DELIVERABLE

Project Acronym: EUCases
Grant Agreement number: 611760
Project Title: European and National Legislation and Case Law Linked in Open Data Stack

D3.1 NLP Toolkit Report

Authors:

- Florian Schmedding  AVERBIS
- Kornél Markó  AVERBIS
- Kiril Simov  IICT-BAS
- Livio Robaldo  UNITO
Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organisation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>13-06-2014</td>
<td>Florian Schmedding</td>
<td>AVERBIS</td>
<td>First draft</td>
</tr>
<tr>
<td>0.2</td>
<td>11-07-2014</td>
<td>Florian Schmedding</td>
<td>AVERBIS</td>
<td>AEP, ATP, Pipeline, Example</td>
</tr>
<tr>
<td>0.3</td>
<td>15-07-2014</td>
<td>Kornél Markó</td>
<td>AVERBIS</td>
<td>Revision, Minor Modifications</td>
</tr>
<tr>
<td>0.4</td>
<td>21-07-2014</td>
<td>Livio Robaldo</td>
<td>UNITO</td>
<td>Contributions of UNITO</td>
</tr>
<tr>
<td>0.5</td>
<td>22-07-2014</td>
<td>Kiril Simov</td>
<td>IICT-BAS</td>
<td>Contribution of IICT-BAS</td>
</tr>
<tr>
<td>0.6</td>
<td>29-07-2014</td>
<td>Kornél Markó</td>
<td>AVERBIS</td>
<td>Example Update, Minor Modifications</td>
</tr>
<tr>
<td>1.0</td>
<td>31-07-2014</td>
<td>Karsten Gareis</td>
<td>empirica</td>
<td>Formatting for Submission</td>
</tr>
</tbody>
</table>

Date of delivery

<table>
<thead>
<tr>
<th>Contractual:</th>
<th>Actual:</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-07-2014</td>
<td>31-07-2014</td>
</tr>
</tbody>
</table>

Status

| final | draft |

Abstract

This document presents the components of the natural language processing toolkit for the text mining, analytics and search technologies of EUCases. The pipeline is built upon the UIMA-based AEP framework of Averbis. Natural language processing components from other partners are integrated into this framework by appropriate wrappers. This way, the semantic annotators of the AEP framework are available for all languages of the project documents.

Keywords

Natural language processing, tokenizer, sentence splitter, part-of-speech tagger, tag set, stemmer, lemmatizer, chunker, syntactic parser, coreference resolver, concept recognition, named entity recognition, pipeline, terminology, UIMA, type system, analysis engine, Akomo Ntoso, annotation

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This deliverable contains original unpublished work except where clearly indicated otherwise.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.
# Table of Content

Revision History, Status, Abstract, Keywords, Statement of Originality .................. 2
Table of Content ........................................................................................................ 3
List of Figures .............................................................................................................. 4
Executive Summary .................................................................................................... 4

1  The EUCases Natural Language Processing Pipeline .............................................. 5
   1.1  Averbis Extraction Platform (AEP) .................................................................... 5
   1.2  Averbis Type System .......................................................................................... 6
   1.3  Universal Tag Set ............................................................................................... 8
      1.3.1  Mapping from Unito ....................................................................................... 8
      1.3.2  Mapping from IICT-BAS ................................................................................ 9

2  Pipeline Components .............................................................................................. 11
   2.1  Components provided by Averbis ...................................................................... 11
      2.1.1  Input and Output Processors ........................................................................... 11
      2.1.2  Semantic Annotators ....................................................................................... 14
      2.1.3  Language-specific components ....................................................................... 15
      2.1.4  Additional Components .................................................................................. 17
   2.2  Components provided by Unito ......................................................................... 17
      2.2.1  Sentence Splitter ............................................................................................ 18
      2.2.2  Tokenizer ......................................................................................................... 18
   2.3  Components provided by IICT-BAS .................................................................. 18
      2.3.1  Sentence Splitter ............................................................................................ 19
      2.3.2  Tokenizer ......................................................................................................... 19

3  Terminologies .......................................................................................................... 20
   3.1  EuroVoc ............................................................................................................ 20
   3.2  Legal Taxonomy Syllabus .................................................................................. 21
   3.3  Geonames .......................................................................................................... 21

4  Pipeline .................................................................................................................... 22
   4.1  Installation .......................................................................................................... 22
   4.2  Configuration ...................................................................................................... 22
      4.2.1  Flow control ..................................................................................................... 22
      4.2.2  Input and Output ............................................................................................ 23
      4.2.3  Syntactical components .................................................................................. 23
      4.2.4  Regular expressions ....................................................................................... 24
      4.2.5  Concept mapper ............................................................................................. 24

5  Example .................................................................................................................... 26
   5.1  Akomo Ntoso Input ............................................................................................ 26
   5.2  Plain Text Conversion ......................................................................................... 28
Executive Summary

This document contains a description of the natural language processing (NLP) toolkit that will be used for the text mining tasks of the EUCases project. As the responsibility for five languages (Bulgarian, English, French, German, and Italian) relevant to the project is shared among the three project partner Averbis, IICT-BAS, and Unito according to their expertise, a major challenge during this work task was the integration of the different software components into a concise and comprehensive NLP toolkit.

The consortium has decided to build the toolkit on Averbis’ commercially approved UIMA-based Extraction Platform (AEP). Hence, by the help of adjustments to the components delivered by Unito and IICT-BAS, the broad range of semantic annotators of the AEP is available across all five project languages.

Additionally, specific pipeline components have been developed that deal with the Akomo Ntoso legal XML format of the input documents in the EUCases corpus. These components enable the creation of inline annotations within the Akomo Ntoso documents based on the output of the pipeline’s plain text annotators.
1 The EUCases Natural Language Processing Pipeline

The EUCases project aims at providing additional value like cross-references and metadata to legal open documents. In previous work tasks a large corpus of relevant legal documents in five European languages (Bulgarian, English, French, German, and Italian) has been identified and collected from different Open Data sources. In the next step, natural language processing capabilities are required because the content of these documents consists mainly of natural language texts.

The project consortium has agreed to apply the Apache Unstructured Information Management Architecture (UIMA) framework\(^1\) to this task because the consortium member Averbis has profound commercial and research experience in the usage of UIMA pipelines. Additionally, Averbis has built a specific environment—the Averbis Extraction Platform (AEP)—that simplifies the assembly and operation of natural language processing pipelines.

Averbis provides the components for the processing of English, French, and German texts. Moreover, Averbis supplies components for entity recognition, concept mapping, and reading and annotate the legal documents. The components for Bulgarian respectively Italian texts are provided by the consortium member IICT-BAS, respectively Unito.

