Originally published in: Proceedings of Joint CLARIN-D/DARIAH Workshop "Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts" at Digital Humanities Conference 2012. – Tübingen: CLARIN-D, 2012. Pp. 17-23.

Integration of WebLicht into the CLARIN Infrastructure

Emanuel Dima, Erhard Hinrichs, Marie Hinrichs, Alexander Kislev, Thorsten Trippel, Thomas Zastrow

University of Tübingen
Seminar für Sprachwissenschaft
firstname.lastname@uni-tuebingen.de

Abstract

This paper describes the ongoing work to integrate WebLicht into the CLARIN infrastructure. It introduces the CLARIN infrastructure for scholars in the humanities and social sciences as well as WebLicht - an orchestration and execution environment that is built upon Service Oriented Architecture principles. The integration of WebLicht into the CLARIN infrastructure involves adapting it to the standards and practices used within CLARIN, including distributed repositories, CMDI metadata, and persistent identifiers.

Keywords: WebLicht, CLARIN, SOA

1. Motivation

The ESFRI project Common Language Resources and Technology Infrastructure (CLARIN)[1] has the goal of providing an integrated and interoperable research infrastructure for scholars in the humanities and social sciences. One of the main challenges in constructing this CLARIN infrastructure lies in the integration of language data and of services that allow scholars to search, annotate, enrich, and visualize language data on a large scale. CLARIN aims to make such language processing tools and services available for novice users, eliminating technical obstacles that are often encountered. In addition, these tools and services should be capable of operating on data from various sources and should be combinable into complex processing workflows.

2. CLARIN Metadata

An important part of the CLARIN infrastructure involves creating language processing tools and making them available to the larger academic community. Equally important, however, is devising a framework which allows these tools to be discovered and used with minimal effort on the end-user's part. This framework relies heavily on the use of a common metadata format. This section describes the metadata within CLARIN which enables WebLicht (Henrich et al., 2010) to discover language tools and create processing chains for annotating text corpora. Section 3. describes in detail how WebLicht will use CLARIN's metadata for building processing chains.

2.1. Persistent Identifiers

A persistent identifier (PID) is a unique character string used to identify a digital object (ISO 24619:2011, 2011). PIDs can be used as references to digital objects in a wide range of documents. The URL of the actual digital object can be obtained simply by resolving its PID by means of a designated service. PIDs are widely applied within the CLARIN metadata to refer to webservices, resources, etc.

[1]CLARIN: `http://www.clarin.eu`

2.2. CMDI

The CLARIN project utilizes the CMDI[2] (Component Metadata Infrastructure) metadata format, which provides a unified library of building blocks (called components). These components can be reused to compose a wide range of increasingly complex metadata components. Each of the elementary components is normally linked to a field descriptor in a registry of concepts and terminology widely used by the linguistic community, such as ISOcat[3] (ISO 12620:2009, 2009). The CLARIN infrastructure provides the CMDI Component Registry[4], in which arbitrarily complex metadata components can be added for any resource type, reusing existing components (Broeder et al., 2010). The advantage of this building-block approach is that it imposes no limitations on the expressiveness of the metadata. CLARIN requires that every resource is described with CMDI metadata, and that this metadata is assigned a PID and made available for harvesting via the OAI-PMH protocol.

2.3. Center Resource Repositories

Each individual CLARIN center will operate its own repository for storing its resources and metadata. Currently there are nine CLARIN centers, located at academic and research facilities. When a center wishes to inform the wider CLARIN community about one of its resources, it makes the corresponding metadata (and optionally the resource itself) publicly available in its repository. This metadata can be harvested via the standard OAI-PMH protocol, enabling integration of the centers' tools and resources into a broad range of applications. For example, WebLicht must employ a harvester to gather CMDI metadata about available WebLicht-compatible webservices.

