Better Access to Case Law

# D2.1 Linking Data

## analysis and existing solutions

This document presents the Linking Data problem analysis. A survey of the existing solutions for legal link extraction is presented and general considerations about common approaches and differences are given. A list of general requirements for the extraction tool is obtained as a result of the analysis of use cases and user stories provided by the partners. A viable solution for the realization of an extraction tool that is extensible to any European jurisdiction, taking into account the know-how gained by all the solutions and approaches presented, as well as the collected requirements, is described.

**2 May 2016**

Tommaso Agnoloni, Lorenzo Bacci (ITTIG-CNR)

This project is co-funded by the European Union

**Reference Number:** 00001
Florence, 2 May 2016

# D2.1 Linking Data: analysis and existing solutions

## Table of Content

## REVISION HISTORY

| Revision | Date | Author | Organisation | Description |
|---|---|---|---|---|
| 1.0 | 22.02.2016 | Tommaso Agnoloni, Lorenzo Bacci | ITTIG-CNR | First release |
| | | | | |
| 2.0 | 16.03.2016 | Tommaso Agnoloni, Lorenzo Bacci | ITTIG-CNR | Revision |
| | | Monica Palmirani | CIRSFID-University of Bologna | |
| | | Marc van Opijnen | UBR\|KOOP | |

**Statement of originality**

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

**Disclaimer**

This publication has been produced with the financial support of the Justice Programme of the European Union. The contents of this publication are the sole responsibility of the authors and can in no way be taken to reflect the views of the European Commission.

## TERMINOLOGY

Some terms used in this document have a specific meaning. To avoid any misunderstandings, such terms are defined in the table below[1]

| Term | Explanation |
|---|---|
| Legal citation | An explicit, plain text, human readable mention of a legal document (or its fragment) as found in another text |
| Legal reference | A machine readable representation of the same data specified in the citation, normalized using a naming convention |
| Legal identifier | A string univocally associated to a document (or its fragment) to identify it |
| Legal link | A generic term including any connection (human or machine readable) to a legal document as found in another text |

[1] See also https://wiki.oasis-open.org/legalcitem/FundamentalRequirements3rd

# 1. AUTOMATIC LEGAL LINK EXTRACTION

## Objectives and task overview

The main objective of the Workstream is the improved accessibility of case law, for instance within the ECLI Search Engine of the European eJustice portal, by having computer readable – and hence searchable – legal references within judicial decisions, especially to jurisprudence and national/European legislation (*linked data*).

This is accomplished by adding the optional references metadata, expressing the relations between the judicial decisions and national and European legislation, preceding case law and other legal sources. These reference structures, which are made visible by the judge by way of his citations to these sources, are extremely important for effective and efficient legal research, but currently not machine-readable.

Since their numbers run in the millions, manual tagging is not feasible. Therefore, the objective of WS-2 is the design and development of a common European open source software infrastructure for the automatic legal link extraction from texts that can be implemented at the national level.

Although this will not be implemented for all languages and jurisdictions, a major step in this direction will be made. At least one full national implementation will be developed.

## Problem and scope

Despite the fact that drafting rules and citation guidelines exist at both national and EU level, exceptions to the recommendations are very frequent and in practice legal citations have a diversity of styles, variants and formats.

- legal citations are language and jurisdiction dependent, depending on the legal tradition of each country;
- each country has its own established practice for the citation of national legislation and national case law;
- within each national member state, citations of case law documents differ by source and target court, citation attribute order, spelling variants, lack of normalization of dates and numbers, etc.;
- within each national member state, citations of legislative sources differ as well by target legal source, national authority, etc.;
- within each national member state the official judicial documents might be published in different collections or official journals, so the information about the legal citation are multiple;
- national jurisprudence also contains citations to EU legal sources, (legislation and case law) for which specific citation practices (and their variants) in the national language are used;

At least four different ways of citing legal sources are used in practice:

- by citation attributes: typically a composition of attributes like date, number, type of document, court or authority name, name of the parties in some jurisdiction[2] and so on, including all possible kind of variants and abbreviations;
- by title or short title: the citation reproduces the official title of the referred act or its abbreviation;
- by alias: using a common (colloquial) name of the cited source (this includes abbreviations);
- by identifier: the citation is done using a standard identifier (national, European e.g. ECLI).

Moreover, multiple citations are widely used in legal texts. They can be citations to multiple sources of the same typology, or of the same court. Or they can be citations to multiple partitions of the same cited document.

The problem scales in complexity when aiming to cover this diversity for each member State of the EU and for the EU supranational jurisdictions (EU legislation, EU case law of the European Court of Justice (CJEU) and European Court of Human Rights (ECHR)), and for different languages.

Despite this complexity, legal link extractors exist and have proven their efficacy in several languages and jurisdictions.

The challenge of the BO-ECLI Linking Data workstream is to provide common (cross language and cross jurisdiction) software allowing the specialization of common link extraction services and functionalities to national jurisdictions.

## Methodology

To this aim the chosen approach has been to take stock of existing solutions in order to collect approaches, analysis and solutions on which to build the EU common platform.

In order to describe the most relevant features of the existing approaches and the implementation choices by each solution provider, the first phase of the Workstream activities has been dedicated to the collection of schematic factsheets for existing national solutions.

Similarly, all the partners and associate partners of Workstream 2 have been asked to provide, based on the know-how gained in their national experience, a document describing

[2] http://liv.ac.uk.libanswers.com/faq/49340

use cases and user stories about the main functionalities and usage scenarios for the link extraction tool.

In Section 2 the results of the analysis of the national solution factsheets are reported in terms of a synthetic survey and general considerations about common approaches and differences.

The results of the analysis of use cases and user stories documents and a list of general requirements for the extraction tool are illustrated in Section 3.

After taking into account all the existing solutions, know-how on the domain, approaches and requirements, Section 4 describes a viable solution for the realization of a new extraction tool that is extendable to any European jurisdiction. In particular, the scope and the responsibilities of *common platform* and *national implementation* are defined and the interactions among common platform, national implementation and external resources are illustrated, along with an explanatory diagram.

## References

- BO-ECLI project Grant Agreement JUST/2014/JACC/AG/E/-JU/6962 - Annex I Project description and Implementation
- Marc van Opijnen, Canonicalizing Complex Case Law Citations (2010). Legal Knowledge and Information Systems - JURIX 2010: The Twenty-Third Annual Conference, Edited by Radboud G.F. Winkels. Available at SSRN: http://ssrn.com/abstract=2046274
- Marc van Opijnen, Searching for References to Secondary EU Legislation, in S. Tojo, ed., Fourth International Workshop on Juris-informatics (JURISIN 2010)
- EUCases project (FP7-ICT-2013-SME-DCA) Deliverable D3.6, Report on Linking Tools

# 2. ANALYSIS OF EXISTING NATIONAL SOLUTIONS

A *national solution factsheet* in the form of a document template have been distributed to the partners in order to obtain a schematic overview of the existing link extraction solutions.

The complete national solution factsheets provided by the partners of the Workstream (CIRSFID University of Bologna, UBR|KOOP, Cendoj, ITTIG, University of Torino) are reported in the Annex 1 of this document.