However, as these are non-UIMA components, Averbis wrapped them into UIMA-compliant analysis engines by the help of both partners. In order to make the output of the Bulgarian and Italian components available to the rest of the pipeline the type system and the tag set for part-of-speech tags used by Averbis has to be employed.

The terminologies EuroVoc and Legal Taxonomy Syllabus that have been selected for the project are managed within the Averbis Terminology Platform (ATP), as part of the AEP, in order to make them available for concept mapping.

1.1 Averbis Extraction Platform (AEP)

The Averbis Extraction Platform (AEP), the text mining platform of Averbis, offers tools for the specific extraction of information from documents. The solution identifies single information units, as well as relevant facts and contexts which are of relevance. Here, many modular software components are implemented which are integrated in a comprehensive solution.

- Document classification: articles and text documents are automatically classified for indexing using a freely-definable category system. For example, agency notifications can be assigned to the corresponding departments (e.g. “Economy”, “Politics”).

- Concept recognition with the use of terminologies: here, the lexicon structure is flexible and enables the entry of synonyms and various attributes which play a role in the annotation process. The lexicon matching procedure can be carried out for associated or individual text blocks.

- Recognition of entities by mere statistical calculation of scores of different pieces of information and attributes of context words, thus precisely identifying persons, product names, organizations or geographic information.

- Sentiment Analysis and Opinion Mining: Qualitative value assessments are recognized reliably and evaluated on sentence level.

\(^1\) [http://uima.apache.org/](http://uima.apache.org/)
The AEP is fully embedded in the Apache UIMA Framework. This framework provides interfaces to connect individual analysis components to each other as well as to determine a specific execution order. Thus, a transparent and standardized connectivity between the individual analysis components is guaranteed. On top of UIMA, Averbis provides a full set of annotators and tools that further simplify text mining tasks:

- The AEP provides a huge set of annotators ranging from domain independent to domain specific solutions. Each of these annotators ships with unique outstanding features resulting in high performance annotation pipelines.
- The AEP comes with a powerful tool with a rich graphical user interface to interactively annotate documents and make examinations on the annotation results. Here, the user can choose between different pipelines available, as well as specific annotators within a specific pipeline.

Besides the UIMA pipeline, the Averbis Terminology Platform (ATP), a comprehensive tool for handling terminologies and ontologies, is a part of the AEP, too. The term hierarchies aid in the formal sorting of the individual fields of expertise and establish rules on the contexts of the corresponding expressions, thus enabling conclusions to be drawn from the existing data, contradictions to be detected and missing knowledge to be added.

For the project, the terminologies EuroVoc and Legal Taxonomy Syllabus were imported into the ATP in order to make them available to the AEP’s concept mapping engine.

### 1.2 Averbis Type System

To support the design goal of data modelling and interchange between different pipeline components, UIMA requires that all annotations conform to a user-defined schema, called type system. A type system is a collection of inter-related type definitions. Type systems are user-defined. UIMA does not specify a particular set of types that developers must use. Developers define type systems to suit their application’s requirements.

As an integral part of the Averbis Extraction Platform, there is the Averbis Type System (see Figure 1) – a UIMA compliant annotation type system covering all levels of text analysis including structural, syntactic and semantic processing of textual documents.

This type system has been designed in-house by the Averbis core development team in order to meet the requirements of large-scale business-problems. In many aspects, it is similar in spirit with the type system described by Hahn et al. (2007)\(^2\).

Within the Averbis Type System, there are two entry points:

- Annotations that apply to the whole document instead to a specific position inherit from the type uima.tcas.DocumentAnnotation. This holds, for example, for the Category annotation by which predefined or automatic categories can be assigned to a document.
- The second and much larger group of annotations within the Averbis Type System are those annotations which are bound to a specific position in the document. A typical example is the Entity-Annotation which spans over the respective entity mention within the text. This group covers several layers of text processing and the respective types:
  - Structure and metadata layer: including types for different document zones (title, abstract, section etc.) and generic meta data annotation types

---

- Syntax layer: including types for most syntactic processing steps starting from sentences, tokens, part-of-speech tags, chunks, stems, lemmata etc.
- Semantic layer: including types for most common semantic structures including entities, concepts, relations between both, extracted keywords etc.
- Evaluation layer: including types for automated evaluation of annotations including false positive, true positive and false negative annotations.

**Figure 1 - The Averbis Type System (not all types included for reasons of clarity).**
The Averbis Type System can easily be extended and adapted for project specific needs (see the extension packs in Figure 1). Potential type additions for the EUCases project do not interfere with AEP core components. Currently, there are no EUCases specific types yet.

1.3 Universal Tag Set

Averbis provides natural language processing for several languages. In order to reduce the number of language-specific components Averbis has decided to use only one part-of-speech tagset. This means that all components that rely on part-of-speech tags can be used almost independently from the processed language. The employed tagset is taken from A Universal Part-of-Speech Tagset from Petrov, Das, and McDonald (2011)\(^3\).

This tagset consists only of twelve tags:

- **NOUN** nouns
- **VERB** verbs
- **ADJ** adjectives
- **ADV** adverbs
- **PRON** pronouns
- **DET** determiners and articles
- **ADP** prepositions and postpositions
- **NUM** numerals
- **CONJ** conjunctions
- **PRT** particles
- **.** punctuation marks
- **X** other, e.g., abbreviations and foreign words

Both Unito and IICT-BAS mapped their part of speech tags onto this tagset in order to make the output of their language-specific components compatible to the other components such as concept mapping of the EUCases pipeline.

1.3.1 Mapping from Unito

In the EUCases project, Unito will use its linguistic environment called TULE (Turin University Linguistic Environment). TULE is available in lisp and Java; for the EUCases project, Unito decided to use the Java version of TULE.

The TULE tagset is very similar to the Universal Part-of-Speech tagset, so that the mapping is straightforward. TULE’s tagset includes the following part-of-speech tags:

- **ADJ** adjectives, e.g. alto (tall), qualche (some), etc.
- **ADV** adverbs, e.g. rumorosamente (loudly), presto (quickly), etc.
- **ART** articles, e.g. il (the), una (a), etc.
- **CONJ** conjunctions, e.g. e (and), ma (but), etc.
- **INTERJ** interjections, e.g. ciao (hello), addio (goodbye).
- **MARKER** markers, e.g. tags in XML files.

- **NOUN** nouns, e.g. tavolo (table), mela (apple), etc.
- **NUM** numbers, e.g. 1, 3, etc.
- **PREDET** predeterminers, e.g. entrambi (both), tutti (all), etc.
- **PREP** prepositions, e.g. su (on), senza (without), etc.
- **PRON** pronouns, e.g. tu (you), loro (they), etc.
- **PUNCT** punctuation, e.g. the comma, the question mark, etc.
- **SPECIAL** special symbols, e.g. &; £, $, etc.
- **VERB** verbs, e.g. mangiare (to eat), cantare (to sing), etc.

