Automatic Assignment of Section Structure to Texts of Dutch Court Judgments

Trompper, M.; Winkels, R.

Published in:
Legal Knowledge and Information Systems

DOI:
10.3233/978-1-61499-726-9-167

Automatic Assignment of Section Structure to Texts of Dutch Court Judgments

Maarten TROMPPER and Radboud WINKELS [a]
[a] *Leibniz Center for Law, University of Amsterdam, PO Box 1030 P.O. Box 1030 NL 1000 BA Amsterdam, the Netherlands*

Abstract. A growing number of Dutch court judgments is openly distributed on Rechtspraak.nl. Currently, many documents are not marked up or marked up only very sparsely, hampering our ability to process these documents automatically. In this paper, we explore the problem of automatic assignment of a section structure to these texts. We experiment with Linear-Chain Conditional Random Fields to label text elements with their roles in the document (text, title or numbering). In this subtask, we report F_1 scores of around 0.91 for tagging section titles, and around 1.0 for the other types. Given a list of labels, we experiment with Probabilistic Context-Free Grammars to generate a parse tree which represents the section hierarchy of a document. In this task, we report an F_1 score of 0.92.

Keywords. Automatic Markup, Conditional Random Fields, Probabilistic Context-Free Grammars, Court Judgments

1. Introduction

The Council for the Judiciary in the Netherlands (Raad voor de Rechtspraak) publishes an open data set of Dutch case law online at Rechtspraak.nl, with cases dating back to about 1970. Most documents contain little semantic markup, such as element tags detailing the structure of (sub-)sections in a document.

It is useful to have such a section hierarchy, however. It is obviously useful for rendering documents to human users: a clear section hierarchy allows us to display a table of contents and to style section titles. Furthermore, because sections usually chunk similar kinds of information together, a good section hierarchy also allows search engines to better index texts by localizing semantic units, which in turn makes these documents better searchable for legal users. It is also a stepping stone to make the documents machine readable and more amenable to discourse analysis. A richly marked up document facilitates advanced text mining operations, such as automatically extracting the final judgment, extracting the judge's considerations, etcetera, as metadata.

Recently, more richly marked up documents have been published on Rechtspraak.nl. (See Figure 1.) Still, there is an overwhelmingly large portion of documents which contain no or only sparse markup. To illustrate: at the time of writing, 78.7% of all judgment texts on Rechtspraak.nl do not contain any section tag, implying that a large number of documents are barely marked up. These documents are mostly from before 2013. Older

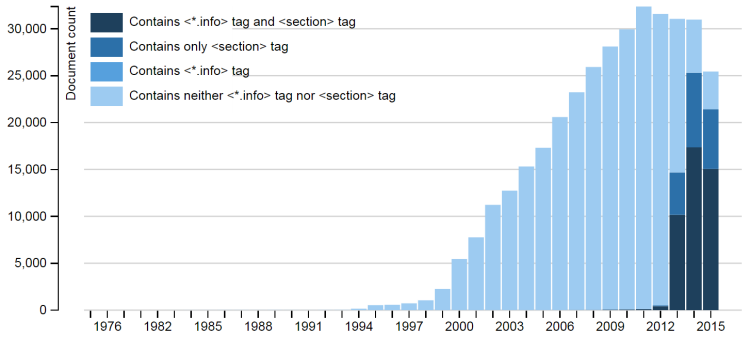

Figure 1. Chart demonstrating the number of judgment documents that are published each year by Rechtspraak.nl with different kinds of markup. In particular, we are interested in the number of `*.info` tags, which are headers that contain metadata about the case and section tags. In the following, we do not consider `*.info` tags; they serve here to illustrate richness of markup.

case law documents still produce legal knowledge, so it is desirable to have these older documents in good shape as well. The problem that we investigate in this paper, then, is whether we can enrich the markup of scarcely marked up documents in Rechtspraak.nl by automatically assigning a section hierarchy to the text elements. We divide this problem into the following subtasks:

1. Importing documents from the Rechtspraak.nl web service;
2. Tokenizing relevant text elements, explored in section 2;
3. Labeling these text elements with their respective roles, explored in section 3;
4. Combining the tokens in such a way that they represent the most likely section hierarchy, explored in section 4

For a comprehensive study on the legal and technical background of the digital publication of Dutch case law, see [14]. For a general overview of Rechtspraak.nl's web service, consider [12].

