

CLARIN-D User Guide

Written by CLARIN-D AP 5

CLARIN-D User Guide

Written by CLARIN-D AP 5

Version: 1.0.1 (also available as a PDF file [<http://media.dwds.de/clarin/userguide/userguide-1.0.1.pdf>])

Publication date Publication date: 2012-12-19

Abstract

Der CLARIN-D User Guide ist ein praktischer Leitfaden für die Anpassung und Integration existierender Sprachressourcen in die CLARIN-D-Infrastruktur. Er behandelt in seinem ersten Teil grundlegende Themen und Modellierungsprinzipien, die auf alle Arten von linguistischen Ressourcen und Werkzeugen anwendbar sind (Datenkategorien, Metadaten, Annotationen, rechtliche Aspekte, Qualitätsmanagement). Im zweiten Teil des User Guides werden ressourcenspezifische Eigenschaften der Infrastruktur dargestellt (Korpora, lexikalische Ressourcen, linguistische Werkzeuge und deren Verknüpfung). Hierbei wird besonderes Augenmerk auf die Beschreibung technischer Spezifika gelegt.

This user guide serves as a comprehensive overview on the CLARIN-D infrastructure. It describes prerequisites for and methods of integrating linguistic tools and resources. Part I, “Basic concepts” provides readers with background information and fundamental principles of the CLARIN-D infrastructure that apply for all types of linguistic resources and tools. Part II, “Linguistic resources and tools” covers specific issues for a broad range of resources and tools with strong emphasis on best practices for the technical aspects of their integration and interoperability.

Licensed under Creative Commons, CC BY-ND 3.0 DE [<http://creativecommons.org/licenses/by-nd/3.0/de/>]

Table of Contents

Introduction and background	iv
1. About this book	v
2. Who should read this book?	v
3. How to use this book	vi
4. Release history	vi
I. Basic concepts	1
1. Concepts and data categories	4
1. Data Categories and Data Category Registries	4
2. ISOcat, a Data Category Registry	5
2. Metadata	13
1. Managing and Accessing Data	13
2. Objects, Collections, Granularity	14
3. Types of Resources and Metadata Components	16
4. Lifecycle Management	17
5. Existing MD sets	17
6. The Component Metadata Initiative (CMDI)	19
7. Aggregation	26
8. Recommendations	29
3. Resource annotations	30
1. Aspects of annotations	31
2. Exchange and combination of annotations	36
3. Recommendations	41
4. Access to resources and tools – technical and legal issues	43
1. Single Sign-on access to the CLARIN-D infrastructure	43
2. Legal Issues	45
5. Quality assurance	46
1. Aspects of the quality of resources	46
2. Recommendations	47
II. Linguistic resources and tools	49
6. Types of resources	51
1. General recommendations	51
2. Text Corpora	52
3. Multimodal corpora	61
4. Lexical resources	65
7. Linguistic tools	73
1. Hierarchies of linguistic tools	73
2. Automatic and manual analysis tools	75
3. Technical issues in linguistic tool management	77
4. Automatic segmentation and annotation tools	77
5. Manual annotation and analysis tools	92
6. Multimedia tools	93
7. Recommendations for CLARIN-D tool designers	95
8. Web services: Accessing and using linguistic tools	96
1. Web Services	96
2. Service-oriented architectures	98
3. WebLicht – A service-oriented architecture for linguistic resources and tools	98
4. WebLicht usage scenarios	102
5. Integrating existing linguistic tools into WebLicht	114
Bibliography	117

Introduction and background

Axel Herold, Lothar Lemnitzer, BBAW Berlin

The BMBF funded project CLARIN-D will develop a digital infrastructure for language-centered research in the humanities and social sciences. The main function of the CLARIN-D service centres will be to provide relevant, useful data and tools in an integrated, interoperable and scalable way. CLARIN-D will roll this infrastructure out in close collaboration with expert scholars in the humanities and social sciences, to ensure that it meets the needs of users in a systematic and easily accessible way.

CLARIN-D is building on the achievements of the preparatory phase of the European CLARIN initiative [<http://www.clarin.eu>] as well as CLARIN-D's Germany-specific predecessor project D-SPIN [<http://d-spin.org>]. These previous projects have developed standards to be met by the CLARIN-D services centres, technical standards and solutions for key functions, a set of requirements which participants have to provide, as well as plans for the sustainable provision of tools and data and their long-term archiving.