Therefore, the categories **ADV**, **CONJ**, **NOUN**, **NUM**, **PRON**, and **VERB** are left unchanged: they correspond to the universal homonymous Part-of-Speech.

On the other hand, **ART** and **PREP** are converted to **DET** and **ADP** respectively. **PUNCT** and **SPECIAL** are all converted to “.” [dot].

Adjectives are more complex, as many of them could be determiners. For instance, in “queste mele” (these apples), the word “queste” (queste) is a demonstrative adjective, but it is also a determiner, in that it acts as a definite article of the word “mele” (apples). For identifying these cases, we inspect the dependency tree: in case the adjective has an argument (marked with the syntactic label ARG), we convert it in **DET**, otherwise we leave it as **ADJ**.

**PREDET** are special cases of non-determiner indefinites adjectives, so that they are mapped into **ADJ**.

**INTERJ** and **MARKER** are mapped into X. However, note that legal text should not contain words belonging to these two part-of-speechs.

Finally, in Italian it is possible that articles and pronouns are respectively amalgamated into prepositions and verbs. For instance, “della” is a compound that amalgams the preposition “di” (of) and the article “la” (the, feminine), while “mangiando” amalgams the verb “mangiando” (eating) with the pronoun “lo” (it). The TULE parser signals such cases, which are mapped into the Universal Part-of-Speech **PRT**.

### 1.3.2 Mapping from IICT-BAS

The Bulgarian pipeline's tagging module uses a rich morphosyntactic tagset of 680 tags. The BTB-tagset is a positional tagset and contains detailed information about the morphosyntactic properties of each word. For instance, the tag "Ncmsi" indicates that a word is a common noun (Nc), masculine (m), singular (s), indefinite form (i). The information encoded in the tagset is valuable for achieving good performance during parsing of Bulgarian data.

In order to ensure correspondence to the modules provided by Averbis, the tagset was mapped to the Universal Tagset. The mapping is as follows:

- The common noun forms, the proper names and the family names are mapped to Noun
- The verbal word forms are mapped to Verb
- The adjective word forms are mapped to Adj
- The adverb forms are mapped to Adv

---

• All the pronouns are mapped to Pron
• All the prepositions are mapped to Adp
• The numeral forms are mapped to Num
• The conjunctions are mapped to Conj
• The particles are mapped to Prt
• Punctuation marks are mapped to . [dot]
• All the rest forms are mapped to X

In Bulgarian, articles and determiners are part of the word forms. Therefore, there is no mapping to DET.
2 Pipeline Components

The consortium decided to build the natural language processing pipeline on the comprehensive facilities provided by the Averbis Extraction Platform in favor of a coherent approach to the processing of the five languages (Bulgarian, English, French, German, and Italian) relevant to the project. That means that the AEP’s commercially approved semantic annotators like concept mappers or entity recognizers can be used and applied consistently across language borders without the need for a language-specific redevelopment of such components. Thus, only the syntax analysis remains language-specific.

Of course, the components delivered by the partners Unito and IICT-BAS must be adjusted in a way that makes their behavior similar to the language-specific components of Averbis. However, this can be achieved with little effort: First, a UIMA wrapper has to be created for each component. Second, a component’s annotations have to be mapped onto the Averbis Type System including their start and end positions. Third, the part of speech tags have to be mapped onto the universal tagset used by Averbis.

The remaining part of this section gives an overview about the pipeline components provided by each partner including the adaptions made to the non-UIMA components. In order to reduce the possible complications and to gain experience with the chosen processing model the language-specific parts have been reduced to sentence detection, tokenization, and part-of-speech tagging. In a first release of the pipeline, these steps are sufficient for the semantic annotation process.

2.1 Components provided by Averbis

Averbis supplies the complete natural language processing pipelines for the languages English, French, and German. As described above, the semantic annotation components (e.g. concept mapping) of these pipelines are also used for the two remaining languages Bulgarian and Italian.

Apart from the language-specific syntax analysis and the semantic annotation there is a demand for pipeline components dealing with the Akomo Ntoso legal XML format that is used for the documents in the EUCases corpus. The problem and the solution to it are explained in the following section. The syntax analysis and the semantic annotation components are described thereafter.

2.1.1 Input and Output Processors

The natural language processing components provided by the project partners are working on plain text. That means that any other format must be converted into plain text before any useful processing can take place. Each annotation of following pipeline components is given with start and end positions with respect to the plain text.

However, the input and output format for the EUCases pipeline is the Akomo Ntoso legal XML format, that is, the text content of such an XML document has to be analyzed and the annotations have to be written back into the original XML document.

A major part of the annotations in Akomo Ntoso are inline annotations that surround the respective text parts, e.g. “... the <organization>parliament of Kenya<organization> approved ...” where the organization element declares that the enclosed text refers to an organization. Additional descriptions about such an annotation, like a reference to the

http://www.akomantoso.org/
respective Linked Data source or any other normalized form (date and time values, for instance), can be put directly into some attributes of the annotation element or be put into the metadata section of Akomo Ntoso documents. In the latter case, the reference between the annotation and its metadata is established by its identifier. An explanation about annotation approach in Akomo Ntoso is given by Barabucci et. al. (2010)\(^6\).

The very exact way of writing the annotations back to the Akomo Ntoso legal XML documents is not yet finally clear and has to be specified by the project partners. However, the major part of transforming the plain text annotations of the natural language processors into XML inline annotations has been solved by two pipeline components, \texttt{Xml2TextAE} and \texttt{XmlAnnotatorAE}.

\textbf{Xml2TextAE}

This component transforms any XML document into plain text to makes it available to the natural language processing task. Unfortunately, this transformation is not straight-forward.

\textbf{Problem description}

The natural language processing pipeline components expect plain text as input. Annotations are made stand-off (apart from the input text) with start and end markers regarding single character positions inside the text. Thus, only the text content of an XML file must be passed to these components.

However, just taking the text content of the XML root node may not yield to the expected results. This is because separate text blocks can be contained in separate XML elements which are not necessarily spaced by text nodes in between. Thus, the text content of XML documents may contain erroneously joined texts potentially disturbing the text mining results. The following examples illustrate the problem with simplified XHTML documents.

```html
<html>
<head><title>Example</title>
<body>
  <p>Hello World!</p>
  <p>This is an example.</p>
</body>
</html>

<html>
<head><title>Example</title>
<body>Hello World!</p><p>This is an example.</p>
</body>
</html>
```

The HTML paragraph element \texttt{p} means that it contains a closed part of text and therefore “Hello World!” and “This is an example.” are to be considered independent. The text content of the first example does reflect this due to the line break and the whitespace after the first paragraph element. The text content of the second example, though, does contain the wrong text “Hello World!This is an example.”