## Survey

This section briefly illustrates a summary of the existing solutions in the field of legislative and case-law link extraction.

CIRSFID-University of Bologna developed *SPeLT-ref*, a tool from the SPeLT framework. SPeLT-ref is written in PHP and JavaScript and it is based on the definition of macros of regular expressions in several JSON configuration files. It runs as a web service.

UBR|KOOP presented *eXtendable Legal Link eXtractor*. The software relies on XML, XSLT and Java. It uses Apache Cocoon in order to implement a pipeline of components that perform the identification of titles and aliases (Trie-based dictionaries), the parsing of references based on grammars (Parsing expression grammars) and the look-up of identifiers with internal and external registries of references. It runs as a web service.

University of Torino presented *SDFTreeMatcher*, a tool written in Java. It is based on rules (SDFRules) expressed in external XML files. It relies on POS tagging and Named Entity Recognition. It is able to connect to the Eur-Lex database in order to obtain ECLI identifiers. It runs as a web service.

CENDOJ relies on a stand-alone closed-source software. The software relies on structured inputs and it is based on rules in order to identify the fields of the citations. The look-up of the identifiers is done by connecting to official national registries available in Spain.

ITTIG presented *Prudence* and *Linkoln*. The software are written in Java and provided as Java libraries. They both rely on a a pipeline of components that perform entity identification, reference recognition and production of identifiers. They use JFlex in order to generate lexical scanners based on start conditions and macros of regular expression. The identifiers are generated automatically, without help from with external services or registries.

## Considerations

*Approach*

The most significant common trait among the tools presented by the partners involved in the BO-ECLI project is the rule-based approach. Generally, a citation is a very distinct information within a text, hence a rule-based approach for the identification of citations can be more effective and accurate than a machine learning approach. Each existing tool is based on rules, with different implementations: *SPeLT-ref* and *SDFTreeMatcher* read regular expressions from external configuration files and use them on the fly, the *eXtendable Legal Link eXtractor* benefits from a pre-compiled parser expression grammar, *Prudence* and *Linkoln* use pre-compiled lexical scanners.

*Programming language*

Another quite common choice done by the partners of the project is about the programming language. Almost every presented tool rely on the Java environment and on XML and JSON for configuration files and external resources.

*Identifiers*

Every presented tool, after reference recognition, produces, or try to produce, an identifier for the reference in a standard format (generally ECLI and ELI). The specific approaches are different and country dependent. The Spanish tool connects to an official national web service that allows the look-up of the identifier from the metadata of the reference; other tools benefit from local or remote registries for resolving a reference to an identifier; still others completely rely on an automatic composition of the metadata of the reference to produce the identifier based on a predefined country dependent syntax.

*Aliases and titles*

Legislative and case-law citations can be explicit, typically a composition of date, number, type of document and so on, or implicit, like an alias or a title, a textual fragment used to implicitly refer to a specific document. In Italy the use of aliases in legislative and case-law texts is moderate. In other countries and for European case-law and legislation, citations through titles are much more common. The tools presented deal with this issue through the use of controlled vocabularies, populated either with the help of external services or completely by hand.

Recognizing titles and aliases means looking for patterns of multi terms in the text. This task can be hard and heavy from a computational point of view, especially when the titles and the aliases are many and long. In order to manage alias and title recognition, a controlled vocabulary is pre processed and represented as a *trie* structure by the *eXtendable Legal Link eXtractor*, or as a deterministic finite state automata by *Linkoln*. In SpeLT-ref frequent citations vocabularies are created based on the user experience and with the support of the University of Turin's tool for the Named-entity recognition task.

*I/O*

Concerning the input, the presented existing solutions mainly focus on the processing of plain text. With no assumptions about the structure of the input, a link extractor can be used with any kind of document or even fragment of texts.

In order to be flexible enough to be integrated and used in different contexts, the results of the link extraction process (references and identifiers) can be provided:

- as objects of an application programming interface;
- as XML elements of a generic XML stream;
- through specific mark-up of the original input text.

# 3. ANALYSIS OF USE CASES AND USER STORIES

All the partners and associate partners of Workstream 2 have been asked to provide, based on the know-how gained in their national experience, a document describing *use cases* and *user stories* about the main functionalities and usage scenarios for the link extraction tool.

Inputs provided by the partners and associate partners of the Workstream (CIRSFID University of Bologna, UBR|KOOP, ITTIG-CNR, Supreme Court of The Czech Republic) are hereby summarized.

## General requirements

From the analysis of the use cases and user stories documents follows a synthetic list of general requirements that are expected to be satisfied by the extraction tool.

The extraction tool:

- must be able to automatically identify national and European judicial and legislative references by citation attributes, alias, title or by citation written in a standard European identifier, from plain text or unstructured document formats;
- should present the results of the extraction (references and identifiers) as objects of an application programming interface or as XML elements of a generic XML stream;
- should present, as an additional output, the original text with an appropriate mark-up of the textual citations, possibly associated with an hyperlink;
- must be able to provide the possibility for execution through an application programming interface, for command line execution and for batch processing;
- must provide the configurability of the language and jurisdiction of the input text
- should provide the configurability of a default identifier for the extracted references;
- must provide a set of metadata about provenance along with the output in order to guarantee reproducibility and bug tracking.
- should allow the possibility to specify metadata of the input text or document;
- should provide the appropriate means for allowing supervision of the results of the extraction;
- must allow the possibility of connecting to external resources for the look-up of identifiers from references or for the identification of citations by aliases or titles.

# 4. THE EXTRACTION TOOL

The extraction tool that is going to be developed within this project is expected to provide the means to be extended by more and more member States. For this reason, the tool will be composed by a common piece of software (the *common platform*) and by multiple national specializations, each implementing the peculiarities of a specific language and jurisdiction (the *national implementations*).

The main target of the developed tools will be new implementers from member states/jurisdictions who do not have a legal link extraction solution in place, or who want to join the BO-ECLI open source toolkit *e.g.* to replace their proprietary solutions.

As an outcome of the development activity along the duration of the project, at least one national solution, the one covering legal links extraction from Italian case law, will be fully implemented. The (at least partial) national implementation for an additional jurisdiction, to be selected in the next phases of the workstream's activities, is desirable.

Based on the considerations emerged from the analysis of existing national solutions (see. sect. 2), the BO-ECLI link extraction toolkit will be developed within a Java environment. The specific architectural design and technological choices will be the subject of the next phase of the project and are beyond the scope of the current document.

In this section the responsibilities and the scope of the common platform and a national implementation are depicted in a general way. Also, the modalities of interaction and interfacing among common platform, national implementations and resources are described in this section. An abstract model of the main foreseen components, their separation into common and national specific tasks and of their high level interaction, is sketched as a basis for the next detailed architectural design.

## Common platform

The common platform:

- is a software architecture that realizes a generic pipeline for legislative and case-law link extraction;
- includes the knowledge of the legislative and case-law link extraction domain (e.g. metadata of the reference, court codes, document type, etc.);
- can be configured for different languages and jurisdictions;
- should implement the recognition of citations written in a standard European format (like ECLI);
- provides the means to connect to local or remote identifiers look-up registries or services;
- provides the means to connect to local or remote controlled vocabularies of European or national aliases or titles;

- should be as easy to integrate as possible in web services;
- should present a clear separation with the national implementations and with the language dependent resources by providing the appropriate object oriented protocols and interfaces;
- should provide metadata about provenance within the output in order to guarantee reproducibility and bug tracking.