The outcomes of these projects which are relevant and accessible to the user of the CLARIN-D infrastructure are:

- WebLicht, an environment for the orchestration of linguistic tools and the sequential annotation of primary data. If you want to learn more about Weblicht, please consult Section 3, “WebLicht – A service-oriented architecture for linguistic resources and tools” or log in to WebLicht [<https://weblicht.sfs.uni-tuebingen.de/>] directly if you signed up for an account.
- The resource inventory of the Virtual Language Observatory [<http://catalog.clarin.eu/ds/vlo/>] (VLO). You can use the VLO to browse and search through many language resources and check their availability and other metadata.
- The CLARIN-D helpdesks [<http://de.clarin.eu/en/training-helpdesk.html>]. We will refer you to these helpdesks for topics and issues which are too specific for this general user guide. There are always questions which will remain open. If you have one, please do not hesitate to contact the most appropriate helpdesk.

One of the major tasks of CLARIN-D will be the maintenance and hosting of resources and tools, ours as well as yours. Indeed, the backbone of the CLARIN-D infrastructure is a network of resource centres which offer services to this end (e.g. repositories and persistent identifiers for resources and tools, expertise in language technology standards and questions regarding interoperability). If you want to contribute to CLARIN-D with your resource(s) and/or tool(s), it is a good idea to get in touch with the CLARIN-D center which is nearest to your site. You will find a list of CLARIN-D service centres [<http://de.clarin.eu/en/clarin-d-centres.html>] on the CLARIN-D website.

If you want to become an active part of the wider CLARIN-D community because you feel that language resource and tools is something which is or could be relevant for your research, you might consider joining one of the discipline-specific working groups [<http://de.clarin.eu/en/discipline-specific-working-groups.html>] (“Facharbeitsgruppen”, F-AG). You are always welcome to join in.

CLARIN-D and the European CLARIN consortium also organize regular tutorials and workshops that allow you to learn more about the CLARIN-D infrastructure and to get hands-on training using it.

1. About this book

This user guide serves as a comprehensive overview on the CLARIN-D infrastructure. It describes prerequisites for and methods of integrating linguistic tools and resources.

The user guide is on the highest level divided in two parts. The first part introduces basic concept and practices of the CLARIN-D infrastructure at large. If you are new to CLARIN-D, you should read this part first. All types or resources to be integrated are affected by the policies which are outlined in this part of the user guide. Here, we cover the issues related to concepts and data categories, we introduce meta-data in general and the Component Metadata Infrastructure (CMDI) in particular, chapter 4 deals with (mainly linguistic) annotation of primary data, chapter 5 introduces issues of quality and chapter 6 serves as a short introduction on accessibility of resources and rights management.

The second part covers issues which are specific to particular types of resources (e.g. corpora, lexical resources) or to particular tools (e.g. taggers, parsers). It also presents the CLARIN-D way of orchestrating tools in tool chains.

CLARIN-D is an evolving project and the descriptions provided in this user guide cannot be considered as definite and unchangeable. Research requirements as well as new technical developments may well call for new standards and practices and of changes in existing ones. We therefore consider this document to be a living one which will be subject to changes in the future. You will find this document as an electronic one with a version number attached to it. A version tracker will keep you up-to date on latest changes to this document.

Each section of this user guide is concluded by a list of recommendations that briefly summarize the current technical prerequisites for tool and resource integration with respect to the publication date of the user guide.

A central glossary for CLARIN specific terms [<http://www.clarin.eu/external/index.php?page=glossary>.] has been collected on the CLARIN EU website [<http://www.clarin.eu/>] and provides pointers to most of the technologies that are mentioned throughout this user guide.

2. Who should read this book?

Researchers from all disciplines of the humanities will find help and guidance for resource and tool integration. The user guide introduces and motivates the CLARIN-D policy towards these issues. It also provides pointers to existing documentation of common practice in the field.

The procedures described in this user guide also provide developers with methods and metrics to evaluate linguistic tools and resources from a purely technical point of view. Evaluation from a developer's perspective tries to answer the question how difficult or expensive it is to integrate external resources into the CLARIN-D infrastructure. For resources that are developed from scratch this part will also serve as a guideline for minimizing the costs for adaptation and finally integration into the CLARIN-D infrastructure. A major aim of this user guide will be in particular to support developers in achieving interoperability with data formats that are endorsed by CLARIN-D. Besides the two intended audiences sketched above, the editors hope that this user guide might also be of value for researchers external to the national CLARIN consortia, e.g. for funding bodies and their reviewers.

3. How to use this book

If you are new to CLARIN-D, you should at least skim through the chapters of the first part. If you are already experienced in sharing resources within a community, some of the issues will sound familiar to you. You should nevertheless learn about the policy and recommendations of CLARIN-D (so you should at least check the recommendations at the end of each section).

If you want to provide and share a particular resource, you might want to move straight ahead to the relevant section of Chapter 6, *Types of resources*. The same holds for a particular tool – move on to Chapter 7, *Linguistic tools* and the relevant sections. Even more relevant in this case will be Chapter 8, *Web services: Accessing and using linguistic tools*, which explains how tools of different sorts are made interoperable. This chapter is also for you if you look for a solution to the linguistic analysis of your (textual) primary data, i.e. a single document or a corpus.

