Parsing User Queries using Context Free Grammars

Kees van Noortwijk
vannoortwijk@law.eur.nl
Erasmus School of Law
Rotterdam, The Netherlands
Rechtsorde BV
Den Haag, The Netherlands

Christian F. Hirche
hirche@me.com
Rechtsorde BV
Den Haag, The Netherlands

ABSTRACT

In legal information retrieval, query cooking can significantly improve recall and precision. Context free grammars can be used to effectively parse user queries, even if the number of items to recognize is high and recognition patterns are complicated.

CCS CONCEPTS

• **Information systems** → **Query intent**; *Link and co-citation analysis*; • **Applied computing** → *Law*.

KEYWORDS

legal information retrieval, query cooking, text parsing, context free grammars

ACM Reference Format:
Kees van Noortwijk and Christian F. Hirche. 2023. Parsing User Queries using Context Free Grammars. In *Proceedings of The first international workshop on Legal Information Retrieval, to be held at ECIR 2023 (LegalIR '23).* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

The use of digital information sources these days is a vital part of the work of almost every lawyer, now that traditional information sources such as books and journals to a large extent have been replaced by their digital counterparts.[1] The retrieval systems used to search these digital collections and retrieve relevant legal documents usually have access to millions of documents. Because of that, even basic queries consisting of just one or two keywords usually deliver a few relevant documents, be it as part of a much larger set of not-so-very-relevant ones. However, that is often not enough for professional users, who not only want information that is as *complete* as possible, but who also do not want to wade through large amounts of irrelevant stuff to eventually find what they are looking for. In other words, a legal information retrieval system should be finetuned to deliver optimal recall and precision, with results carefully ranked according to their relevancy. In [5] it is argued that specifically recall – the ratio between the number of relevant documents retrieved and the number of such documents being present in the database – is important from the legal perspective, but is often also difficult to measure.

LegalIR '23, April 2, 2023, Dublin, Republic of Ireland

To optimize recall, it is important that the *initial* query places all documents that could possibly be relevant in the initial list of search results. This list can subsequently be filtered, using 'facets' like the type of document, the area of law, etcetera, to increase precision. But documents absent from that initial list will not be part of the final set, no matter how exact the filtering options will be set. That is why it is important to optimize the results of the initial query: what is missed there, cannot be regained in subsequent (filtering) steps. One way to improve the quality of the initial query is to not take the terms in that query for granted, but to use algorithms to find out what these terms might *mean* and what the intention of the user might be to use them in the query. For instance, if the user would have entered a number followed by the full name or abbreviation for a certain piece of legislation and the words 'case law', it will probably not be useful to just return documents containing this combination of words/items. Instead, the system should look for case law documents containing decisions relating to the article of law that can be derived from the number and the law name. The latter information could be present in the 'body text' of a (case law) document, but also in metadata that are part of it.

This is only one example of a possible improvement of query effectiveness, achieved through analysis – followed by automatic adjustment – of a user query before that query is executed. Another example might be the automatic addition of synonyms to a search query, or the recognition of well-known legal terms to add corresponding articles of law or even certain case law identifiers to the query. This process of analysing and adjusting a query is called *query cooking*. It is probably used in the majority of document retrieval systems these days, but arguably is particularly useful in collections of documents all relating to a particular field or subject area, because in that case algorithms and rules can be applied that relate to that particular subject area. For instance, in the field of law, rules can be defined that are capable of recognising articles of law, case law identifiers or 'nicknames' that might be in use to refer to these, as well as references to legal textbooks and law guides. This paper assesses methods to implement such rules in a retrieval system for various types of legal content, paying attention to functionality as well as to maintainability.

2 QUERY COOKING

A query cooking function in a document retrieval system can perform several functions, such as:

- pattern matching, to find terms or groups of terms that conform to a certain specification; for instance: a number in a particular format, such as the Celex numbers that are used to identify EU documents[2], or a number preceded or followed

by a non-numeric string, which combination could designate a particular law article;
- word group identification, to automatically search sets of terms that constitute one concept as a 'phrase' (as if it would have been enclosed in double quotes);
- identification of known (legal) concepts, nicknames and other keywords, which can be searched by adding to the query corresponding (case law) identifiers, articles of law or other references.