2.4. CLARIN Center Registry

In order to ease the discoverability of resources within the CLARIN infrastructure, the center registry has been put in

[2]CMDI: `http://www.clarin.eu/cmdi`
[3]ISOcat: `http://www.isocat.org`
[4]CMDI Component Registry: `http://catalog.clarin.eu/ds/ComponentRegistry/#`

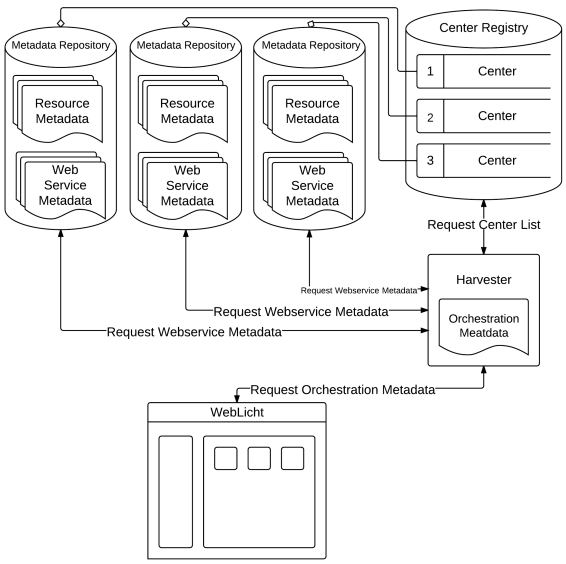

Figure 1: CLARIN Metadata in WebLicht

place. The CLARIN Center Registry stores CMDI metadata describing each of the nine CLARIN centers. This metadata includes the location of each center's repository, as well as the center's type (A, B, or C), contact information, CLARIN compliance status, etc. Figure 1 shows how the center registry, together with the center repositories, will be utilized by WebLicht to collect webservice metadata. An integral part of this webservice metadata is so-called orchestration metadata, which specifies detailed input and output descriptions as well as webservice parameters (see section 3.3.). This orchestration data is the key component for constructing workflows in WebLicht.

3. WebLicht

WebLicht is a service orchestration and execution environment built upon Service Oriented Architecture principles. Although it is currently mostly used for incremental automatic annotation of text corpora, other uses such as audio data processing are under development. In order to fulfill the integration requirements the following components are necessary:

- A harvester for aggregating metadata about distributed webservices
- An orchestration engine for building valid webservice workflows
- A user-friendly web application for construction and invocation of workflows, which relies on the harvester and the orchestration engine

The orchestration engine and the web application are already a part of the current version of WebLicht, while the harvester is presently under development.

3.1. Webservices

A WebLicht-compatible webservice is simply a synchronous REST-style webservice. The client establishes an HTTP connection to the webservice and initiates a POST request containing the input data. Depending on the webservice, the client can also set various processing parameters using the query string in the URL. The webservice processes the input data synchronously and returns the result as output data, using the same HTTP connection. Currently there are approximately one hundred WebLicht-compatible webservices located across Europe. The data format used by the majority of WebLicht-compatible webservices is a valid XML format, TCF (Text Corpus Format), fully compatible with the Linguistic Annotation Format (LAF) and Graph-based Format for Linguistic Annotations (GrAF) developed in the ISO/TC37/SC4 technical committee (Ide and Suderman, 2007). A set of webservices allow conversions between TCF and most of the other formats used in the linguistic community.

3.2. Harvesting of Webservice Metadata

Since webservice metadata is distributed over all the CLARIN center resource repositories, there is no central location from which to obtain webservice metadata. This necessitates the collection of webservice metadata (called harvesting) from each resource repository. Using information from the CLARIN center registry, a harvester is able to target a specific metadata set contained in the resource repositories. Based on given requirements, the harvester will locate the appropriate sets within each center resource repository, read the sets via the OAI-PMH protocol, and aggregate the results.

For example, the WebLicht harvester must periodically check all the center repositories for WebLicht-compatible webservice metadata sets and cache them, as shown in Figure 1. With the help of the harvested metadata, WebLicht can assist the user in building webservice workflows in a type-safe manner. A webservice workflow is a sequence of webservices that are executed in succession, each one receiving as input the output of the previous one. Using the descriptions of the input and output in the metadata for each webservice, the system can guarantee the correctness of a webservice workflow from a data perspective.

3.3. Orchestration Metadata and Chaining

Each webservice is described in the CLARIN repositories by a piece of metadata. This contains information about the webservice creators, access rights, development status, description, input and output format specifications, registration date, URL, etc. An important part of the metadata is the so-called orchestration metadata, which is used by orchestration engines to match webservices to a given input data and to create workflows.