As the audience of this user guide is diverse we mark sections aiming at specific readers explicitly:



Technical details

These sections contain information aimed specifically at technical staff such as software engineers and developers. Here you will also often find pointers to technical specifications of formats and protocols.



Tips and high level summaries

These sections summarize the most important points discussed throughout the user guide and provide a very compressed view on a specific topic.



Additional information

These sections contain valuable background information many of our readers will already be familiar with.

4. Release history

1.0.1 (2012-12-19, @rev208)

minor typographic corrections, added PDF version

1.0.0 (2012-12-14, @rev205)

first release

Part I. Basic concepts

Table of Contents

1. Concepts and data categories	4
1. Data Categories and Data Category Registries	4
2. ISOcat, a Data Category Registry	5
2.1. Data categories in ISOcat	6
2.2. Data category types: simple, complex and container DCs	6
2.3. Specifying a data category: administrative, descriptive and linguistic part	8
2.4. Ways how to use the ISOcat data category registry	9
2.5. RELcat, a relation registry	11
2. Metadata	13
1. Managing and Accessing Data	13
2. Objects, Collections, Granularity	14
3. Types of Resources and Metadata Components	16
4. Lifecycle Management	17
5. Existing MD sets	17
5.1. Dublin Core Metadata Initiative (DCMI)	18
5.2. ISLE Metadata Initiative	18
5.3. Open Language Archive Community (OLAC)	18
5.4. CHAT	18
5.5. Text Encoding Initiative (TEI)	19
5.6. Component Metadata Initiative (CMDI)	19
6. The Component Metadata Initiative (CMDI)	19
6.1. What is it?	19
6.2. Why yet another metadata format?	20
6.3. The CMDI model	20
6.4. Explicit semantics	23
6.5. Procedure	24
6.6. Profile and component adaptation	25
6.7. Preferred components and profiles	26
6.8. Converting existing metadata to CMDI	26
7. Aggregation	26
7.1. Metadata Harvesting	27
7.2. Metadata gathering and searching: the Virtual Language Observatory (VLO)	27
8. Recommendations	29
3. Resource annotations	30
1. Aspects of annotations	31
1.1. Inline vs. stand-off annotations	31
1.2. Multi-layer annotation	34
1.3. Relations between annotation types	35
2. Exchange and combination of annotations	36
2.1. Representing and exchanging complete annotations: getting independent of a specific representation format	36
2.2. Introduction and monitoring of data categories: relating specific tagsets	40
2.3. Handling different concepts: issues in transferring annotation schemes....	41
3. Recommendations	41
4. Access to resources and tools – technical and legal issues	43
1. Single Sign-on access to the CLARIN-D infrastructure	43
1.1. Gaining access to a resource	44

1.2. Granting access to a resources	44
1.3. Technical details of the single sign-on infrastructure	44
2. Legal Issues	45
5. Quality assurance	46
1. Aspects of the quality of resources	46
1.1. Well-formedness and schema compliance	46
1.2. Adequacy and consistency	46
1.3. Metadata	47
2. Recommendations	47

Chapter 1. Concepts and data categories

Kerstin Eckart, Universität Stuttgart

A distributed infrastructure, such as provided by CLARIN-D, which aims at covering heterogeneous language resources and tools, has to deal with the use of concepts which are similar but not identical when used by different stakeholders. This means that the ranges which these concepts cover overlap but they are not co-extensive. To complicate matters, identical terms might refer to different concepts if used by different parties, leading on the one hand to polysemy of terms, and, on the other hand, to synonymy. Both are not desirable but are not avoidable either in such a broad domain.

Explicit description of key concepts and of terms which signify these concepts is therefore vital for the success of an endeavour such as building a comprehensive infrastructure of language resources and tools. It is a keystone for the interoperability of language resources and tools. It is also necessary to make explicit the relations between the used concepts (e.g. synonymy, generalization / specialization).

A preliminary step to reach this goal is to collect concepts and notions which are used by the stakeholders. A further step is to provide definitions for these concepts and to link them to terms which are commonly used in the communities. It will also be necessary to keep track of concepts which emerge in course of the further development of the infrastructure.

That means that terminology management is a task which will never be finished. As an ongoing task which involves many stakeholders in the field it calls for tools which facilitate the description of categories and concepts and the transparent use of them when describing primary data (through metadata) as well as linguistic annotations and their descriptions.

In this chapter we introduce a *Data Category Registry* (DCR) to keep track of existing and new concepts, and discuss ISOcat [<http://www.isocat.org/>], the Data Category Registry utilized within CLARIN-D, in detail and mainly from a (technical) users perspective.

