do resemble the laws. Usually, the fixed formulas differ, so patterns have to be added to accommodate those. Also, such regulations may contain text that has not been split into numbered paragraphs, but consists of unnumbered paragraphs, which will also require some slightly different approach. In many other countries, similar structure for legislation is used, which suggests that a similar method will also apply on those regulations (which has been confirmed by the parser used by Bacci, Spinosa, Marchetti, and Battistoni, 2009).Thus, with stricter adherence to the guidelines, and some additions to those guidelines, the legislator can create documents that are easier to parse, which will result in better handling of new laws by a parser. It should be noted, however, that quoted structures of older laws inside new laws will still require the more complicated approach.