Unfortunately, simply introducing a space after each XML element could also lead to false results because of inline elements that apply to parts of a word. A short example makes that clear: \texttt{<p>Hello <b>Wo</b>rld!>}. Here, the first two letters should be written in bold face but must not be separated from the rest of the word.

---

Solution

A solution to the problem includes distinguishing between block, inline, and container elements. In the text transformation, block elements can be enclosed by newlines but inline elements not. Container elements can be skipped. The type of the elements can be declared by a document type definition, for instance. This is the case for HTML and Akomo Ntoso. Nevertheless, Averbis has developed a generic approach that is independent from schema definitions.

This approach is implemented in the pipeline component `Xml2TextAE`. Beginning at the root the transformation from XML to text visits the XML elements in pre-order:

- If the children of the element contain a text node with any content other than whitespace, the element is rated as block element. Additionally, block elements are considered to contain only inline elements. Therefore, the whole text content of the element enclosed in newlines is returned in order to guarantee the separation of block elements in the transformation. The traversal continues in pre-order as if the current element was a leaf of the tree.
- Otherwise the element is considered as container element being irrelevant for the transformation. The possible whitespace is taken to be insignificant whitespace that is present only for pretty-printing. The traversal continues as usual in pre-order.

To give an example, consider the previous HTML documents from above. According to the approach, the elements `html`, `head`, and `body` are container elements and do not influence the transformation, and the elements `title` and `p` are block elements. The element `b` from the abbreviated example above is an inline element. Thus the text transformation will be in each case:

Example

Hello world!

This is an example.

XmlAnnotatorAE

After the natural language processing has finished on the text transformation the obtained annotations must be inserted into the original XML document. The annotations are given with start and end markers regarding character positions of the analyzed text. These positions have to be mapped to text nodes of the XML document in order to insert annotation elements at the correct place.

Mapping positions to text nodes

The pipeline component `XmlAnnotatorAE` implements a procedure that finds the correct text node and the offset inside the text node by counting the characters in document order from the beginning up to the required position. The counting does revert basically the effects of the above transformation from XML to text. It omits characters that are insignificant whitespace according to the transformation but increases the counter for the additional newlines when it visits a block element. The additional newlines are considered to be included inside the block element if an annotation should point to it.

Considering the transformation output from the previous section, position 0 points to the newline – suppose each newline being a single character – at the beginning. The counting procedure maps it to the first character of the text node below the `title` element. Position 1
is mapped to the same character, too. A sentence annotation from position 10 to 22 affects the text node below the first p element.

Creating annotations

As explained before, the most important part of an annotation has to be inline surrounding the respective text content of an Akomo Ntoso legal XML document. However, there might be the case that the annotation conflicts with the present hierarchical structure of the document. For instance, this is the case for an annotation ranging from position 16 to 28. The positions belong to two different text nodes with different parent nodes. This problem may also arise if the natural language processing generates overlapping annotations, for example from position 24 to 31 and from 29 to 34.

Currently, the XmlAnnotatorAE component uses so-called milestones in such cases. Milestones are a proposal of the Text Encoding Initiative\(^7\) and refer to empty XML elements that denote the start and the end of a section. Both elements are usually related to each other by some unique attribute value.

The milestones approach is fairly easy to implement. Other approaches that split a range which does not fit into the XML tree into smaller pieces such that each piece can be inserted into the hierarchy require more sophisticated computations. Also, additional attributes connecting the individual pieces by some unique value are necessary, too.

To illustrate the said we provide an example for both approaches for the annotation from position 16 to 28 beginning with the milestones.

```html
<html>
<head><title>Example</title>
<body>
  <p>Hello <annotation id="m1" pos="begin"/>World!</annotation></p>
  <p><annotation idref="m1" pos="end"/> is an example.</annotation></p>
</body>
</html>

<html>
<head><title>Example</title>
<body>
  <p>Hello <annotation id="s1">World!</annotation></p>
  <p><annotation idref="s1">This</annotation> is an example.</annotation></p>
</body>
</html>
```

The appropriate method certainly depends on the usage of the annotated documents. As this is not yet clear Averbis opted for the simpler approach. Of course, a later replacement is no difficulty at all.

2.1.2 Semantic Annotators

The Averbis Extraction Platform provides a comprehensive set of semantic annotators. Owing to the integration of the project partner’s natural language technologies into the UIMA framework of the AEP, these annotators can be employed for all document languages of the EUCases project.

The currently configured pipeline uses a concept mapper based on the terminologies Eurovoc, Legal Taxonomy Syllabus, and Geonames (see Section 3 for details).

Additionally, a regular expression annotator is used to recognize dates and legal references (preliminary, part of work task 3.3).

Concept Mapper
With the annotations generated by the previous pipeline components the AEP concept mapper is able to create mappings between concepts of a terminology and free text phrases in various morphological layers such as word-, stem- and subword-level. Based on an input sequence of words and their corresponding part-of-speech tags, a linguistic filter can be defined by means of regular expressions over the input sequences. As a result a list of candidate terms is generated that can be sorted according to different measures like term weight or absolute frequency. The concept mapper contains a number of features such as concept disambiguation and filtering to improve the mapping process. Nevertheless, the background terminology essentially determines the overall quality of the mapping. Hence, to guarantee valuable mappings the terminology has to be modelled closely to the content’s domain and should be extended constantly.

Named Entity Recognizer
Entities such as persons, organizations or places can be identified with concept mapping given appropriate lists of instances. However, the maintenance of such lists can be very time consuming or even impossible when cannot be known in advance. In such situations machine learning based approaches that are able to generalize the identification of entities constitute an adequate solution. The named entity recognition used in the AEP pipeline is able to identify named entities based on trained statistical models using Conditional Random Fields.

However, a sufficiently large set of training documents must be available before useful results can be expected. Therefore, the pipeline configuration still excludes this component. Places are currently recognized by the aid of the Geonames terminology and concept mapping.

Regular Expression Annotator
Some entities like dates or e-mail addresses can reliably be identified by regular expressions. In the EUCases project, references to legal resources are also a subject to regular expressions. The AEP contains an analysis engine for the detection of such entities. The regular expressions are given in a separate configuration file.

2.1.3 Language-specific components
Averbis provides the language-specific processing components for English, French, and German. In a basic setup, only the sentence splitter, the tokenizer, and the part-of-speech tagger are required. However, additional components can already be applied without interference. Stemming, morpho-semantic analysis, and chunks can be useful for free-text search but also for concept mappers.

The accuracy of machine-learning based components is outlined with respect to the training corpora and language.