1. Data Categories and Data Category Registries

Descriptive terms, that signify linguistic concepts, are used in the description of resources, e.g. linguistic tools or linguistically annotated data. To give a simple example: if a corpus is described by a set of metadata, one important information to be conveyed to the prospective user is the object language(s) which this corpus covers. This information can be expressed in different manners, e.g. by using the name of the object language(s) in the language in which the metadata are provided:

Example 1.1. Unformalized data category value

Language: German

or by using a so-called language code, such as the ones provided by [ISO 639-3:2007]. Also the name of the descriptive category label itself may vary between “language”, “language name”, “Sprache”, “langue”, etc.:

Example 1.2. Formalized data category value

Language: deu

Sprache: deu

The point is that all the above descriptions have the same meaning: the object language of this resource is German. The statement has two parts: an attribute and a value that is assigned to this attribute. Both the attribute and its value, as well as many other attributes and values, are called data categories. To build an infrastructure of interoperable resources, it is necessary to keep track of all these categories in a category registry and to make an explicit reference to these categories by referring to them by unique category identifiers. This suffices for all the data categories which are already available. For those data categories which are used by a data provider and which are not yet registered, the data provider should be responsible to provide a new entry to the registry, providing information which is sufficient for other users to understand the meaning of this category.

For our example above, the fact that “Sprache” and “Language” mean the same in the given context can be expressed by providing a reference to a data category by an identifier which is the same in both cases. Concerning the value of this label, it is always preferable to resort to an existing standard, i.e., [ISO 639-3:2007] in that case.

This method of applying category names should also be used for information about the labels used for the (linguistic) annotation of primary data, e.g. to linguistic categories which give information about the part-of-speech of a linguistic unit which is part of the resource. Explicit reference to a data category and its description is helpful also in this case. To give a simple example: according to a linguistic framework which guides the part-of-speech annotation of a resource, the category *noun* might either refer to a concept which contains common nouns only or to a concept which contains both common nouns and proper nouns.

Terminology management in the domain of language resources and linguistic annotations is covered by the ISO standard [ISO 12620:2009]. See also the section on DCR on the CLARIN standards guidance web page [<http://clarin.ids-mannheim.de/standards/>] for information on the ISO document and relations to other standards. It is CLARIN-D policy to follow [ISO 12620:2009]. The standard is embodied by the terminology management platform ISOcat [<http://www.isocat.org/>]. It is CLARIN-D policy to organize terminology management through this platform.

2. ISOcat, a Data Category Registry

ISOcat is the reference implementation of ISO standard 12620 mentioned above. Its mission is defined as follows:

ISO 12620 provides a framework for defining data categories compliant with the ISO/IEC 11179 family of standards. According to this model, each data category is assigned a unique administrative identifier, together with information on the status or decision-making process associated with the data category. In addition, data category specifications in the DCR contain linguistic descriptions, such as data category definitions, statements of associated value domains, and examples. Data category specifications can be associated with a variety of data element names and with language-specific versions of definitions, names, value domains and other attributes.

The use of ISOcat is endorsed by CLARIN-D. It is a core facility for the terminology management in the CLARIN-D infrastructure. This is also planned for its upcoming spin-offs RELcat and SCHEMACat. In the following we give a description of some features of ISOcat. The presentation is largely based on a tutorial by Menzo Windhouwer (MPI, Nijmegen) and Ineke Schuurman (KU Leuven and University of Utrecht). While the first sections describe some details from the technical user's perspective, the last section sums up ways how to use ISOcat. For more information, help pages and tutorial material see the ISOcat manual [<http://www.isocat.org/files/manual.html>] and <http://www.clarin.eu/faq/3507>

2.1. Data categories in ISOcat

In the context of ISOcat a data category (short: DC) is an elementary descriptor in a linguistic structure or an annotation scheme. Its specification comprises three parts: an *administrative part* for administration and identification of the DC, a *descriptive part* for documentation which can also be written in various working languages and a *linguistic part* describing the conceptual domain(s). The UML class diagram of the data model for the specification can be found on the ISOcat website [<http://www.isocat.org/files/12620.html>].

In the following we will give an example for a DC specification, but before we will have a look at the different types of data categories differing by their conceptual domains. There are three main types of DCs: complex, simple and container DCs.

2.2. Data category types: simple, complex and container DCs

Simple DCs are atomic and can neither contain other DCs nor be assigned a value. For example, to represent *neuter*, *masculine* and *feminine* (as possible values for grammatical gender) in ISOcat, for each of them a simple DC is needed.

Complex DCs can be assigned a value and appear with three characteristics: open, closed and constrained.