## National implementation

A national implementation includes the implementation of the object oriented protocols and interfaces provided by the common platform in order to cover the peculiarities of a specific language or jurisdiction. A national implementer will be in principle free to choose how to realize the implementation of a specific language or jurisdiction dependent task as long as the protocols for interfacing with the common platform are followed.

For example, implementing the module for the identification of a language or jurisdiction dependent entity in a text (like an issuing authority) could be done either through generic regular expressions and macros, with more complex grammars or connecting to a controlled vocabulary. No matter how the entity identification task is performed, the module annotates the text and delivers it back following the protocols and interfaces provided by the common platform.

In the architectural design phase, the specific technology used for the developed national implementation will be chosen and will constitute the guiding model for additional future national implementations. The implementation of each module will be documented in the iterative development phase and will be reusable as a starting point for new national implementations.

## Integration and protocols

The communication between the common platform and a national implementation is based on sending and receiving the original input text, enriched with annotations, through the appropriate interfaces to and from the modules implementing the specific language or jurisdiction dependent tasks.

The annotations are language independent and follow a scheme that includes the appropriate tags for identifying the textual entities of a citation and the appropriate list of attributes of a reference.

The annotation scheme, consisting of a list of tags and attributes, is part of the domain knowledge, it resides within the common platform and it is shared among the national implementations.

## Resources

From the analysis of the existing tools for case-law and legal link extraction and from the know-how brought by the project partners and summed up in section 2, there are mainly two kind of external resources that can be exploited:

- registries of national and European references (or the metadata of references) associated with the corresponding identifiers in various format;
- lists of textual fragments representing aliases or titles (controlled vocabularies) associated with a reference (or the metadata of a reference).

Maintenance and update of such resources is outside the scope of the extraction tool. A separate web service could act as a unique point of access for the editing of the resources, while the extraction tool could remotely connect to it in order to load the resource, or to make a local copy.

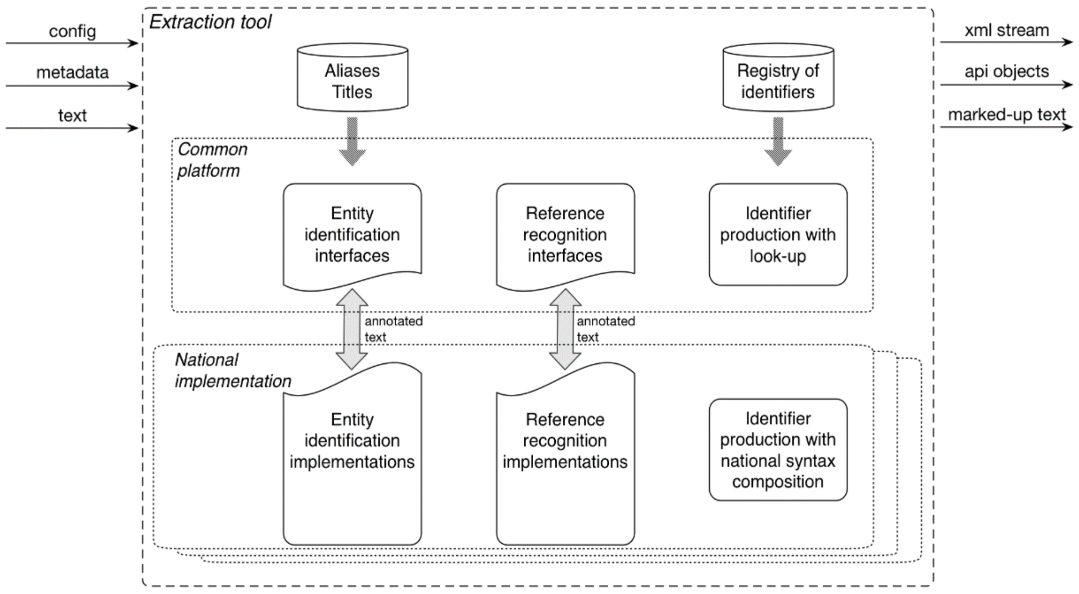
## Diagram

Hereinafter an abstract model of the main foreseen components, their separation into common and national specific tasks and of their high level interaction, is sketched as a basis for the next architectural design.

The model is kept deliberately generic and inclusive in order to leave room to decide, in the next phase of the design, on the proper architecture and on the implementational choices and modalities of interaction.

The following diagram depicts in an abstract way:

- the inputs and outputs of the overall extraction tool;
- the main phases of the extraction process;
- the interactions between the common platform and the resources;
- the interactions between the common platform and the national implementations.


config
metadata
text
Extraction tool
Aliases Titles
Registry of identifiers
xml stream
api objects
marked-up text
Common platform
Entity identification interfaces
Reference recognition interfaces
Identifier production with look-up
annotated text
annotated text
National implementation
Entity identification implementations
Reference recognition implementations
Identifier production with national syntax composition

Though in principle, according to this abstract model, each implementer will be free to choose how to realize the implementation of a specific language or jurisdiction dependent task as long as the protocols for interfacing with the common platform are followed, in the next phase of the project the specific technology used for the national implementations developed within BO-ECLI and recommended for new implementers will be chosen. In such a context thus, the different national implementations sketched in the diagram will only differ for the language and jurisdiction peculiarities and not for the implementation technology.

## ANNEX 1 – National solution fact sheets

**National Solution Fact Sheet**

**SPeLT**

***CIRSFID – University of Bologna***

### 1. GENERAL INFORMATION

CIRSFID has developed SPeLT(Semantic Parser of Legal Text), a framework of tools for parsing of legal texts. Currently, SPeLT supplies two main tools for parsing; SPeLT-ref and -ref and SPELT-struc.

Also, SPELT supplies a set of rules, called SPELT-anony, that can be used by third-parties software in order to anonymize specific entities in judgements and other legal documents.

Lastly, SPELT supplies a markup tool, called SPeLT-mark, that can use the results of a parser to markup an unstructured document with AkomaNtoso. To accomplish this, SPELT-mark can use results returned by SPELT-struc and SPELT-ref, but it can also use results returned by third-parties software.

#### 1.1. SPELT-ref

The aim of SPELT-ref is to identify legal references in judgments and other legal documents. It is intended to work with unstructured document, but it is also able to identify legal references in partially structured documents or in fully structured documents. SPELT-ref, in its latest version, can extract legal references from documents written in Italian, English, Spanish and Romanian, and it can extract references from different types of legal documents like judgments, bills, acts and reports.

It was originally commissioned by the Uruguayan “Cámara de Representants” for Uruguayan bills and acts.

#### 1.2. SPELT-struc

The aim of SPELT-struc is to identify the logical structure of legal documents, so SPELT-struc is meant to identify the parts of legal documents that have legal semantic meaning and to identify relations among these parts. The set of extracted fragments can be used by markup software in order to markup the document by means of legal XML standards, like AkomaNtoso.