Sentence Splitter
Due to the ambiguity of punctuation the problem of sentence boundary recognition is known to be a non-trivial task. The AEP contains a sentence splitter relying on machine learning
methods provided by the OpenNLP framework\(^8\). Models trained on manually annotated text corpora (first row in the following tables) are available in different languages, such as German or English. Additionally, a simpler rule-based sentence splitter is also available. The Accuracy estimation is given in Table 1.

Table 1 - Sentence Splitter: Accuracy

<table>
<thead>
<tr>
<th>Corpus</th>
<th>ELRA Crater</th>
<th>FRAMED</th>
<th>Tiger</th>
<th>Genia</th>
<th>OANC-Slate</th>
<th>MASC-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Fr</td>
<td>De</td>
<td>de</td>
<td>en</td>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>F1</td>
<td>0.420</td>
<td>0.960</td>
<td>0.970</td>
<td>0.986</td>
<td>0.939</td>
<td>0.915</td>
</tr>
</tbody>
</table>

**Tokenizer**

Tokens describe single words as well as punctuation marks such as periods and commas. Just like in the case of the sentence recognition problem, words itself may contain numbers ("B2B"), additional characters such as ("Dr."), or dashes ("e-mail") making tokenization a non-trivial task. The AEP tokenizer is based on a machine learning approach using algorithms provided by the OpenNLP framework and trained on general text corpora in different languages. There is also a rule-based tokenizer included in the AEP. Accuracy values are shown in Table 2.

Table 2 - Tokenizer: Accuracy

<table>
<thead>
<tr>
<th>Corpus</th>
<th>ELRA Crater</th>
<th>FRAMED</th>
<th>Tiger</th>
<th>Genia</th>
<th>OANC-Slate</th>
<th>MASC-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Fr</td>
<td>De</td>
<td>de</td>
<td>en</td>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>F1</td>
<td>0.990</td>
<td>0.997</td>
<td>0.999</td>
<td>0.998</td>
<td>0.998</td>
<td>0.990</td>
</tr>
</tbody>
</table>

**Stemmer**

Reducing the number of inflections by mapping terms to their stem improves the concept mapping process and reduces the feature in the case of statistical methods. The AEP stemmer is rule based and available for a number of languages.

**Morpho-Semantic Analyzer**

Many Indo-European languages, such as German or Russian, are fusional languages, characterized by complex forms of composition, derivation and inflection. This makes morphological analysis an inevitable subtask for processing text in such languages because stemming algorithms prove to work rather inefficient here. Language-specific sub-words sharing the same meaning are linked within and across languages and grouped into concept-like equivalence classes.

**Part-of-Speech Tagger**

Based on the tokens generated by the Tokenizer, the AEP part-of-speech tagger assigns part-of-speech information, such as nouns, articles, verb and so on to each of these tokens. It is a statistical tagger trainable on different languages and tagsets. Its accuracy estimation is shown in Table 3.

Table 3 - POS Tagger: Accuracy

<table>
<thead>
<tr>
<th>Corpus</th>
<th>ELRA Crater</th>
<th>FRAMED</th>
<th>Tiger</th>
<th>Genia</th>
<th>OANC-Slate</th>
<th>MASC-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Fr</td>
<td>De</td>
<td>de</td>
<td>en</td>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>F1</td>
<td>0.971</td>
<td>0.971</td>
<td>0.966</td>
<td>0.985</td>
<td>0.979</td>
<td>0.952</td>
</tr>
</tbody>
</table>

\(^8\) [http://opennlp.apache.org/](http://opennlp.apache.org/)
Chunker

Given the part-of-speech annotations chunks of tokens are generated by the AEP chunking. Typical chunks are noun phrase chunks grouping nouns and modifiers describing the nouns in more details. Noun phrase chunks play a central role for a number of subsequent pipeline steps. In most case the subject of analysis of analysis engines is restricted to these kinds of chunks in order to reduce the number of false positive annotations. This component is based on a statistical approach, too. The accuracy values are given in Table 4.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>FRAMED</th>
<th>Tiger</th>
<th>Genia</th>
<th>OANC-Slate</th>
<th>MASC-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>de</td>
<td>De</td>
<td>en</td>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>F1</td>
<td>0.942</td>
<td>0.959</td>
<td>0.939</td>
<td>0.960</td>
<td>0.887</td>
</tr>
</tbody>
</table>

2.1.4 Additional Components

Currently, there are two more components in the pipeline. Language guessing is important to determine the correct flow. Stopword tagging can also be enabled but is of minor importance for the project.

Language Guesser

Language detection is a very important task in natural language processing due to the fact that most of the subsequent analysis engines in the processing pipeline are language dependant.

The component reliably distinguishes more than twenty languages including the five project-relevant languages. The target languages can be restricted if a-priori knowledge about the documents is available in order to improve the accuracy. It works with n-gram based statistical language model features and multi-class Support Vector Machines.

After setting the document language, a document get send to the language-specific sub-flow of the EUCases pipeline.

Stopword Tagger

Stopwords are normally not useful for document retrieval methods based on term frequency and inverse document frequency. At present, such retrieval is not foreseen in the EUCases project thus stopword tagging is of minor importance. Still, it may also be used for the AEP concept mapper for fine-tuning the results.

2.2 Components provided by Unito

Unito’s natural language processing components are contained in Java libraries. Therefore, the integration into the Java based Averbis Extraction Platform is considerably easy. The main source of problems could be incompatible third-party libraries that are used by Averbis and Unito. Happily, this was no issue until now.

As stated above, the partners agreed to begin with a simplified setup requiring only the three components sentence splitter, tokenizer, and part of speech tagger.
2.2.1 Sentence Splitter

The output of the sentence splitter and its wrapper to place it into the UIMA framework employed by the AEP is fairly simple. Unito’s library needs to expose a method that takes a plain text as input and delivers the start and end positions of the contained sentences. In the current state, it delivers only end positions but it is not necessarily the case that one sentence starts where the previous one ends.

As the sentence splitter outputs only sentences, the mapping onto the Averbis Type System is not difficulty at all: each sentence is assigned to the type `de.averbis.extraction.types.Sentence`.

2.2.2 Tokenizer

The output of the tokenizer is a list of tokens. Each token is mapped by the UIMA wrapper to the type `de.averbis.extraction.types.Token` of the Averbis Type System. Similar to the AEP tokenizer, the wrapper provides the option to store an additional normalized version of the token’s text.

2.2.3 Part-of-Speech Tagger

Different from the sentence splitter and the tokenizer, the output of the part-of-speech tagger must include additional information beside the start and end position markers. Each token must convey its original tag and the respective tag (both as Java strings) from the universal tagset according to the mapping in Section 1.3.1.