In the CLARIN CMDI Component Registry, an extendable core webservice metadata format is specified. By design, this core format can be easily extended to describe a wide variety of webservices. Such an extended format describing WebLicht-compatible webservices is registered in the CMDI Component Registry, which elaborates on the orchestration metadata in the core format. The orchestration metadata in this extended format will be referred to in this paper as *orchestration metadata*. The purpose of this orchestration metadata is to describe the profile of the data

that a webservice accepts and the profile of its output, implicitly defining a type system. This I/O data specification is designed to be as straightforward and generic as possible. It is domain neutral and independent of the actual data format used by the webservices, but at the same time simple to understand, use and generate. The orchestration metadata is a list consisting primarily of data features, each feature containing zero or more values. The input description specifies what properties input data must have in order to be correctly processed by the webservice. The output description specifies properties of the output data that is generated by the webservice.

```
<Input>
   <ParameterGroup>
      <Name>Input Parameters</Name>
      <Parameters>
         <Parameter>
            <Name>type</Name>
            <WebServiceArgValue>informat</WebServiceArgValue>
            <AllowManualSelectionFallback>true</AllowManualSelectionFallback>
            <Values>
               <ParameterValue>
                  <Value>text/plain</Value>
                  <WebServiceArgValue>plaintext</WebServiceArgValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>application/msword</Value>
                  <WebServiceArgValue>doc</WebServiceArgValue>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>lang</Name>
            <WebServiceArgValue>language</WebServiceArgValue>
            <AllowManualSelectionFallback>true</AllowManualSelectionFallback>
            <Values>
               <ParameterValue>
                  <Value>de</Value>
                  <WebServiceArgValue>de</WebServiceArgValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>en</Value>
                  <WebServiceArgValue>en</WebServiceArgValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>es</Value>
                  <WebServiceArgValue>es</WebServiceArgValue>
               </ParameterValue>
            </Values>
         </Parameter>
      </Parameters>
   </ParameterGroup>
</Input>
<Output>
   <ParameterGroup>
      <Name>Output Parameters</Name>
      <ReplacesInput>true</ReplacesInput>
      <Paremeters>
         <Parameter>
            <Name>type</Name>
            <Values>
               <ParameterValue>
                  <Value>text/tcf+xml</Value>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>lang</Name>
            <RefInputParameter>lang</RefInputParameter>
            <Values>
               <ParameterValue>
                  <Value>de</Value>
                  <RefInputParameterValue>de</RefInputParameterValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>en</Value>
                  <RefInputParameterValue>en</RefInputParameterValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>es</Value>
                  <RefInputParameterValue>es</RefInputParameterValue>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>version</Name>
            <Values>
               <ParameterValue>
                  <Value>0.4</Value>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>text</Name>
         </Parameter>
      </Parameters>
   </ParameterGroup>
</Output>
```

Listing 1: CMDI Orchestration Metadata Example 1

For example, consider the simple case of a tool which takes a text document as input and produces a TCF document (which is a widely used internal format for linguistic processing within WebLicht) as output. Listing 1 shows the input and output specifications for such a tool, extracted from the CMDI orchestration metadata.

The CMDI metadata specifies that the input of the corresponding webservice must be either a plain-text or a Microsoft Word document. In order to better understand how this is achieved, consider the summary of input features, which is shown in Table 1.

Feature	Value(s)	URL Query String
type	application/msword	informat=doc
	text/plain	informat=plaintext
lang	de	language=de
	en	language=en
	es	language=es

Table 1: Input Specification Summary for Example 1

The input specification asserts that the ”type” feature of the input data must be ”application/msword” or ”text/plain”. The ”type” feature in this case designates the data mime type. Similarly, the ”lang” feature must be either ”de”, ”en”, or ”es”, which indicates the language of the input document. Additionally it can be seen that a query string is assigned to each feature value which will be passed to the tool as part of the URL.

The CMDI metadata in Listing 1 also specifies that the output will be a TCF file with version 0.4, a language identical to the language of the input text document, and a text layer (represented in the TCF file as a special node). Table 2 shows the shows the summary of output features for our example.