A common characteristic of these functions is that known terms (from previously compiled lists) and patterns need to be identified within the query. Specifically for identifying patterns, regular expressions [4] are often used. A simple regular expression to recognise a Celex-number could for instance be:

```
[0-9cCeE]\d{4}\D{1,2}\d{3,4}.*
```

Law articles are already more complex to cover, as they consist of at least two elements (the law abbreviation, to be matched against a list, and the article number). However, as Van Opijnen et al. ([6], par. 3.4) already stated, regular expressions can have drawbacks in large-scale environments, as multiple types of items to recognize and many possible matches can lead to very complicated setups that can be difficult to debug and maintain. Instead, they proposed an alternative for the specific task of recognising legal references in document texts, in the form of *grammars*, in particular so-called Parsing Expression Grammars (PEGs). A grammar is a set of rules used to recognize language elements. In the case of PEGs, this recognition is performed *without ambiguity*, in other words, each string that is parsed can have only one valid 'parsing tree' at the most. Any possible choices that might result from the grammar are considered in an ordered form, choosing the first valid option while ignoring subsequent ones. Theoretically, this can be expected to work well for parsing strings containing strictly-defined legal references, as such references can be resolved to one and only one publication.

In practice, however, precluding ambiguity when parsing legal references does not always work well. In some cases, two or more publications can share the same title, abbreviation, or other identifying designation. Then, a legal reference containing such an ambiguous designation can become ambiguous itself. Aggravating the problem, in case of parsing of user queries, legal references are often short and miss context, which makes them more prone to ambiguity.

In addition, even when a legal reference can be parsed unambiguously, its surrounding context, which usually is just natural language content, cannot be parsed unambiguously (see for example [3]). Therefore, when attempting to use PEGs to parse legal references inside a longer text, a two-step approach is necessary. In the first step, unambiguous legal references must be identified and separated from surrounding text. In the second step, the actual parsing will occur.

3 CONTEXT FREE GRAMMARS

As an alternative approach, which does not suffer from these issues, so-called Context Free Grammars (CFGs) can be used. These grammars allow for ambiguity, which means that, in principle, parsing a text could result in several alternative parse trees. Choosing one parse tree over the other is done using priorities assigned to parse rules. First, this makes parsing ambiguous legal references possible. Second, CFGs can also be used to parse the text surrounding a legal reference, which cannot be parsed by a PEG, eliminating the need to use a two-step approach.

In the case of a user search query, ambiguous ways of parsing will lead to alternative interpretations of the query. These alternative interpretations can either be discarded or can be used to create a (processed) query containing elements that are to be searched alternatively (Boolean: OR). Usually, that is exactly what is needed here: queries are seldomly completely exact and can contain combinations of terms of which only a subset is present in the document the user intends to find. Query cooking can help to make the most of what was input, at the same time providing information that can subsequently be used for the optimal ranking of search results – for instance by adding 'boosting' to documents that exactly match recognised elements.

Query parsing using custom-made CFGs is now used in the Dutch legal information retrieval system Rechtsorde. It uses an implementation of an Earley parser. A slightly simplified excerpt of the grammar to identify a reference to the law "Burgerlijk Wetboek" (the Dutch Civil Code) is shown below:

Listing 1: Excerpt of example grammar to recognise legal references

```
text: (legal_reference delimiter |
    any_other_text delimiter | delimiter)*
legal_reference: regular_law | bw |
    publication //...
any_other_text.-100: ANY_CHARACTER //low
    priority to default to a legal reference

bw: [bw_law_prefix SEP] bw_references |
    identifier_bw
bw_references: identifier_bw [SEP]
    bw_book_reference [SEP
    bw_article_reference [SEP
    part_and_sub_ref]]
| bw_book_reference SEP identifier_bw [SEP
    bw_article_reference [SEP
    part_and_sub_ref]]
| bw_book_reference SEP bw_article_reference
     SEP identifier_bw [SEP part_and_sub_ref
    ]
| bw_book_reference SEP bw_article_reference
     SEP part_and_sub_ref SEP identifier_bw
    //...
bw_book_reference: [KEYWORD_BOOK SEP]
    NUM_BOOK
bw_article_reference: [KEYWORD_ARTICLE SEP]
    num_article_bw

KEYWORD_BOOK: "boek"
NUM_BOOK: "1".."8" | "10" | "7a"
//...
```

The grammar in Listing 1 shows the hierarchical construction of a grammar. This means that a starting rule is defined by one or more other rules or terminals, which are defined by one or more other rules or terminals, and so on. The hierarchy ends when a rule is exclusively defined by terminals, which are a character, string, or regular expression. In the example, the starting rule is called `text`. This rule's definition allows for zero or more instances of either (1) a legal reference (rule: `legal_reference`) followed by a delimiter (rule: `delimiter`) or (2) something else (rule: `any_other_text`) followed by a delimiter. One level below, the rule `legal_reference` is defined as the combination of the rules `regular_law`, `bw`, `publication`, and others not shown in the excerpt. The rule `any_other_text`, on the other hand, refers to the terminal `ANY_CHARACTER`, which might be any character or string. The text "-100" behind the rule name indicates that this rule has a low priority and whenever a rule with a higher priority can match, it will have preference over this rule.