The UIMA wrapper creates annotations according to the universal tags:

- NOUN `de.averbis.extraction.types.POSTagNoun`
- VERB `de.averbis.extraction.types.POSTagVerb`
- ADJ `de.averbis.extraction.types.POSTagAdj`
- ADV `de.averbis.extraction.types.POSTagAdv`
- PRON `de.averbis.extraction.types.POSTagPron`
- DET `de.averbis.extraction.types.POSTagDet`
- ADP `de.averbis.extraction.types.POSTagAdp`
- NUM `de.averbis.extraction.types.POSTagNum`
- CONJ `de.averbis.extraction.types.POSTagConj`
- PRT `de.averbis.extraction.types.POSTagPart`
- `[dot]` `de.averbis.extraction.types.POSTagPunct`
- X `de.averbis.extraction.types.POSTag`

The original tag is stored in the attribute `value` which is available in each of the classes. The semantic annotators of the AEP do not require it but it may be used by project-specific components or other annotators delivered by project partners.

2.3 Components provided by IICT-BAS

Similarly to the Unito pipeline, the components provided by IICT-BAS are implemented in Java – a fact which greatly simplifies their integration into the Averbis Extraction Platform. In the first step, IICT-BAS provided the three basic components for sentence splitting, tokenization, and part of speech tagging.
The Bulgarian processing module includes a combination of rule-based components (tokenization and sentence splitting), and hybrid components (part-of-speech tagging). The latter makes use of statistical techniques and linguistic resources in the form of a morphological lexicon and gazetteers.

Since Bulgarian is an analytical language with rich word inflection, the task of part-of-speech tagging becomes more complex. It is better defined as morphosyntactic annotation due to the high number of grammatical features encoded in the tagset.

### 2.3.1 Sentence Splitter

The processes of tokenization and sentence boundary detection are highly dependent on each other, and are therefore carried out jointly in the Bulgarian processing module. The UIMA wrapper for this step stores each detected sentence as an Averbis Type System annotation `de.averbis.extraction.types.Sentence`. The input to the module is in plain text format. The start and end positions of each sentence are stored in accordance with the start position of the first token in the sentence and the end position of the last token in the sentence.

### 2.3.2 Tokenizer

The UIMA wrapper for the tokenization and sentence splitting module stores each detected token into the type `de.averbis.extraction.types.Token` and optionally normalizes the token's text. The output of the module already includes information about the position of each token, as well as additional information such as the lemma of the token. This additional information could prove useful in the following processing steps, but is currently not stored.

### 2.3.3 Part-of-Speech Tagger

Beside the annotation range, the part-of-speech tagger returns the original tag used internally by the natural language processing components of IICT-BAS and the corresponding universal tag (see Section 1.3.2 for the mapping).

The UIMA wrapper creates the corresponding types for the universal tags. Of course, the mapping for it is the same as the one given in Section 2.2.3. The original tag is also stored in the UIMA annotation as a feature "value" and is available to specific components in the subsequent pipeline flow.
3 Terminologies

The EUCases consortium has selected two terminologies for categorizing the legal documents: EuroVoc\(^9\) from the European Union and the Legal Taxonomy Syllabus\(^10\) that is provided by the consortium member Unito. Apart from terms, the Legal Taxonomy Syllabus contains relations from terms to European and national legal concepts and between these concepts. Thus, it helps to identify related documents across language borders and national legal systems.

Furthermore, the Geonames\(^11\) terminology is used for the recognition of location information. There are several other terminologies accessible through the Averbis Terminology Platform (ATP) that can easily be added to the EUCases pipeline when necessary.

3.1 EuroVoc

The EuroVoc terminology is used for the categorization of the legal documents. The current version 4.3 was imported into the ATP making it available to the AEP concept mapper (see Figure 2).

---

9 http://eurovoc.europa.eu/
10 http://legal-informatics.di.unito.it/syllabus/
11 http://www.geonames.org/
3.2 Legal Taxonomy Syllabus

Like EuroVoc, the Legal Taxonomy Syllabus in its current version 2 was imported into the ATP (see Figure 3) based on the Syllabus’ OWL export. It enables the identification of European and National legal concepts. Apart from classification purposes, this terminology provides for finding similar or related documents across different languages through its relations between European and national legal concepts.

However, these relations are not relevant for the annotation process by the AEP concept mapper. Even so, they have been imported into the ATP as XREF links to supply an editor with this context information. Finding related documents, however, will be done by taking the annotations from a document and searching the related legal concepts on national or European level within the original Legal Taxonomy Syllabus.

Figure 3 - Legal Taxonomy Syllabus (Averbis Terminology Platform).

3.3 Geonames

The Geonames terminology containing the names of countries and cities is frequently used by Averbis. It is already contained in the Averbis Terminology Platform. Obviously, it serves to recognize places.
4 Pipeline

This section briefly covers the current pipeline installation and configuration.

4.1 Installation

The Averbis Annotation Editor (see Figure 4) and the Averbis Terminology Platform (see Figure 2) have been installed on the EUCases backend server techno.eucases.eu (operated by the consortium member APIS).

The EUCases natural language processing pipeline is installed as Windows service permanently accessible through the Apache ActiveMQ message broker\(^\text{12}\). The Annotation Editor offers a comfortable user interface to execute it. Additionally, there are tailored shell scripts that enable the batch processing of sample texts. Both methods are well suited for testing and improving the pipeline’s performance.

Figure 4 - Averbis Annotation Editor.

4.2 Configuration

The configuration of an AEP pipeline is contained in a single file. The pipeline components, the pipeline flow, and the broker endpoint are set up by a list of key-value-pairs.

4.2.1 Flow control

The pipeline is configured for the five project languages. A dummy configuration is added in case that the language guesser fails to choose an allowed language.