Feature	Value(s)	Input Reference
type	text/tcf+xml	
version	0.4	
text		
lang	de	lang=de
	en	lang=en
	es	lang=es

Table 2: Output Specification Summary for Example 1

The output specification declares that the value of the ”type” feature is ”text/tcf+xml”, meaning the webservice output format is TCF. The ”version” feature has a value of ”0.4”, which means that version 0.4 of TCF file format will be produced. The ”text” feature without a value indicates that the resulting TCF file will contain a text layer (the notion of layers is specific to TCF). Finally, the ”lang” feature has three possible values, each referencing a value of the ”lang” feature in the input specification. Thus the language

of the output TCF document will depend on the language of the input text document.

The output specification can operate in two different modes: addition or replacement. Using the addition mode means that the declared output features for the webservice are added to the ones already in the input data. This mode is used when webservices simply augment the original input data, such as in a case where an additional annotation layer is added to an input TCF document.

In the example above (Listing 1) the replacement mode is used as declared by the element ReplacesInput. A value of ”true” indicates that the webservice completely transforms the input data, so that the profile of the output data is determined solely by the declared output features of the webservice.

Listing 2 shows another example, illustrating the use of the addition mode. It describes a part-of-speech tagging tool which augments the input TCF document with a part-of-speech annotation layer.

```
<Input>
   <ParameterGroup>
      <Name>Input Parameters</Name>
      <Parameters>
         <Parameter>
            <Name>lang</Name>
            <AllowManualSelectionFallback>false</AllowManualSelectionFallback>
            <Values>
               <ParameterValue>
                  <Value>de</Value>
               </ParameterValue>
               <ParameterValue>
                  <Value>en</Value>
               </ParameterValue>
               <ParameterValue>
                  <Value>fr</Value>
               </ParameterValue>
               <ParameterValue>
                  <Value>it</Value>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>tokens</Name>
            <AllowManualSelectionFallback>false</AllowManualSelectionFallback>
         </Parameter>
         <Parameter>
            <Name>type</Name>
            <AllowManualSelectionFallback>false</AllowManualSelectionFallback>
            <Values>
               <ParameterValue>
                  <Value>text/tcf+xml</Value>
               </ParameterValue>
            </Values>
         </Parameter>
         <Parameter>
            <Name>version</Name>
            <AllowManualSelectionFallback>false</AllowManualSelectionFallback>
            <Values>
               <ParameterValue>
                  <Value>0.4</Value>
               </ParameterValue>
            </Values>
         </Parameter>
      </Parameters>
   </ParameterGroup>
</Input>
<Output>
   <ParameterGroup>
      <Name>Output Parameters</Name>
      <ReplacesInput>false</ReplacesInput>
      <Parameters>
         <Parameter>
            <Name>lemmas</Name>
         </Parameter>
         <Parameter>
            <Name>postags.tagset</Name>
            <RefInputParameter>lang</RefInputParameter>
            <Values>
               <ParameterValue>
                  <Value>penntb</Value>
                  <RefInputParameterValue>en</RefInputParameterValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>stein</Value>
                  <RefInputParameterValue>fr</RefInputParameterValue>
                  <RefInputParameterValue>it</RefInputParameterValue>
               </ParameterValue>
               <ParameterValue>
                  <Value>stts</Value>
                  <RefInputParameterValue>de</RefInputParameterValue>
               </ParameterValue>
            </Values>
         </Parameter>
      </Parameters>
   </ParameterGroup>
</Output>
```

Listing 2: CMDI Orchestration Metadata Example 2

The input specification for this example is summarized in Table 3. In particular, it tells us that the input format must be a TCF document (version 0.4) in one of four languages (de, en, fr, or it), containing a *tokens* annotation layer. In this case, no URL query string is required by the webservice.

Feature	Value(s)	URL Query String
type	text/tcf+xml	
version	0.4	
tokens		
lang	de en fr it	

Table 3: Input Specification Summary for Example 2

Since the ReplacesInput element has a value of ”false”, the output specification for this tool, summarized in Table 4, must not explicitly specify the type, version, or language of the output document, as all the input features will be retained in the output. Thus only the additional annotation layers must be specified. In this case, *lemmas* and *postags.tagset* layers are added, where the value of the *postags.tagset* feature depends on the input language. Namely, if the input document is in English the *penntb tagset* will be used for part-of-speech tagging, whereas the *stein tagset* will be used for French and Italian documents, and the *stts tagset* will be used for German documents. This example illustrates the use of input references to create complex output/input dependencies.