SPELT-struc, in its latest version, can identify structure in documents written in Italian, English, Spanish and it can identify structure in different types of legal documents like judgments, bills, acts and reports.
It was originally commissioned by the Uruguayan "Cámara de Representants" for Uruguayan bills and acts.

### 1.3. SPELT-mark

SPELT-mark has the aim to markup unstructured documents with AkomaNtoso. It gets in input a set of offsets, resulted by a parsing process, and an unstructured legal documents. It searches for the offsets in the documents, and surrounds offsets with the most proper AkomaNtoso tag, according to a set of configurations' files.

It is independent from language and can markup all documents' types defined by AkomaNtoso.

It was originally commissioned by the Uruguayan "Cámara de Representants" for Uruguayan bills and acts.

### 1.4. SPELT-anony

The aim of SPELT-anony is to anonimize the judgment on the base of the University of Turin module, that detects the NER (Named-entity recognition), and the legal rules expressed in logic. The anonymization is made on the Akoma Ntoso XML file in order to not affect the original manifestation, but in meantime to produce an additional XML manifestation without the sensitive and personal data involved in the judgment.

## 2. GENERAL APPROACH

SPELT's tools are used sub sequentially in order to create a structured and marked-up document by starting by an unstructured legal document.

SPELT-ref and SPELT-struc rely on regular expressions and configurations' files. Regular expressions are created on-the-fly and applied to text in a specific order according to configuration files. Configurations files are built according to legal texts' languages, to legislative rules of specific countries, districts and offices or institutions.

SPELT-mark relies on configurations' files and on rules deduced by AkomaNtoso schema. By starting by an offset, it understands what are the AkomaNtoso elements that can be used to markup the offset and, if there are more than one, it discriminates in favor of the most proper by means of configurations' files. Configurations' files are created according to legal texts' languages, to legislative rules of specific countries, districts and offices or institutions.

SPELT-anony is used in combination of any NER.

## 3. PROGRAMMING LANGUAGES

SPELT-ref and SPELT-struc are developed in PHP, and there are a node.js and a python porting in progress.

SPELT-mark and SPELT-anony are developed in client-side javascript.

## 4. ARCHITECTURE / FRAMEWORK

All the tools currently supplied by SPELT are developed with standard javascript, node.js and PHP libraries.

## 5. KIND OF RULES / GRAMMAR

SPELT-ref and SPELT-struc use generic regular expressions. Regular expressions are defined in JSON configurations files according to specific use cases, such as legal texts' language, countries' legislative rules and so on. SPELT-ref and SPELT-struct extract the proper regular expressions by JSON configurations' files and apply them to the text.

SPELT-anony is besed on rules using for now a simple grammar if-than with regular expression. In future we intend to use LegalRuleML for expressing more complex rules and in order to foster some legal reasoning engine (SPINDle).

## 6. MODULARITY / CONFIGURABILITY / EXTENSIBILITY

As discussed in previous sections, SPELT is intended to be a framework that supplies parsing tools. For this reason, it is extendable with more tools for parsing or with tools related to other parsing tasks.

SPELT, SPELT-struc, SPELT-ref, SPELT-mark and SPELT-anony rely on JSON configurations files. The set of JSON configuration file can be extended, restricted and modified. Also,pre-written JSON configurations' files supplied by SPELT's tools, can be modified to be adapted to specific use cases.

## 7. DEPLOY

SPELT and all SPELT's tools are released as web-services. They are intended to be used on the web but can by accessed by desktop software by means of REST APIs.

## 8. DEPENDENCY OF EXTERNAL PROCESSING

The software is not dependent on any NLP or XML framework and the input doesn't require any external preprocessing.

## 9. SOURCE CODE AVAILABILITY

Open source for non-commercial use. Modified MIT license. For the commercial use it is necessary an agreement with University of Bologna – CIRSFID.

## 10. INTENDED USERS

Since the software is deployed as a REST WEB services it is intended to be used by other software.

With a minimal effort, it also can be used as PHP or javascript library, so it can be used by developers that have to build software related to legal context as well as software related to other contexts.

## 11. USERS' FEEDBACK MECHANISM

Currently no automatic feedback or formal bug-report system is supported.

## 12. LINK CONSTRUCTION AND VALIDATION MECHANISM

SPELT-ref currently supports the followings references' styles:

- AkomaNtoso
- URN:NIR/URN:LEX
- CELEX
- ECLI
- ELI

## 12. STAGE OF DEVELOPMENT

First release of SPELT-ref, SPELT-struc and SPELT-mark were published on January 2013. First release of SPELT-anony was released on January 2016.

## 13. COVERAGE

### 13.1. Language / Jurisdiction

SPELT-ref and SPELT-struc support Italian, English, Spanish and partially Romanian. The following documents are supported:

- Italian bills, acts and other legal regional documents;

- Uruguayan bills, acts and other legal documents;
- US act;
- FAO codex standards and basic text.

Partial support for the extraction of judicial references to European case law is also provided.

Extraction of references to European legislation is planned to be released by mid 2016 but not available yet.

### 13.2. references type

Both legislative and judicial citation are covered. The software offers a wide coverage on both legislative and judicial issuing authorities and legacy citations. Besides the most common citation styles, many forms of less common and legacy citations are covered by the parsers.

### 13.3. document types

The software supports all the legal documents types supported by the AkomaNtoso standard.

### 13.4. document formats

The input is plain text that can came in input in different formats; txt, rtf, doc, docx, odf, pdf, html.

### 13.5. coverage limits

Currently SPELT-ref supports all the references types; single references, multiple references and ranges of references.

### 13.6. recognition by titles/aliases

Recognition by title aliases is supplied in Italian, English (UK) and Spanish (UY). It is handled by means of JSON vocabulary of common titles and aliases.

## 14. REUSABLE KNOWLEDGE

### 14.1. reusable rules

All rules for Italian, English and Spanish can be reused for partial recognition of references in other languages that are latin alphabet-based.

### 14.2. reusable (external) resources

SPELT and SPELT’s tools do not use external resources.

### 14.3. reusable code

The code is available on request after the check of the nature of the purpose and of the body that made the request.

### 14.4. reusable analysis/know how

The documentation is available on line in http://lime.cirsfid.unibo.it and in several papers.

## 15. MORE FEATURES

### 15.1 Additional information

Forthcoming (March 2016): technical documentation on software and APIs will be released in March 2016.

## National Solution Fact Sheet

***National solution full name***

***Responsible organization: CENDOJ***

### GENERAL INFORMATION

**general approach** (e.g. rule based / statistical)

Rule based

**programming language**

**architecture / framework**

**kind of rules / grammar**

**modularity / configurability / extensibility**

**deploy** desktop / web

**dependency on external processing**

structural xml markup

**source code availability** (incl. License)

No

**intended users**

**user's feedback mechanism** (y/n, how)

**link construction and validation mechanism** (e.g. also through reference registries look-up, supported "serialization" formats ELI, ECLI, CELEX, akn, urn:lex, national,..)