1.1. Related Work

The problem of automatically assigning semantic markup to plain-text documents has existed since the rise of hypertext in the late 1980s: see [7] for one historic example that predates XML. The general problem of automatic markup from digitally scanned documents is discussed in [2]. They define parsing a section structure as a task in macro-level markup. This is in contrast to micro-level markup, such as named entity recognition. They review some general solutions but argue that general automatic markup will remain a problem for a long time.

Indeed, most approaches to automatic markup are domain-specific. Somewhat recently, the problem has been addressed in legal informatics as well. A similar set-up to ours is described by [3], but applied to Italian law texts. They successfully apply Hidden Markov Models to distinguish headers and footers from body elements. Interestingly, they train a separate HMM for every law type. For parsing the section hierarchy in the body they use non-deterministic finite state machines, which corresponds to the class of (non-deterministic) regular expressions. The system shows some intolerance to syntactical errors in the input but can handle common input issues. The system uses much fewer

features than ours, which may be explained by the fact that legislative texts tend to have a stricter structure than judgment texts even though it also tends to be more deeply nested.

Another approach to parse the structure in legislation is described by [6]. This system exploits the fairly strict conventions found in section titles in Dutch legislation by using rule-based pattern recognition to correctly identify 96% of articles. A similar system might work for court judgments as well, but is probably more labour-intensive because the writing style is less strict for court judgments, and so more rules must be maintained. In any case, using stochastic methods allows for some noise in the data, making it easier to deal with typing errors or non-adherence to expected patterns.

More towards metadata generation are [15] and [8], the latter reporting "an urgent need to automatically identify information in legal texts" and both describing automated techniques for mining legal arguments from court judgments. Extracting legal arguments is somewhat of a holy grail in text mining court judgments and benefits a lot from semantic annotations. Our present work has a supportive role in this objective by offering a robust, modular XML markup pipeline that may be extended to other corpora.

2. Tokenization

Regarding tokenization, we need to do some forward thinking in order to determine how to split XML texts from Rechtspraak.nl. We assume a text to be decomposable into a list of tokens, which correspond to the terminal nodes in a section hierarchy. We use the following four terminal types in our section hierarchy: numberings, title texts, text blocks and newlines. These types were inspired by the existing XML tags of Rechtspraak.nl.

One complication with creating a list of tokens is that Rechtspraak.nl delivers an XML tree, which is potentially more rich than the linear list that we reduce the document to. But the existing grouping tags are often of negligable semantic value. Also, classifying a tree structure of tokens instead of a linear list requires a much more complicated pipeline, although it can be done efficiently with CRFs (as in [4]). So we have chosen to ignore most of those 'higher-level' tags.

Our tokenization algorithm returns a linear sequence of tokens, which serves as input for our tagging operation. In the next section, we explore how to tag a list of text elements with the four target labels introduced above.

3. Tagging Elements

In this section, we consider how to label these tokens with any of the four labels introduced before. Even as a human reader, it can be hard to distinguish what should be called a section, and thus what is a section heading. This means that there is some subjectivity involved in tagging. Consider, for example, a numbered enumeration of facts which might either be considered a list or a section sequence. For our purposes, we call a 'section' any semantic grouping of text that is headed by a title or a number.

We experiment with Linear-Chain CRFs for labeling the tokens, and we compare the results to a hand-written deterministic tagger that utilizes features that are largely the same as those used by the CRF models. It turns out that both approaches score around 1.0 on all labels except section titles. For section titles, Linear-Chain CRFs significantly outperform the hand-written tagger in terms of recall, while trading in some precision.