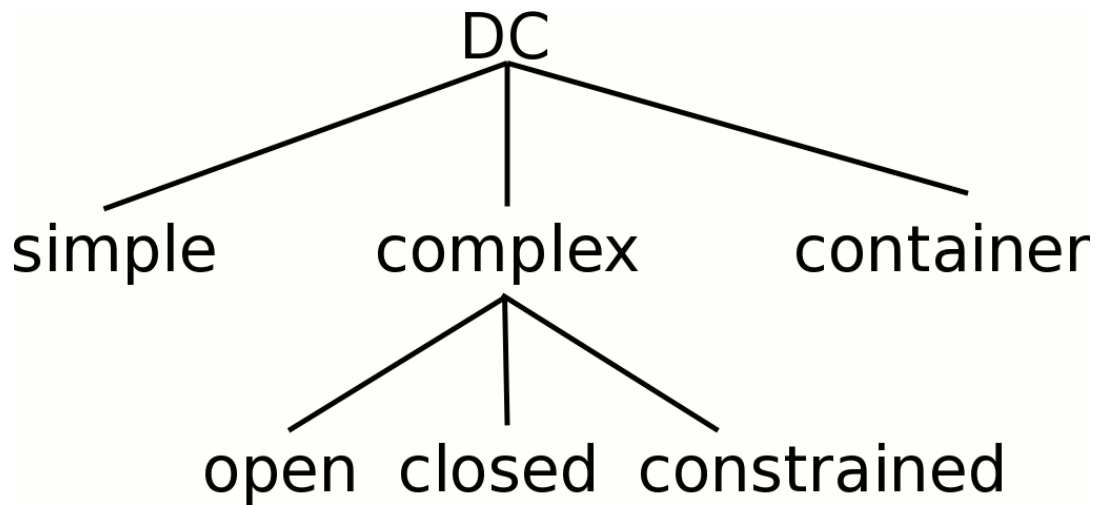
The values assigned to complex open DCs are arbitrary within a chosen data type e.g., the complex open DC *lemma* can be assigned values of type string.

The values assigned to complex closed DCs are part of a closed vocabulary consisting of simple DCs, e.g., *grammatical gender* would be a complex closed DC if the simple DCs *feminine*, *masculine* and *neuter* are in its conceptual domain.

Values assigned to constrained DCs need to fulfill certain constraints, e.g., the complex constrained DC *email* might restrict its values to strings containing an @ character.

Figure 1.1, “DC Types” provides an overview of the data category types.

Figure 1.1. DC Types



Container DCs would in principle include other DCs (simple, complex and container). For example, a container category *lexicon* might include another container *lemma* which again includes the complex open DC *writtenForm* and the container *lexicon* might also include the complex closed DC *language*. See Figure 1.2, “Examples for DCs of several types” where container DCs are orange, complex closed DCs are green, complex open DCs are yellow and simple DCs are white.