Feature	Value(s)	Input Reference
lemmas		
postags.tagset	penntb stein stts	lang=en lang={fr, it} lang=de

Table 4: Output Specification Summary for Example 2

The feature and value name semantics are not yet formally specified. However, consistency of the feature names is necessary for interoperability of webservices within the same domain. In order to ensure this consistency, the feature and value names must be kept in a concept registry such as ISOcat, which is widely used in the CLARIN infrastructure. The use of these standardized feature and value names will then be applied to currently registered WebLicht-compatible webservices and enforced when registering a new webservice.

This system of data-describing assertions makes chaining possible (even before any webservices have actually been invoked). Chaining is the process of automatically find-

ing appropriate webservices when the characteristics of the data to be used as input are known. Each time a webservice is selected, the characteristics of the resulting data are recomputed and used to generate a new set of suitable webservices. Automatically finding new webservices to be applied in the chain is now just a matter of filtering through the list of the available webservices.

3.4. Example Processing Chains

The WebLicht architecture allows a high degree of flexibility with respect to webservice processing chains. For example, Figure 2 shows a small subset of possible processing chains available for the German language. Each circle represents an individual webservice (labeled with the institute which created it and an identification number) and the rectangular boxes represent linguistic annotation states. An annotation state indicates which linguistic annotation layers are present in the data document. Often it is possible to reach an annotation state through different processing chains. This allows users to compare results from different tools and to pick-and-choose those tools that they deem best suited to their needs.
Figure 3 shows the connectivity of all of the webservices currently available in WebLicht for the German language.

4. Conclusion

We presented the requirements of integrating WebLicht into the CLARIN infrastructure and described the ongoing work to fulfill them, including harvesting CMDI metadata from the CLARIN center registry and the distributed center resource repositories. In addition, the importance and utility of the CMDI metadata within WebLicht was described.
Once integration is complete, WebLicht-compatible webservices will be easily discoverable and thus the entire CLARIN developer community will be able to utilize them (also directly - without using the WebLicht web interface). As for users, they will benefit from immediate access to new tools as soon as they are registered in any CLARIN center.