Lineair-Chain CRFs differ from the closely related Hidden Markov Models in one important regard: instead of modeling a joint probability $p(\mathbf{x}, \mathbf{y})$ of the observation vector $\mathbf{x}$ and label vector $\mathbf{y}$ occurring together, we model the conditional probability $p(\mathbf{y}|\mathbf{x})$ of labels given the observations. For an exposition on Lineair-Chain CRFs as applied to the current project, refer to [13]. For a more thorough tutorial into CRFs generally, including skip-chain CRFs, one may refer to [11].

We define around 250 different feature functions on our tokens, which consist mostly of regular expressions for known section title patterns. In addition, a number of features are defined for the text length, whether the element contains bracketed text, whether the numbering is in sequence with a previous number, etcetera.

3.1. Results

We measure classifier performance with the often-used F_1 and $F_{0.5}$ scores. Generally speaking, F_β-scores are composite metrics that combine the precision and recall of a classifier: $F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$. Precision is the fraction of true positives out of all positives, recall is the fraction of true positives out of all relevant elements and $\beta \in \mathbb{R}$ is a number that represents the number of times we place the importance of recall above that of precision. For $\beta = 1$, precision is equally as important as recall, and for $\beta = 0.5$ precision is twice as important as recall.

For all tokens except for section titles, all models yield F-scores between 0.98 and 1.0. Section titles are harder to label, so in Table 1, we consider the F-score for section titles specifically. We compare using CRFs to a manually written tagger that uses many of the same features as the CRFs, but decides on labels deterministically using if-else rules. We see that using CRFs significantly improves recall, but also hurts precision.

Although both methods of tagging can in theory produce perfect scores, we feel that using CRFs is a superior method to hand-writing rules for labeling. CRFs are considerably less labour-intensive to maintain because they are able to automatically make refined decisions by taking a lot of evidence into account, including any possible hand-written rules.

Table 1. F-scores for tagging section titles

	Precision	Recall	F_1	$F_{0.5}$
Deterministic tagger (baseline)	0.95	0.74	0.83	0.90
CRF	0.91	0.91	0.91	0.91

4. Parsing a Section Hierarchy

After we have labeled a sequence of text elements, we wish to infer the section hierarchy. That is: we need to invent some procedure of creating a tree structure in which these tagged text elements are the leaf nodes, and may be children of ‘section’ nodes. This problem is very much akin to constituency parsing for natural languages, and that is why we approach the problem as parsing a token sequence with a Probabilistic Context-Free Grammar (PCFG).

A lot of work has been done in parsing (Probabilistic) Conext Free Grammars in applications of natural language processing and parsing programming languages. More recently, PCFGs have been used for other applications such as modeling RNA structures, as in [9]. Because of the broad interest in CFGs, a number of efficient parsing algorithms have been set forth. In our work we implement an efficient probabilistic Earley algorithm, after [10].

4.1. Results

Evaluating performance on a parse tree is not as straightforward as it is for classification. Like in the previous section, we evaluate our grammar using an F-score, but notions of precision and recall are harder to define for constituency trees. We use a metric known as PARSEVAL (due to [1]) with labeled precision and labeled recall as in [5]. In this metric, precision is the fraction of correct constituents out of the total number of constituents in the candidate parse, and recall is the fraction of correct constituents out of the total number of constituents in the correct parse, where 'correct constituent' means that each non-terminal node has the same label and the same yield, and yield is defined as the list of leaf nodes in a parse tree in order. Over a set of 10 random documents, we report an average F_1-score of 0.93 (precision 0.93; recall 0.92).

Delving into problematic parses, we see that there are a number of recurring types of errors that our parsing grammar makes. Firstly, it often occurs that subsections are not preceded by a full numbering. For example, consider a section numbering sequence such as the following: 1, 2, 3.1, 3.2. Our current grammar assumes that section 3.1 is a subsection of section 2, since section 2 is the first preceding supersection to 3.1. Another issue is that the grammar has difficulty in deciding whether non-numbered sections should be subsections or not. Indeed, this can be difficult to determine based purely on typography. These are not fundamental problems, however, and can be resolved by refining the document grammar or parsing algorithm.