**Several mechanism:**

1. Find pre-defined formats: ECLI, ELI, CELEX, ROJ (NATIONAL), NBOE

(NATIONAL LAW IDENTIFIER)

2. Text analysis.

   For wich use:

- Dictionaries of words that can appear in citations: courts, locations,…
- Dictionaries of words that can appear in Spanish and European

  legislative citations:
    - By its scope .
    - By its name.
- Patterns for dates, sentence number, appeal number,numbers of articles, sections, headings, paragraphs, clauses, etc.…
- Semantic rules with terminology used in citations.

2.1. For Case law citations

Normalization.

The text is evaluated and the following information is extracted from it:

Court

Community/Province/City

Type of decision

Appeal number and/or decision number

Decision date

Trying to build a citation :

- Key (compulsory)

  • Acronym court (TC, TS,..) o extended(Constitutional Court…)

  • Acronym Type of decision (S, A,..) o extended(Judgement Auto (Order))

  • Mixed. Court+ type of decision (STS, ATC,…)

- Reference (compulsory)

  It can be of two types, not excluding

        • Decision Date

        • Number/Year

    - Extra data(optional)

        • Non-mandatory information that sometimes appears: Court Chamber, section,…

Linking.

A search is performed in the database, in the metadata :

It is not found in the repository (No link)

Only one is found (1 link)

There are several that meet the requirements (Multiple links)

2.2. For Case legislative citations .

Normalization.

The text is evaluated and the following information is extracted from it:

- Legal Standard:
    - If it exists in the list of standards, acronym is included (LECRIM, LEC,…)
    - If it doesn't exist:
        - Type of legal standard(law, decree, royal decree,.
        - Date
        - Nº/nº o Nº de nº
        - Name
- Number (including paragraphs, sections, numbers,..)

Trying to build a citation wich has 3 parts :

– Article

• It is the first part of a citation

• Articles, Legal Provisions(additional, Exemption, final)

• Bis, ter, quarter,…

– Legal Standard. They can be of two types:

• Fixed. Standards with commonly used acronyms that are included in lists ..(CE, LECRIM,..)

• Variables. They come structured by the following components::

– Legal Standard: Law, consolidated text, decree, etc.F

– Date

– Nº/nº o Nº de nº

– Name

– Complement.

• They are numbers or letters that come with the article. (number, paragraph, section (inciso) or rule (regla)).

Linking.

- A search of the Legal Standard acronyms is performed in the database, in the metadata (link)
-

**stage of development** (research / beta / released)

Released

**COVERAGE**

**language / jurisdiction**

spanish /civil, criminal, administrative, labor, constitutional, military

**references type** (e.g. legislative / judicial)

legislative/judicial

**document types** (legislation, case law .. )

legislation, case law

**document formats** (txt, html, XML..)

XML

**coverage limits** (e.g. multiple references, incomplete citations, internal alias?)

**recognition by titles/aliases** (y/n, how)

YES. using dictionaries and sinonimous

**REUSABLE KNOWLEDGE**

It 's propietary.

**reusable rules** (are there reusable/portable set of recognition rules? In which format?)

**reusable (external) resources** (e.g. catalogs, code lists, reference repositories: description, list, format, reusable analysis)

**reusable code** (is the code available and reusable?)

**reusable analysis/know how** (e.g. national citation rules and practices, EU sources citations analysis)

**MORE FEATURES**

**...**

**ADDITIONAL INFORMATION** (e.g. technical documentation, other)

…

## National Solution Fact Sheet

***National solution full name: SDFTreeMatcher***

***Responsible organization: UNITO***

### GENERAL INFORMATION

**general approach:** rule-based

**programming language:** the SDFTreeMatcher package is a software developed Java and XML. XML is used to write the rules that tag the words in the sentence, the Java software execute the rules and perform post-operations, e.g. named entity recognition on legal text, on the basic of the associations word->tag.

**architecture / framework:** Java and XML

**kind of rules / grammar:** ad-hoc simple XML grammar (the grammar includes six main XML tags only).

**modularity / configurability / extensibility:** highly modular, easy to extend for recognizing other kinds of named entities. Easy to interface with statistical approaches for populating the XML grammar (XML rules and Java code are independent. The Java software simply executes the XML rules that may be externally produced via statistical techniques).

**deploy:** both desktop or web, via Java webservices.

**dependency on external processing:** the grammar act on POSTagger analysis, in any POSTagging format associate word with the basic (standard) features (POS, lemma, gender, number, etc.). The SDFTreeMatcher system is language-independent, and it can be quickly adapted to any POSTagging format, e.g., the one of the Stanford parser (http://nlp.stanford.edu/software/lex-parser.shtml).

**source code availability:** the code may be released under the MIT license.

**intended users:** the SDFTreeMatcher is general enough to be used for any rule based task on natural language text.

**user's feedback mechanism** -

**link construction and validation mechanism** (e.g. also through reference registries look-up, supported “serialization” formats ELI, ECLI, CELEX, akn, urn:lex, national,..)

**stage of development** research

## COVERAGE

**language / jurisdiction:** references to EU legislations in Italian and English, references to Italian legislation and case law.

**references type** legislative and judicial

**document types** legislation and case law

**document formats** Input format: txt or XML. Output format Akoma ‘Ntoso.

**coverage limits** multiple references can be handled. Rules to cover incomplete citations and internal alias have not been written yet.

**recognition by titles/aliases -**

## REUSABLE KNOWLEDGE

**reusable rules** it is easy to convert SDFRules from/to other pattern-matching rule formats.

**reusable (external) resources** no

**reusable code** the code may be released under the MIT license.

**reusable analysis/know how** no relevant know-how on legal reference analysis.

## MORE FEATURES -

## ADDITIONAL INFORMATION -

This project is co-funded by the European Union

# National Solution Fact Sheet

## *Prudence, Linkoln*

## *ITTIG-CNR*

## GENERAL INFORMATION

ITTIG developed two separate software for link extraction: Prudence and Linkoln.

*Prudence* was originally developed for judicial reference extraction from national case law texts, commissioned by the Italian Ministry of Justice and specifically used by the Supreme Court of Cassation and by the Civil Tribunal of Milano to process their case law.

*Linkoln* is stilll under development and its open source release is planned by mid 2016. Commissioned by the Italian Senate for the automatic legislative reference extraction from any kind of normative document: national, regional, primary and secondary legislation.

**general approach** (e.g. rule based / statistical)

Both Prudence and Linkoln, two software developed by ITTIG for the automatic extraction of judicial and legislative citations in Italian texts, rely on rules.

The extraction process is performed in three steps: entities identification of each element that compose a citation, recognition of a reference in a particular pattern of entities and, finally, the serialization of a reference to an identifier in a specified convention, like ECLI for judicial references and urn:lex for legislative references.

**programming language**

The environment is Java.

**architecture / framework**

The software was developed as a Java library without reference to any particular framework.

**kind of rules / grammar**

In both softwares we made use of Jflex, a free tool for developing efficient lexical scanners in Java. Jflex is based on standard regular expressions, macros and start conditions; other features of Jflex are the lookahead and pushback capabilities and the possibility to import generic lists of macros for reuse. Jflex is used to compile the jflex file containing the rules and the start conditions to a java file that represents a fast lexical scanner implementing the rules. Rules update requires recompilation to take effect.