Figure 1.2. Examples for DCs of several types

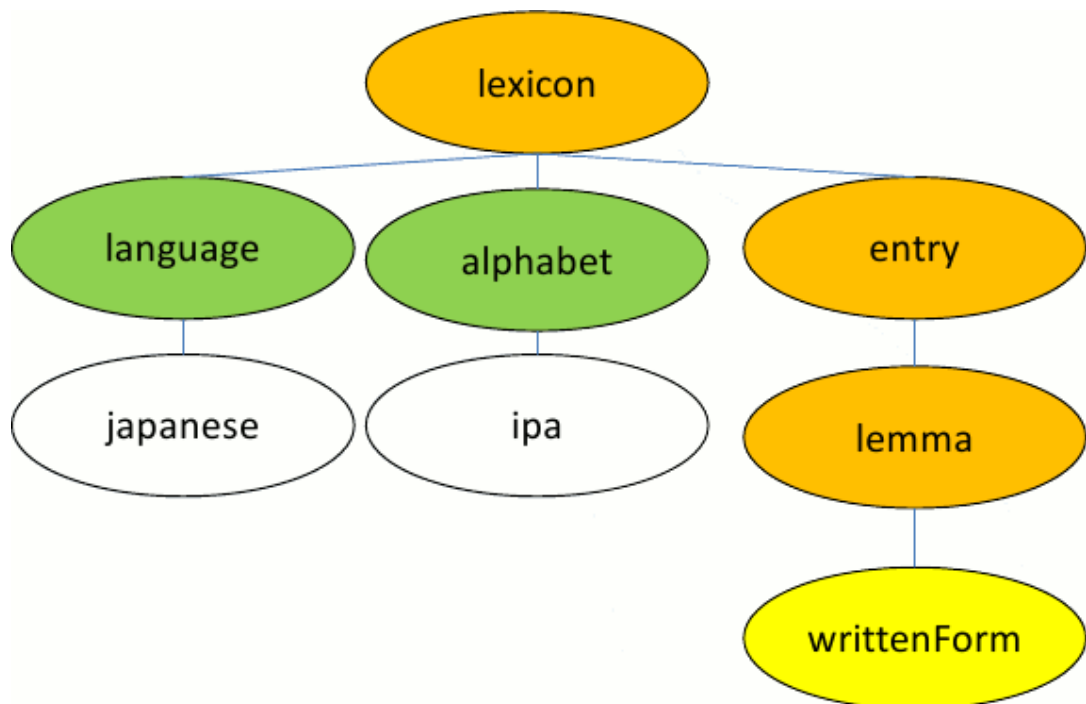


Figure by Menzo Windhouwer

There is one restriction regarding recursion as it is not explicitly stored in ISOcat. Therefore, complex DCs only take simple DCs or basic datatypes (e.g. string) as values (value-domain relations up to depth one) and relations between container categories are not explicitly stored in ISOcat.

2.3. Specifying a data category: administrative, descriptive and linguistic part

In the administrative part a DC is identified. It gets an *identifier* and a *justification*. It is also assigned one of the types mentioned above.

The *identifier* is a name in camel case, i.e. all of its words are written in one joint character sequence without spaces, starting each new word with a capital letter as in `camelCase`). It has to start with an alphabetical character (`firstPerson` rather than `1stPerson`), be meaningful (no abbreviation) and written in English. So, for example, in describing the DC for grammatical gender one could choose the identifier `grammaticalGender`. Nevertheless, the identifier is not unique as a DC can be registered under the same “name” in more than one thematic domain.

The unique identification for the DC is done by automatically assigning it a *persistent identifier* (PID). This unique reference is guaranteed to be resolvable for coming decades. Usually the PIDs of data categories in ISOcat look like the following URI, with a varying number at the end for different categories:

Example 1.3. ISOcat PID

```
http://www.isocat.org/datcat/DC-1297
```

The *justification* explains why this DC is needed or where it is used. Its origin, e.g. the name of the tag set or the piece of literature where this DC was described, can also be given.

In the descriptive part, the DC can be documented in various working languages, it is embedded into one or more *profiles* and it is assigned a *data element name*.

The definitions in various languages should be more or less translations of each other, they should be understandable, and they should not rely on other specific terminological items unless those are defined elsewhere by another DC. In the latter case, one has to make sure that those related definitions cover exactly the term that should be used in the new definition and that the new definition references them explicitly. At least an English language section has to be present. Each language section also includes the correct full name(s) related to the DC in the respective language. On top of that, each DC should be defined to be as much reusable as possible, while being still correct for the purpose of the author.

For example, in a particular tagset a personal pronoun might be “a pronoun referring to persons”, but in general it often also refers to other entities (The cat has five kittens. *She* ... The table was very expensive but I like *it* very much.). Therefore, a more general definition would help making the DC more easily reusable. This effect is increased even more, when keeping the definition as neutral as possible, i.e., without reference to a specific language or project. Definitions like “In English a personal pronoun ...” or “In STTS a personal pronoun ...” restrict the usability of the DC. Information about the origin of a DC can be stated in the administrative part. Nevertheless, the topmost constraint for the definition of the DC is to be valid for the purpose of the author and then for as many other users as possible.

The DC is embedded into one or more profiles. Profiles are managed by so-called *thematic domain groups* (short: TDG), i.e., formally established groups of users who assume a certain responsibility for respective domains such as *Metadata*, *Morphosyntax*, *Terminology*, etc. TDGs

can be proposed via the subcommittees of ISO/TC 37 (the technical committee on *Terminology and other language and content resources*).

The *data element name* is the place to include abbreviations or tags used for this DC. These names do not have to be in English or even any other language, they are language independent. Returning to the example of *grammaticalGender*: *GEN* or *gramGender* could be data element names taken from an application-specific tagset, or a domain-specific XML schema.

In the linguistic part, complex DCs are assigned their conceptual domain(s).

In case of a complex open DC the base data type is specified and in case of a complex constrained DC the base data type is selected and the constraints are expressed in a constraint language, e.g. as an XML Schema regular expression or in Object Constraint Language. For complex closed DCs the simple DCs that are possible values are selected. To give an example for the conceptual domain of a complex closed DC, we go back to the DC *grammaticalGender*. Here another important feature becomes evident: multiple conceptual domains can be assigned, one for each profile and one for each object language respectively. For example with respect to the profile for morphosyntax its conceptual domain can include a range of DCs, e.g. *feminine*, *masculine*, *neuter* and *commonGender*. The language specific French conceptual domain includes *feminine* and *masculine*.