\footnotesize
# project languages

\footnotesize\(^\text{12}\) \url{http://activemq.apache.org/}
aep.LanguageCategorizerAE.allowedLanguages=de,en,fr,bg,it
# sets document language to “zz”, see corresponding flow below
aep.LanguageCategorizerAE.useUnknownLanguage=true

# language independent pipeline for all documents to be processed, “AAE”
# refers to the flow controller (below), dependent on the languages
aep.fixed.flow =
    BinaryFileCR,Xml2TextAE,LanguageCategorizerAE,AAE,RegExAnnotatorAE,EntityTokenLinkerAE,ConceptMapperAE.geo,ConceptMapperAE.eurovocexact,ConceptMapperAE.eurovoc,ConceptMapperAE.syllabusexact,ConceptMapperAE.syllabus,XmlAnnotatorCC,XmiCC

# settings for the language specific pipeline elements
aep.AAE.flowController = DocumentLanguageFC
aep.DocumentLanguageFC.flow[it] = it_flow
aep.group.de_flow =
aep.group.en_flow =
aep.group.fr_flow =
    SentenceMaxentAE,SimpleTokenizerAE,SnowballStemmerAE,MorphoSemanticAE,POSTaggerMaxentAE,ChunkerMaxentAE
aep.group.it_flow =
aep.group.bg_flow =
    SimpleSentenceSplitterAE,SimpleTokenizerAE,RegExAnnotatorAE,EntityTokenLinkerAE,ConceptMapperAE.geo,ConceptMapperAE.eurovocexact,ConceptMapperAE.eurovoc
aep.group.zz_flow = SimpleSentenceSplitterAE,SimpleTokenizerAE

4.2.2 Input and Output

# location and type of the input documents
aep.BinaryFileCR.sofaID = binary
aep.BinaryFileCR.mimetype = application/xml
aep.BinaryFileCR.inputDirectory = C:/uima/eucases/xml

# location for the output of processed documents
aep.XmiCC.outputDirectory = C:/uima/eucases/output
aep.XmlAnnotatorCC.sofaID = binary
aep.XmlAnnotatorCC.outputDirectory = C:/uima/eucases/output

4.2.3 Syntactical components

Currently, a simple tokenizer is used for the pipeline. The statistical tokenizer has to be adjusted to the legal documents based on a sufficiently large sample corpus. The morphosyntactic analysis is for decoumpounding complex word forms. The sentence detector additionally makes sentence annotations when a linebreak occurs.
aep.MorphoSemanticAE.msiEngineLanguages=de, en, fr, it
4.2.4 Regular expressions

Regular expressions are used for detecting and normalizing dates. The rules are stored in a separate file. The resulting annotations will be linked to the token annotations.

```java
aep.RegExAnnotatorAE.ConceptFiles=customer/regexFiles/regexFile.xml
aep.EntityTokenLinkerAE.sourceComponentID=org.apache.uima.annotator.regex.impl.RegExAnnotator
```

4.2.5 Concept mapper

The concept mapper can be run in default (applying stemming, morphosyntactic analysis and/or lower case) and exact mode (e.g. for acronyms). In exact mode, unless the spelling of a term in the dictionary and the spelling of a word in the document correspond exactly no match is reported. This mode is usually necessary for terminologies containing abbreviations or codes instead of words. For brevity, the configurations for the Geonames and the Syllabus terminology are omitted.

```java
aep.ConceptMapperAE.eurovocexact.dictionarySourceFiles =
    customer/dictionaries/concept_mapper/eurovoc_4_3.exact.xml
aep.ConceptMapperAE.eurovocexact.dictionaryPreprocessingPipelineElements =
    SimpleTokenizerAE
aep.ConceptMapperAE.eurovocexact.mappingMode = original
aep.ConceptMapperAE.eurovocexact.useSerializedDictionary = true
aep.ConceptMapperAE.eurovocexact.serializedDictionaryDirectory =
    customer/dictionaries/concept_mapper/
aep.ConceptMapperAE.eurovocexact.caseMatch = CASE_MATCH
aep.ConceptMapperAE.eurovocexact.orderIndependentLookup = false
aep.ConceptMapperAE.eurovocexact.matchOnlyTermsWithNouns = true
aep.ConceptMapperAE.eurovocexact.searchStrategy = CONTIGUOUS_MATCH
aep.ConceptMapperAE.eurovocexact.addSentenceAnnotationsWhilePreprocessing =
    true
aep.ConceptMapperAE.eurovocexact.dictionaryPreprocessingPipelineThreads = 1
aep.ConceptMapperAE.eurovocexact.spanFeatureStructure =
    de.averbis.extraction.types.Sentence
aep.ConceptMapperAE.eurovocexact.mapResolvedAbbreviations = false
aep.ConceptMapperAE.eurovocexact.filterBestMatches = true
aep.ConceptMapperAE.eurovocexact.findAllMatches = false
aep.ConceptMapperAE.eurovocexact.ignoreByConceptMapperAfterMapped = true
aep.ConceptMapperAE.eurovocexact.checkStopwordFlag = false
aep.ConceptMapperAE.eurovocexact.makeConceptAnnotation = true
```

```java
aep.ConceptMapperAE.eurovoc.dictionarySourceFiles =
    customer/dictionaries/concept_mapper/eurovoc_4_3.xml
aep.ConceptMapperAE.eurovoc.dictionaryPreprocessingPipelineElements =
    SimpleTokenizerAE,MorphoSemanticAE
aep.ConceptMapperAE.eurovoc.mappingMode = segment
aep.ConceptMapperAE.eurovoc.useSerializedDictionary = true
aep.ConceptMapperAE.eurovoc.serializedDictionaryDirectory =
    customer/dictionaries/concept_mapper/
```
aep.ConceptMapperAE.eurovoc.dictionaryPreprocessingPipelineThreads = 4
aep.ConceptMapperAE.eurovoc.spanFeatureStructure =
    de.averbis.extraction.types.Sentence
aep.ConceptMapperAE.eurovoc.mapResolvedAbbreviations = false
aep.ConceptMapperAE.eurovoc.filterBestMatches = true
aep.ConceptMapperAE.eurovoc.findAllMatches = false
aep.ConceptMapperAE.eurovoc.ignoreByConceptMapperAfterMapped = false
aep.ConceptMapperAE.eurovoc.checkStopwordFlag = true
aep.ConceptMapperAE.eurovoc.makeConceptAnnotation = true
5 Example

The document concludes with a concrete example. A document from the Austrian legislation (JFR_10199695_78B00652_01.xml) is send through the pipeline and the annotated XML output is shown. The pipeline can be run through the file upload dialog of the Annotation Editor (see Figure 5).