**modularity / configurability / extensibility**

Our software is composed by many modules responsible for:

- the entity identification of each attribute that potentially belongs to a textual citation;
- establishing if a particular pattern of entities forms a reference;
- transforming a reference in an identifier in a specific convention.

During the parsing, the text is internally annotated with a temporary mark-up according to an internal metadata model. Each module can contribute to such annotation.

The extraction process is highly configurable: modules can be turned off, patterns can be filtered in order to extract references of specific types or from specific issuing authorities, the recognition of incomplete citations can be enabled, different conventions for serialization of the reference can be specified, etc.

Adding new modules is also straightforward, for instance a module for the identification of a new entity, or a module for the recognition of a new kind of pattern of reference, or a module that implements the serialization of references to a convention not supported before.

**deploy** desktop / web

Our software is released as Java libraries. The libraries can be deployed either within a standalone application or a web application. A web application for testing and demonstrative purposes was also developed.

**dependency on external processing** (e.g. linguistic processing stack, structural XML markup)

The software is not dependent on any NLP or XML framework and the input doesn't require any external preprocessing.

**source code availability** (incl. License)

Copyright ITTIG, license not specified.

**intended users**

Since the software is released as Java libraries, the immediate users will be system integrators and developers. The libraries can be easily installed and integrated either on a web application or a standalone application, in order to serve purposes like: analyzing a single document, performing a massive extraction from a corpus of documents or from a database, validating citations and produce the correct identifier in a drafting environment, etc.

**user's feedback mechanism** (y/n, how)

Currently no automatic feedback or formal bug-report system is supported.

**link construction and validation mechanism** (e.g. also through reference registries look-up, supported “serialization” formats ELI, ECLI, CELEX, akn, urn:lex, national,..)

We currently support urn:lex for Italian legislation and ECLI for judicial citations. Support for the Italian implementation of ELI is foreseen in 2016.

The link construction process is based on the internal built-in knowledge of the syntax of the italian implementation of standard legal identifiers (Italian ECLI and urn:lex).

No automatic validation mechanism is in place since no authoritative open registries of legal identifiers are available in Italy so far.

**stage of development** (research / beta / released)

Prudence was released on June 2015, Linkoln is currently in beta stage and planned to be released jointly with the Italian Senate by mid 2016.

## COVERAGE

**language / jurisdiction**

Our software support the Italian language and Italian case law. Partial support for the extraction of judicial references to European case law is also provided.

Extraction of references to European legislation is planned to be released by mid 2016 but not available yet.

**references type** (e.g. legislative / judicial)

Both legislative and judicial citation are covered. The software offers a wide coverage on both legislative and judicial issuing authorities and legacy citations. Besides the most common citation styles, many forms of less common and legacy citations are covered by the parsers.

**document types** (legislation, case law .. )

National case law (intensive testing on first instance civil case law, constitutional case law, Supreme Court of Cassation civil and criminal case law)

Any type of legislative document (national, regional, primary and secondary legislation).

**document formats** (txt, html, XML..)

The input is plain text (txt, doc, pdf, html tags are ignored).

**coverage limits** (e.g. multiple references, incomplete citations, internal alias?)

Our software supports the extraction of multiple citations;

incomplete citations are identified and then, depending on the configuration of the software, presented as incomplete references or rejected;

currently, internal citations are identified, but no identifier is produced;

Support for internal aliases is foreseen in the final release of Linkoln.

**recognition by titles/aliases** (y/n, how)

Yes. Specialized modules were developed for the recognition of the main italian legislative aliases, like common names for codes (*codici)* and unified texts (*testi unici)*. Matching with titles is performed as well through Jflex scanners. Jflex rules are automatically generated from catalog items and revised manually.

**REUSABLE KNOWLEDGE**

**reusable rules** (are there reusable/portable set of recognition rules? In which format?)

The rules used for the identification of each attribute that compose an Italian legislative or judicial citation can be reused for the Italian implementation.

Implementations for countries besides Italy can partially benefit from the rules used for the identification of generic information like dates and numbers.

**reusable (external) resources** (e.g. catalogs, code lists, reference repositories: description, list, format, reusable analysis)

Our software makes use of a catalogue of judicial issuing authorities, a catalogue of legislative issuing authorities, a list of type of cited documents (both legislative and judicial) and a catalogue of legislative aliases. These catalogues have been specifically filled in for the purposes of the reference recognition software and are used internally. No publication as open data on the web for third parties reuse have been released so far.

All this information can be exploited in the “BO-ECLI linking platform” national Italian implementation.

**reusable code** (is the code available and reusable?)

The code is completely reusable without any technical or legal restriction. Modules can be entirely or partially reused in the Italian implementation.

**reusable analysis/know how** (e.g. national citation rules and practices, EU sources citations analysis)

We recently collaborated with the civil section of the Court of Milan, the Supreme Court of Cassation (criminal and civil sections), the Constitutional Court and the Senate of the Republic, gathering plenty of experience in the field of legal and judicial citation in Italy. All this experience and know-how is at disposal of the BO-ECLI project.

**MORE FEATURES**

**ADDITIONAL INFORMATION** (e.g. technical documentation, other)

documented Java API (in Italian) for Prudence (judicial references extractor)

**Forthcoming (beginning 2016): documented Java API and commented source code for Linkoln (legislative references extractor) hosted on github.**

## National Solution Fact Sheet

## eXtendable Legal Link eXtractor

## Publications Office of the Netherlands (UBR|KOOP)

### GENERAL INFORMATION

**general approach**

The main components of the eXtendable Legal Link eXtractor ("link extractor) are:

- *Named Entity Recognition* (NER) based on large (~ 200k) entity collections;
- Reference parsing based on *Parsing Expression Grammars* (PEG);
- Link resolution based on internal and external reference collections.

This is a rule-based approach, although the architecture can integrate statistical and language-model based approaches.

Internally, the link extractor only uses XML. No internal object-model is built. Thus the original document structure is preserved.

**programming language**

The link extractor uses different programming language at different levels, ranging from low-level, detailed to high-level, abstract.

| **programming language** | **usage** |
|---|---|
| Parsing Expression Grammar<br><br>Apache Cocoon sitemap XML | Reference language definition<br><br>Pipeline configuration |
| XSLT | Minor pipeline components (local aliases, disambiguation, canonicalization, link resolution) |
| Java<br><br>Scheme (Racket) | Major pipeline components (NER, parser)<br><br>PEG parser generator (Waxeye open source project) |

Note that we use XSLT as a functional programming language that is particularly suitable to transform XML documents; in the user interface code it is also used to generate HTML, but that is not its main purpose.

The run-time environment is a Java 7 Virtual Machine (JVM).

**architecture / framework**

The link extractor is set up according to a *pipe-and-filter* (pipeline) architecture. The pipeline combines processing components in a way that can be easily adapted, parameterized and extended in a simple XML configuration file.

The particular pipeline implementation is Apache Cocoon (2.1), which is a framework for building 'streaming' XML pipelines. Internally, Cocoon uses SAX event streams, which obviates the need for repeated XML serialization and parsing (deserialization).