2.4. Ways how to use the ISOcat data category registry

The data category registry is useful for making (parts of) the semantics of resources explicit and so gives insight where the same semantics are shared in different resources.

Therefore ISOcat can (and should) be used in different situations and for the following purposes

- to look up a data category, e.g. in a situation where you need to make the meaning of a descriptor in your (meta)data or annotation explicit;
- to refer to an existing DC from a descriptor in your data
- to create a new data category, e.g. in a situation where you want to make the meaning of a descriptor in your data explicit but no corresponding data category does exist;
- to negotiate with someone else who has already provided a data category, e.g. if you plan to refer to a data category and you think the meaning of it should be changed (made more general, more specific etc.) or the definition should be improved.

Looking up a DC:

The web interface provides a search engine where existing DCs can be searched for by name, profile and description. Screenshots presenting how to search and inspect DCs in the ISOcat web interface can be found at the ISOcat manual [<http://www.isocat.org/files/manual.html>].

Usually when searching for a specific concept many DCs appear, so the profile classification and the description sections of the resulting DCs usually provide a first insight which category to choose. Sometimes many similar DCs may appear as well as DCs containing vague or even ill-formed definitions. Therefore markings in different shape and colour are visibly assigned to the DCs of the search result and state the correctness of the

specification from a technical point of view. As only DCs with a (green) check mark are in principle qualified for standardization, those are the relevant candidates to be referred to.

As standardization of DCs by the ISO is a complex and therefore slow procedure one often has to refer to non-standardized or not-yet-standardized DCs. As those may still be changing at any time, one has to regularly check the referenced DCs for consistence with the original purpose and eventually look for new ones. A CLARIN approach could also be to select data categories, which can already be seen as de-facto-standardized DCs relevant for CLARIN purposes, into a separate workspace in ISOcat.

Referencing an existing DC:

DCs relevant for a resource can be referenced by their PID and can be selected into a *data category selection*. References to the DCs can be embedded in (the scheme of) the resource. Collecting the references in a schema written in a specific schema language (Relax NG, DTD, OWL, EBNF, XSD ...) is preferred, as putting it in the resource itself mostly means to store the references redundantly as a single DC from an annotation (e.g. *noun*) usually occurs many times in the resource. Example 1.4, “Part of a CMDI XSD specification” is a fragment of an XSD specification of a CMDI meta data profile where an element named `Url` is related to a respective DC in ISOcat:

Example 1.4. Part of a CMDI XSD specification

```
<xs:element
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dcr="http://www.isocat.org/ns/dcr"
  name="Url"
  dcr:datcat="http://www.isocat.org/datcat/DC-2546"
  minOccurs="0"
  maxOccurs="unbounded" />
```

The XML format `<tiger2/>` [<http://korpling.german.hu-berlin.de/tiger2/homepage/index.html>] that constitutes an XML serialization of [ISO 24615:2010] *Language resource management -- Syntactic annotation framework* (SynAF) allows for references of annotation *feature* and *value* elements to data categories of a DCR in the annotation declaration of its corpus header. The following example is cited from [Romary et al. 2011] and links the concepts of the *part-of-speech* feature and one of its possible values (*personal pronoun*) to respective DCs in ISOcat:

Example 1.5. Referencing ISOcat DCs in `<tiger2/>`

```
<head xmlns:dcr="http://www.isocat.org/ns/dcr">
  <annotations>
    <feature
      xml:id="f2"
      name="pos"
      domain="t"
      dcrReference="http://www.isocat.org/datcat/DC-1345">
      <value
        xml:id="f2_1"
        name="PP"
        dcr:datcat="http://www.isocat.org/datcat/DC-1463"/>
      </feature>
    </annotations>
  </head>
```


Creating and registering a DC:

If a DC which is needed is still missing in ISOcat, a new one should be created according to the above specification description. In the manual section of the ISOcat webpage [<http://www.isocat.org/files/manual.html>] there is a screencast which also gives an introduction to how to add a new data category.

Nevertheless, the specification scheme for a DC is complex, so the author has to edit it with care, provide meaningful definitions and examples and not confuse the different name categories (identifier, data element name, name sections for specific languages) or specification sections (e.g. language section in descriptive part vs. linguistic section(s) defining language-specific values for complex DCs).

Negotiating changes to a DC:

In case you find a data category which fits your descriptive needs nearly but not exactly, you might consider, instead of creating a new DC from scratch, contacting the creator (or “owner”) of this already existing DCs by the contact information in her or his user profile in order to adapt the meaning and thereby the coverage of this DC. In other words, re-use of a DC with slight modifications is preferable to inventing a new one. You can also share your own DCs with other ISOcat users and make it possible that these DCs might become recommended for CLARIN purposes.