5. Conclusion and Future Work

We have successfully demonstrated a method to assign a section hierarchy to documents of Dutch court judgments. We have described a procedure to assign types to document elements of either title, numbering, newline or text block, reporting F_1 scores of 0.91 for section titles and between 0.97 and 1.0 for the other types. We have also reviewed a procedure to organize those elements into a section hierarchy using Probabilistic Context-Free Grammars, reporting an F_1 score of 0.92.

Whether these results are good enough to be used in practice depends on one's tolerance to inaccuracies. We prefer errors where we miss opportunities to enrich data to errors that produce false information, so a low recall is preferable to low precision. The scores obtained for the classifier and parser are promising, but the procedures could still be optimized to the corpus. In any case, mislabelings do not distort the text in such a way to render it illegible, so we may be somewhat forgiving of errors.

The broad objective of the above experiments is to make documents of case law machine readable by automatic process, so future work is focused on improving accuracy and scope of the enrichment procedure. As discussed before, both tasks of classi-

fication and section parsing can be improved by provisioning for common errors. We may improve the usefulness of the enrichment procedure further by including automatic micro-level markup, such as annotating the names of relevant entities (court, judge, etc.). All this enrichment may finally lead to automatically producing reliable discourse-level metadata, such as abstract legal arguments, by making it easier to localize information within a document.

5.1. Dissemination

All source code is published under a permissive MIT license on Github and the Central Repository. The experiments are bundled in one Java library for fetching and enriching documents, available on Github at https://github.com/digitalheir/java-rechtspraak-library or under `org.leibnizcenter:rechtspraak` in the Central Repository.

References

[1] Steven Abney, S Flickenger, Claudia Gdaniec, C Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, et al. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics, 1991.

[2] Mohammad Abolhassani, Norbert Fuhr, and Norbert Gövert. Information extraction and automatic markup for XML documents. In *Intelligent Search on XML Data*, pages 159–174. Springer, 2003.

[3] Lorenzo Bacci, Pierluigi Spinosa, Carlo Marchetti, Roberto Battistoni, I Florence, I Senate, and I Rome. Automatic mark-up of legislative documents and its application to parallel text generation. In *Proc. of LOAIT Workshop*, pages 45–54, 2009.

[4] Joseph K Bradley and Carlos Guestrin. Learning tree conditional random fields. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 127–134, 2010.

[5] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics, 1997.

[6] Emile de Maat, Radboud Winkels, and Tom van Engers. *Making Sense of Legal Texts*. PhD thesis, University of Amsterdam, 2009.

[7] Richard Furuta, Catherine Plaisant, and Ben Shneiderman. Automatically transforming regularly structured linear documents into hypertext. *ELECTRON. PUBL.*, 2(4):211–229, 1989.

[8] Marie-Francine Moens, Caroline Uyttendaele, and Jos Dumortier. Information extraction from legal texts: the potential of discourse analysis. *International Journal of Human-Computer Studies*, 51(6):1155–1171, 1999.

[9] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I Saira Mian, Kimmen Sjölander, Rebecca C Underwood, and David Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic acids research*, 22(23):5112–5120, 1994.

[10] Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational linguistics*, 21(2):165–201, 1995.

[11] Charles Sutton and Andrew McCallum. *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press, 2006.

[12] Maarten Trompper. Open legal data survey; Dutch case law. `https://leibniz-internship-report.herokuapp.com/eu-legal-data-survey/nl#rechtspraak.nl`, 2014.

[13] Maarten Trompper. Automatic assignment of section structure to texts of Dutch court judgments. Master's thesis, Utrecht University, jun 2016.

[14] Marc van Opijnen. *Op en in het web: Hoe de toegankelijkheid van rechterlijke uitspraken kan worden verbeterd*. PhD thesis, University of Amsterdam, 2014.

[15] Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward. Approaches to text mining arguments from legal cases. In *Semantic processing of legal texts*, pages 60–79. Springer, 2010.