**kind of rules / grammar**

The main parser uses Parsing Expression Grammars (PEG). A PEG is in some ways like a modularized regular expression. Because the processing model of PEG is similar to regular expressions, it is easy to convert a complex regular expression to a PEG.

PEG has several advantages over regular expressions. The most obvious is the modularization of complex expressions into multiple *non-terminal symbols* and grammar modules. This is similar to procedures and classes in programming languages, and makes a grammar readable and maintainable.
The output of a PEG parser is a parse tree, which can be easily converted into an XML structure. This integrates nicely with the XML-based pipeline architecture. Another advantage is that the PEG model allows for more efficient implementation than complex (synthesized) regular expressions.

Because PEG is a well-understood parsing mechanism, there are many good implementations available. We have chosen to use the open source Waxeye implementation because it is efficient and easy to integrate in our pipeline.

For Named Entity Recognition (NER) we use simple dictionaries. The NER parser can be parameterized to deal with case-insensitivity, ignored characters and word boundaries.

**modularity / configurability / extensibility**

All processing is configured in Cocoon pipelines, which consist of a sequence (actually a directed acyclic graph) of parameterized components. The components can be selected according to input parameters that specify the input type and desired output. As an example, consider the detection of local aliases, which is specified as follows in the pipeline:

```
<!-- Select / parameter / when is similar to if-statement. -->
<map:select type="extractions">
<!-- Extract parameter specifies which references to recognize. -->
<map:parameter name="parameter" value="{extract}"/>
<map:when test="eu/nl-regulations">
<!-- Parse document to find local alias candidates. -->
<map:transform type="waxeye" src="grammars/le-local-alias.waxeye">
<map:parameter name="namespaceURI" value="http://linkeddata.overheid.nl/lx/"/>
<map:parameter name="parseElementTag" value="parse"/>
<map:parameter name="modular" value="true"/>
</map:transform>
<!-- Local aliases get priority over global aliases. -->
<map:transform src="xsl/local-alias-before-global-alias.xslt"/>
<!-- Detect and mark local aliases. -->
<map:transform src="xsl/detect-local-alias.xslt"/>
</map:when>
</map:select>
```

Depending on algorithmic complexity and performance requirements, pipeline components are written in Java (according to a simple interface) or XSLT.

In the above pipeline fragment, the 'le-local-alias.waxeye' grammar is a modular grammar, composed of several reusable sub-grammars. For example, it includes the 'document.waxeye' grammar, which describes general XML document structure.

The core link extractor is extended to cover different types of input and output documents, by embedding it in pre- and post-processing code, as follows:

```
<!-- Complete pipeline to process links in a plain text document. -->
<map:match pattern="extract/**.txt">
<map:generate type="text" src="{1}.txt"/><!-- Load text document from any URL. -->
<map:transform type="expand-ligatures"/><!-- Java component -->
<map:transform src="xsl/input-formats/prepare-text.xslt"/><!-- XSLT component -->
<map:call resource="extract-links"><!-- core link extractor sub-pipeline -->
<map:parameter name="extract" value="{request-param:extract}"/>
</map:call>
<map:serialize type="xml"/><!-- Original text with XML markup for references. -->
</map:match>
```

The link extractor can be configured and extended at the pipeline level and at the grammar level. Both require only limited technical knowledge; Understanding *what* each component does is far more important than *how* it works. In some cases, new Java or XSLT components have to be made, or existing components can be modified.

It is easy to extend the dictionaries used for named entity recognition, which are simply tab-separated value (TSV) files. Within the project we have built a web-based user interface to edit large TSV files.

**deploy: desktop / web**

The link extractor is deployed in a web server, either locally on a desktop or laptop, or on a publicly accessible server. The link extractor is accessed through HTTP and its operations are addressed by URLs.

The link extractor offers a number of web services (APIs) that can be called from other applications, as well as a browser-based user interface. Both interfaces support different types of documents and can be parameterized to produce different types of output.

Every deployment has its own configuration in the form of two *property files*. These files specify things like system paths, logging levels and external server URLs (for link resolution).

A new deployment can be set up in 30 minutes, including downloading and installing third-party software.

**dependency on external processing**

The link extractor depends on external services for reference resolution. These services are based on repositories to find officially designated identifiers like ECLI for references that are found by the link extractor. For example, when the link extractor finds a literature reference "BNB 1956/56, the external service will resolve this to "ECLI:NL:HR:1955:AY1977". The resolved ECLI will be included in the output of the link extractor.

Internally, the link extractor depends on an XML representation of the document. There is a wide variety of preprocessing (generator) and post-processing (serializer) components available for Apache Cocoon. This includes formats such as XML, HTML, plain text, MS Word and PDF. Cocoon supports a number of interfaces and protocols to access web services including ReST, SOAP, JDBC, XQuery and SPARQL.

**source code availability (incl. license)**

Much of the code comes from other open source projects that use the permissive Apache or MIT licenses. This includes third party projects as well as our own. In principle the source code developed by UBR|KOOP is available too.

**intended users**

The intention of the current software is to extract references for public documents, especially judicial decisions, parliamentary documents and official publications. The references will be used to populate a linked data repository within the Dutch government, which will be used for various purposes.

**user's feedback mechanism (y/n, how)**

Currently there is no user’s feedback mechanism, but it is on the wishlist.

**link construction and validation mechanism**

Link construction is a component that comes after the reference recognition (parsing) component. Depending on the type of link and the required linking standard, a reference registry (local or remote) is used. Depending on the results obtained from the reference registries, links are created or marked as potentially invalid or ambiguous (resolve-manual).

At the moment, the link extractor supports links using juriconnect (Netherlands standard for references to law), OEP (Netherlands method for references to official publications and Parliamentary documents), CELEX and ECLI.

**stage of development (research / beta / released)**

Currently we are in the third iteration, which will soon release a version that should be ready for production.

**COVERAGE**

**language / jurisdiction**

Dutch (depends on grammar) / Netherlands, Europe

**references type (e.g. legislative / judicial)**

The link extractor recognizes links to national and European legislation, national and EU judicial decisions, official publications (national and local), and parliamentary documents.

**document types (legislation, case law .. )**

Case law, official publications, parliamentary documents.

**document formats**

The link extractor is currently used to recognize and mark references in XML, HTML and text documents. Processing of MS Word and PDF documents within the Cocoon framework has been used in several other projects and could be added to the link extractor. This will require the use of a commercial component (Aspose.Words).

**coverage limits (e.g. multiple references, incomplete citations, internal alias?)**

Work is going on to recognize incomplete citations, using the document's origin.

Work is going on to correctly link multiple references in the form of series like "article 6, 8 and 10-15".

Internal (local) aliases are recognized. If there is a (global) alias with the same designation, the local alias has priority.

Incorrect or ambiguous citations are marked as such, so they can be processed manually.

**recognition by titles/aliases**

Titles and aliases and other fixed designations are inefficient to recognize using a grammar (or regular expression). Therefore we use a trie-based Named Entity Recognition (NER). A trie is a memory-efficient data structure with fast retrieval.