5.1 Akomo Ntoso Input

The input document is present in Akomo Ntoso legal XML. It contains an Austrian judgment. The documents text is very long thus an excerpt is shown. Omissions are indicated by "[...]". The text content is emphasized with **bold font**.

```xml
<?xml version="1.0"?>
<akomaNtoso xmlns:euc="http://eucases/prop"
xmlns="http://docs.oasis-open.org/legaldocml/ns/akn/3.0/CSD10">
  <judgment name="doc_judgment">
    <meta>
      <identification source="#eucases">
        <FRBRWork>
          <FRBRthis value="/at/judgment/OGH/r/1980-03-05/B652-78/main" />
          <FRBRuri GUID="#SourceURI1" value="JFR_10199695_78B00652_01" />
          <FRBRuri GUID="#SourceURI2" value="JFR_10199695_78B00652_00" />
          <FRBRalias value="B 652/78" name="AliasCaseNr" />
          <FRBRdate date="1980-03-05" name="DateOfDocument" />
```
I.1. Der Beschwerdeführer stellte am 17. Mai 1978 an das Bundesministerium für Landesverteidigung den Antrag um Erteilung einer Ausnahmebewilligung gem. §92 Luftfahrttgesetz, BGBl. 253/1957, in der...
geltenden Fassung (künftig: LFG), zur Errichtung einer Werkstatt mit
Büroräumen auf der ihm gehörigen Parzelle Nr. 1439/15 der KG N. Diese
Parzelle liegt im Sicherheitszonenbereich (rot) der Verordnung des
Bundesministeriums für Landesverteidigung vom 5. Juli 1961 für den
Militärflugplatz Hörsching (Linz), [...]</p>
<p>Mit Bescheid des Bundesministeriums [...]</p>
<p>2. Gegen diesen Bescheid wendet sich die auf Art144 B-VG
gestützte Beschwerde, in der die Verletzung der verfassungsgesetzlich
gewährleisteten Rechte auf Gleichheit aller Staatsbürger vor dem
Gesetz, auf Unverletzlichkeit des Eigentums sowie auf Freiheit der
Gewerbsausübung geltend gemacht und die Aufhebung des Bescheides,
allenfalls die Abtretung der Beschwerde an den VfGH beantragt
wird.</p>
<p>Die belangte Behörde hat die Verwaltungsakten vorgelegt,
eine Gegenschrift erstattet und die Abweisung der Beschwerde
begehrt.</p>
<p>Die Beschwerde war daher abzuweisen.</p>
<p>Die Beschwerde wird abgewiesen.</p>

5.2 Plain Text Conversion

The conversion into plain text read as follow:

Luftfahrtgesetz; keine Bedenken gegen die §§85, 86 und 92 sowie gegen
die Sicherheitszonenverordnung vom 5. Juli 1961 für den
Militärflugplatz Hörsching (Linz); keine denkunmögliche und keine
gleichheitswidrige Anwendung

begründung

I.1. Der Beschwerdeführer stellte am 17. Mai 1978 an das Bundesministerium
für Landesverteidigung den Antrag um Erteilung einer Ausnahmebewilligung


### 5.3 Annotated Output

The annotation results can be reviewed directly inside the Annotation Editor (see Figure 4) or with the UIMA annotation viewer (see Figure 6) if the pipeline was executed through the command line script. Both reveal feedback that can be used for the fine-tuning of the pipeline.

![Figure 6 - UIMA Annotation Viewer.](image)

As mentioned before, in the EUCases project the output is the Akomo Ntosó legal XML file including the discovered annotations. A new pipeline component for the AEP was developed to accomplish this function (see XmlAnnotatorAE on page 13). The XML document is abbreviated like above. The inserted annotations are highlighted in blue color. Some explanations follow below.
### 5.3.1 Inline Annotations

In particular, the following types of annotations are inserted into the Akomo Ntoso XML format:

<table>
<thead>
<tr>
<th>Type</th>
<th>Annotation Example</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td><code>&lt;date date=&quot;1978-05-17&quot;&gt;</code></td>
<td>Date recognition and normalization</td>
</tr>
<tr>
<td>concept</td>
<td><code>&lt;concept eId=&quot;5116&quot; refersTo=&quot;/eucases/eurovoc/5116/&quot; title=&quot;ministry&quot;&gt;</code></td>
<td>Concepts found in the text are annotated with their ID in the source vocabulary, a reference in the source system and the normalized preferred term (in English)</td>
</tr>
<tr>
<td>location</td>
<td><code>&lt;location eId=&quot;2772400&quot; refersTo=&quot;/geonames.org/2772400/&quot; title=&quot;Linz&quot;&gt;</code></td>
<td>Locations and places are annotated together with a unique ID in the reference system</td>
</tr>
</tbody>
</table>

Accordingly, the result of the annotation process for the given example looks like follows.

```xml
<?xml version="1.0"?>
<akomaNtoso xmlns:euc="http://eucases/prop"
xmlns="http://docs.oasis-open.org/legaldocml/ns/akn/3.0/CSD10">
<judgment name="doc_judgment">
<meta>
[...]
<otherAnalysis eId="Leitsatz" source="RIS">
<p><concept eId="1697" refersTo="/eucases/eurovoc/1697/" title="Legal rule">Luftfahrtgesetz</concept>; keine Bedenken gegen die §§85, 86 und 92 sowie gegen</p>
<concept eId="3099" refersTo="/eucases/eurovoc/3099/" title = "Sicherheitszonenverordnung" vom <date date="1961-07-05">5. Juli 1961</date> für den</concept>
<concept eId="195" refersTo="/eucases/eurovoc/195/" title = "airport" >Militärflugplatz Hörsching (Linz)</concept> ); keine denkmögliche und keine</p>
<references source="document">
<TLCReference GUID="dr_1" href="" name="NatLegCore" showAs="B-VG Art7 Abs1 / Verwaltungsakt"/>
<TLCReference GUID="dr_2" href="" name="NatLegCore" showAs="LuftFG §58"/>
<TLCReference GUID="dr_3" href="" name="NatLegCore" showAs="LuftFG §86"/>
<TLCReference GUID="dr_4" href="" name="NatLegCore" showAs="LuftFG §92"/>
<TLCReference GUID="dr_5" href="" name="NatLegCore" showAs="SicherheitszonenV des BMfLV vom 05.07.61 für den Militärflugplatz Hörsching (Linz)"/>
</references>
</otherAnalysis>
</judgment>
<references source="document">
</references>
</akomaNtoso>
```

Die Verletzung eines anderen verfassungsgesetzlich gewährleisteten Rechtes hat das Verfahren ebensowenig ergeben wie die Anwendung einer rechtswidrigen generellen Norm.

Die Beschwerde war daher abzuweisen.
Die Beschwerde wird abgewiesen.

5.3.2 Explanation

The annotations are explained by the help of a few examples:

- `<date date="1961-07-05">5. Juli 1961</date>`
  Here, a date was recognized by the regular expression annotator. The normalized form is placed into the type-attribute.

- `<concept eId="5116" refersTo="eucases/eurovoc/5116/" title= "ministry">Bundesministerium</concept>`
  This annotation references the concept with id 5116 of the EuroVoc dictionary. The preferred term is ministry.

- `<concept eId="na446" refersTo="eucases/syllabus_2_0/na446/" title = "Antrag">Antrag</concept>`
  Similarly, the word Antrag is connected to the concept na446 of the Legal Taxonomy Syllabus. The preferred term is Antrag as well.

- `<location eId="2772400" refersTo="geonames.org/2772400/" title = "Linz">Linz</location>`
  Finally, this is an example for the Geonames dictionary. The Geonames-id of Linz is 2772400.