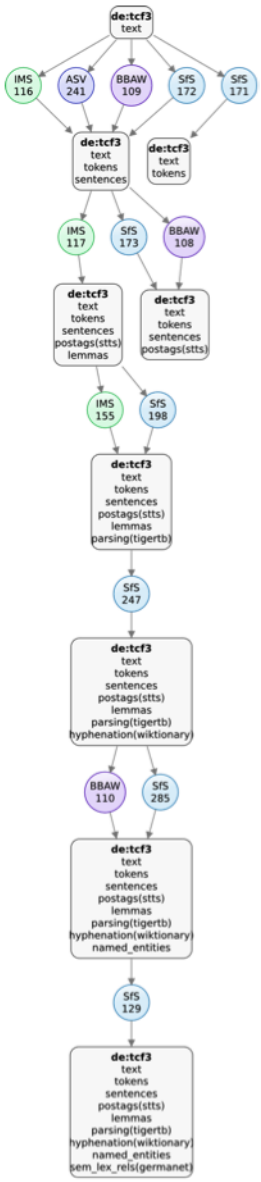

Figure 2: Subset of Webservices Available for German

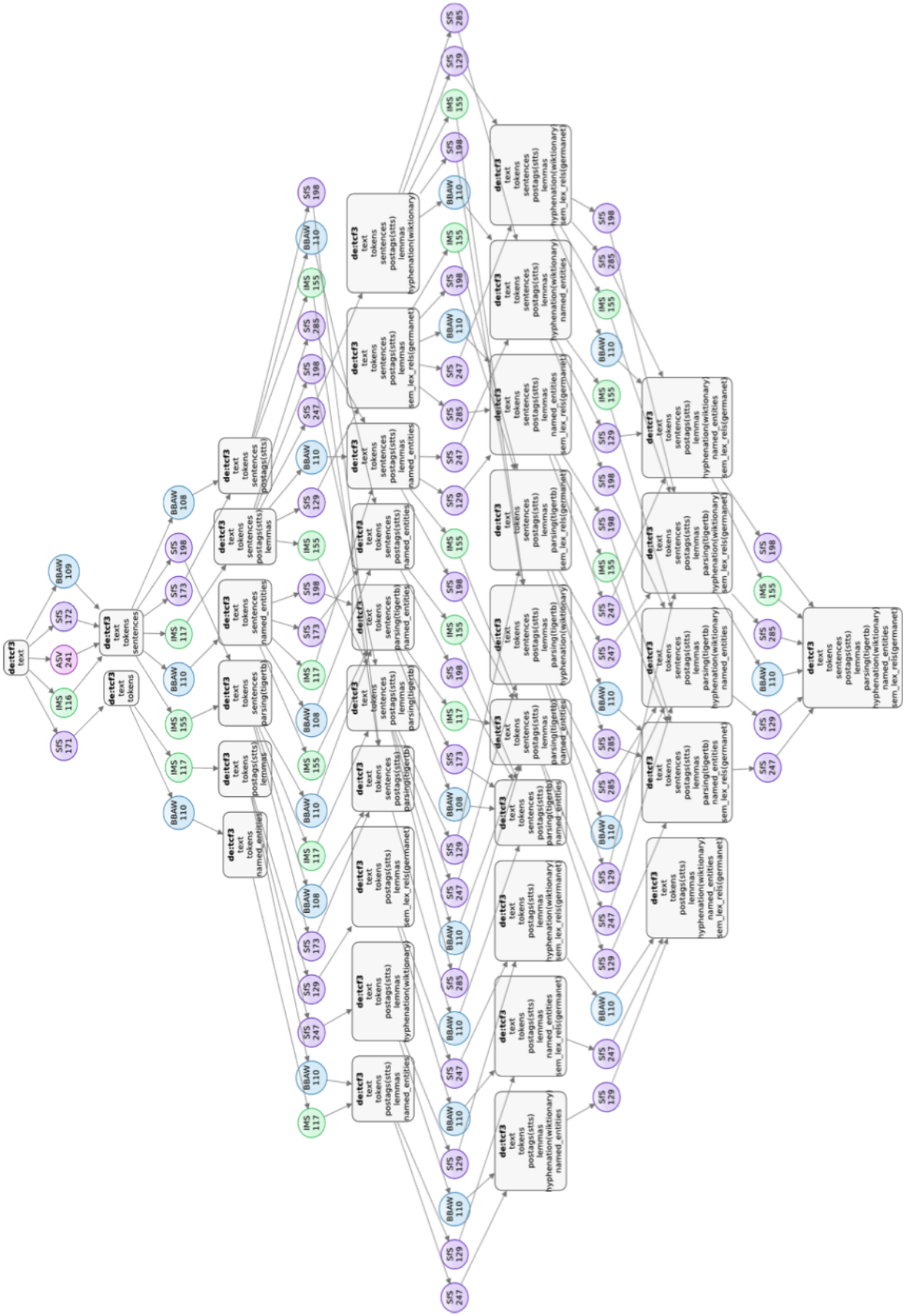

Figure 3: Complete Set of Webservices for German

5. References

D. Broeder, M. Kemps-Snijders, D. Van Uytvanck, M. Windhouwer, P. Withers, P. Wittenburg, and C. Zinn. 2010. A data category registry and component-based metadata framework. In *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC 2010)*, Malta.

V. Henrich, E. Hinrichs, M. Hinrichs, and T. Zastrow. 2010. Service-oriented architectures: From desktop tools to web services and web applications. In Dan Tufiş and Corina Forăscu, editors, *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, pages 69–92. Romanian Academy Publishing House, Bucharest, Romania.

N. Ide and K. Suderman. 2007. Graf: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop, held in conjunction with ACL*, Prague, Czech Republic.

ISO 12620:2009. 2009. Terminology and other language and content resources – specification of data categories and management of a data category registry for language resource. Technical report, ISO.

ISO 24619:2011. 2011. Language resource management – persistent identification and sustainable access (PISA). Technical report, ISO.