When the link extractor server is started, the trie is constructed from a dictionary that maps named entities (phrases) to identifiers. The trie is kept in memory during the lifetime of the server process, so lookups are very fast.

The NER uses several sources:

- BWB (Netherlands consolidated legislation repository);
- CELEX (via UBR|KOOP's own linked data service);
- Lists of additional (unofficial but often used) aliases for BWB and CELEX;
- Obsolete titles (which occur in older documents);
- A list of global aliases for judicial decisions;
- A list of exceptions which are removed from the dictionary.

Like all parts of the link extractor, this can be configured in the Cocoon pipeline.

The BWB and CELEX lists are updated externally, and 'pollution' sometimes occurs (with titles like 'Decision'). In order to deal with this, we made a user interface for content managers, where they can retrieve a list of new or conflicting terms, and mark terms that will be added to the exception list.

**REUSABLE KNOWLEDGE**

**reusable rules**

Reference recognition is done based on Parsing Expression Grammars (PEG) using Waxeye's notation. These grammars are modular, and are reused internally as much as possible.

The top-level modular grammars are:

| | |
|---|---|
| le-eu-regelgeving.waxeye | Modular grammar for European legislation |
| le-jurpointer.waxeye | Modular grammar for composite references to jurisprudence |
| le-links.waxeye | Modular grammar for references to Netherlands law, simple references to jurisprudence, official publications |
| le-lokale-alias.waxeye | Modular grammar for local aliases |
| links.waxeye | Starting point for all types of links |

These modular grammars include and use the following sub-grammars:

| | |
|---|---|
| eu-regelgeving.waxeye | References to European legislation |
| jur-verwijzing.waxeye | Starting point for all references to jurisprudence |
| jur-citatie-ecli.waxeye | References using ECLI |
| jur-citatie-ljn.waxeye | References using LJN (Dutch predecessor to ECLI) |
| jur-citatie-vindplaats.waxeye | References to legal literature |
| jur-citatie-datum.waxeye | Composite references: date |
| jur-citatie-instantie.waxeye | Composite references: court |
| jur-citatie-zaaknummer.waxeye | Composite references: case number |
| jurpointer.waxeye | Composite references |
| oep-verwijzing.waxeye | Starting point for references to official publications |
| oep-oeb-citatie.waxeye | References to official announcements |
| oep-parl-docs.waxeye | References to parliamentary documents |
| wr-verwijzing.waxeye | References to laws and regulations |
| wr-lokale-alias.waxeye | Local aliases |
| document.waxeye | General document structure (included by all modular grammars) |

The exact rules reflect national practices and standards for representing legal references, and contain specific Dutch words and phrases.

This project is co-funded by the European Union

**reusable (external) resources (e.g. catalogs, code lists, reference repositories: description, list, format, reusable analysis)**

The link extractor was made in close collaboration with the Governmental Linked Data (LiDO) project, which contains a large collection of metadata and links. LiDO is set up as a reusable repository and semantic web application, providing ReST web services that produce RDF documents.

The link extractor was designed to work without any database, and does not include any repositories, except some lists stored in TSV format. These lists contain aliases and exceptions, and are reusable.

**reusable code (is the code available and reusable?)**

The link extractor reuses a lot of code that is available in other (open source) projects. The most important external projects are

- Apache Cocoon (which reuses other projects itself)
- Cocoon Components
- Waxeye

The code of the link extractor will be made available. Much of it is easily reusable:

- The pipelines are language independent and can be reused (with modifications) in other contexts.
- The PEG grammars are language-dependent, but can serve as a template for other languages.
- The link extractor contains more than 60 XSLT stylesheets for tasks ranging from simple (adding named entities to a list; preparing SPARQL queries for LiDO) to intermediate (authentication and authorization; construction of links after parsing) to complex (splitting composite references that share some designations). These can be reused except for the complex and highly specific stylesheets.
- The link extractor contains a few Java components, all of which are reusable. Some are very simple, for example ligature expansion in text. Others are more complex, like the component that generates a parser from a PEG, compiles the parser, caches it for future use, and applies it to a streamed XML document.

One of the advantages of the pipeline architecture is that components tend to be simple and focused. Large monolithic software 'masterpieces' are discouraged by this architecture.

**reusable analysis/know how (e.g. national citation rules and practices, EU sources citations analysis)**

The link extractor is based on Marc van Opijnen's thesis "On and In the Web - how the accessibility of Judicial decisions Can be Improved". This is a comprehensive treatise on citation rules and practices in the Netherlands, including the legal framework for publishing legal documents. It is reusable, by those who can read Dutch.

The link extractor project was initiated by UBR|KOOP to create production-ready software that embodies the analysis and knowledge that is present in Marc's thesis.

## MORE FEATURES

### unit and regression tests

The link extractor includes a set of 200 unit tests that contain almost 1000 references. For every new or changed feature, new unit tests are defined first (test-driven development). Before a new version of the software is released, all unit tests must succeed (regression testing).

### security

The administrative user interface of the link extractor is only accessible for authenticated and authorized users. This is based on Cocoon's authorization module and currently uses IP-addresses (for internal server-to-server communication) and username/password (for humans).

The link extractor processes XML documents from external sources. Special effort has been made to protect the software from known XML attacks.[1]

## ADDITIONAL INFORMATION

### documentation

The link extractor is documented by many design documents, and technical documentation including:

- Importing external sources (LiDO)
- Installing the link extractor and its dependencies
- Additional installation instructions for external service providers
- Maintaining grammars
- Interfacing with LiDO
- Unit tests

- Link extractor services (ReST API definitions)
- Specification of alias lists
- Description of pipelines and components

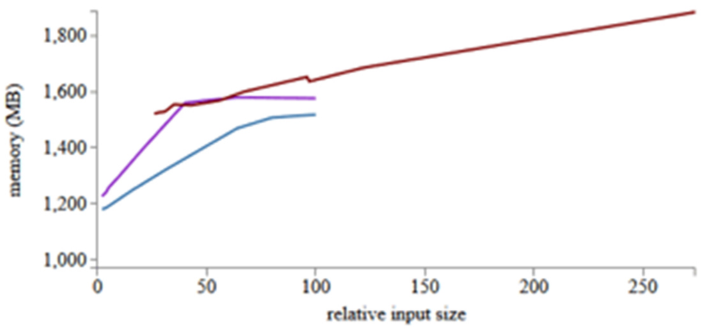
The user interface includes extensive 'help' documentation for end-users and administrators.

The documentation is available in Dutch only.

**performance**

The link extractor processes most documents within 1 to 20 seconds, depending on the size of the document and the hardware.

Memory usage is mainly determined by the cached tries and grammars, and by the size of the document that is being processed. The following graph shows the amount of memory used for documents ranging from very small to rather large (several megabytes).



This shows that for most applications, 2 GB of memory should be enough. Of course, the web server and operating system also need some memory. In practice, the link extractor can easily run on a 3 year old laptop with 4GB of RAM.

[1] During a presentation by Christian Mainka and Christopher Späth (Ruhr University of Bochum) at XML Amsterdam 2015, we were able to successfully apply most of the attacks they described on the link extractor. With their help we could protect the link extractor against these attacks.