Using this grammar to parse a short example text *reference to BW Boek 7* results in a parse tree shown on in Figure 1 on page 4. The parser matches *reference* as rule `any_other_text`, the space character after that as `delimiter`, *to* as rule `any_other_text`, and the space after that again as `delimiter`. Subsequently, *BW* fits the definition of the rule `identifier_bw` (not shown in the excerpt), the space fits the terminal definition for `SEP` and *Boek 7* fits the definition of `bw_book_reference`. Taken together, `identifier_bw`, `SEP`, and `bw_book_reference` fit the definition of `bw_references`, which, in turn, fits the definition of rule `bw`. Several other parse trees are also possible for that text, but have been discarded based on the low priority of the `any_other_text`-rule. Based on this tree, the query cooking process can create a directed query for a legal reference, which is optimised for the field and the format in which these references appear in document metadata.

Implementing this grammar-based form of query cooking has not only improved the overall reliability and speed of the recognition of query elements, but has also made it possible to, in principle, recognise an unlimited number of elements in any single query and process the results of that accordingly. At the same time, maintainability has greatly improved. This result would not have been possible using regular expressions or similar programming techniques.

REFERENCES

[1] David W. Dunlap. 2022. So Little Paper to Chase in a Law Firm's New Library. *New York Times* (October 2022).

[2] EU Publication Office 2000. *Celex Numbers.* Retrieved January 20, 2023 from https://eur-lex.europa.eu/content/help/eurlex-content/celex-number.html

[3] Bryan Ford. 2004. Parsing expression grammars: a recognition-based syntactic foundation. In *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 111–122.

[4] Walter L. Johnson, James H. Porter, Stephanie I. Ackley, and Douglas T. Ross. 1968. Automatic generation of efficient lexical processors using finite state techniques. *Commun. ACM* 11, 12 (December 1968), 805–813.

[5] Kees van Noortwijk. 2017. Integrated Legal Information Retrieval; new developments and educational challenges. *European Journal of Law and Technology* 8, 1 (2017), 1–18.

[6] Marc van Opijnen, Nico Verwer, and Jan Meijer. 2015. Beyond the Experiment: The Extendable Legal Link Extractor. In *Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts, held in conjunction with the 2015 International Conference on Artificial Intelligence and Law (ICAIL), June 08 - 12, 2015, San Diego, CA, USA.*

Received 20 January 2023; accepted 3 March 2023

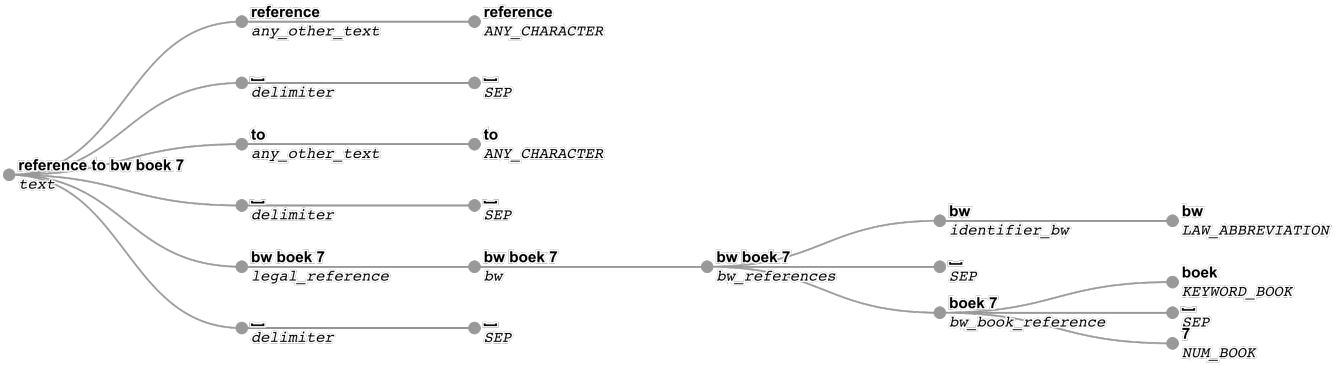

Figure 1: Parse tree for query string "reference to bw boek 7"