2.5. RELcat, a relation registry

While referring to sets of DCs from different resources helps establishing relations between concepts from the outside, relations between complex or container DCs in ISOcat are not stored explicitly. Therefore the upcoming ISOcat spin-off RELcat is added as a relation registry providing different relation types for the ISOcat DCs.

RELcat is a first prototype of a relation registry, see [Windhouwer 2012], where different relations between ISOcat data categories, and also between data categories from different data category registries can be stored.

Figure 1.3, “Examples for relations in RELcat” shows relations for the metadata categories from the example in Section 1, “Data Categories and Data Category Registries”. Making use of RELcat, both ISOcat data categories *languageID* and *languageName* could be related to the Dublin Core [<http://dublincore.org/>] metadata element *language*.

Figure 1.3. Examples for relations in RELcat

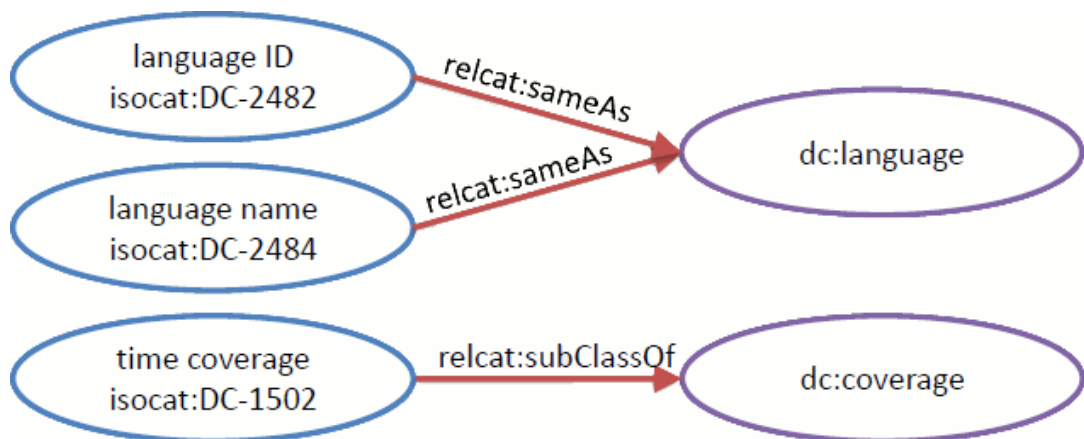


Figure by Menzo Windhouwer from the LREC 2012 tutorial *ISOcat in daily life*

Figure 1.3, “Examples for relations in RELcat” also exemplifies the relation type `rel:subClassOf` of the lowermost relation in the figure, which is utilized to indicate the relation between the two non-equivalent data categories.

A core taxonomy of relationship types is provided for the relations in RELcat, see [Windhouwer 2012], nevertheless other existing vocabularies can be supported by adding their relation types to their proper place in the taxonomy, thereby allowing for the inclusion of existing linguistic knowledge bases. Moreover, generic queries can automatically take multiple relation types into account.

This is also an important feature for the data categories themselves, as data categories which are denoted as equivalent or almost equivalent by the relation type can be automatically included into a query without the user having to know the names or number of the data categories in the set of equivalent data categories.

As RELcat aims at being similarly flexible as ISOcat, the relations stored in RELcat can reflect different views in parallel, e.g. of single users as well as of specific communities, see [Windhouwer 2012].

Chapter 2. Metadata

Peter Wittenburg, Dieter van Uytvanck, MPI for Psycholinguistics
Nijmegen

The research domain is characterized by an enormous increase of the amount of data and the complexity it covers, i.e. the type of implicit and explicit relations included, the heterogeneity of formats and semantic domains, etc. This is also true in the domain of linguistics where it is not the sheer volume only that is creating new challenges for management and access, but it is the extremely growing number of files researchers are creating and using. A field researcher documenting an endangered language for example easily has about 10.000 files on his notebook, which need to be managed. Of course these files cover in particular raw data (AV recordings, texts, etc.), but also many types of annotations, lexica, sketch grammars, notes about various aspects of the language and the field trip, etc. On top of this there are several versions of each work, often several presentation forms (for photos for example JPEG and PNG, versions), extractions of fragments into new files and many other related forms.

It is a common experience that it is almost impossible to manage such a heap of data without having a proper organization and naming scheme. Directory systems were used for many years, but it turns out that these are not appropriate anymore, since they are not meant for sharing and aggregation, do not include the many relations, do not express contextual knowledge, do not support searches, etc.

Throughout this user guide we follow the definition of a *digital object* (DO) as introduced by [Kahn/Wilensky 2006] when referring to different types of stored data as an abstract notion of a digital work that is instantiated in some representational form and is associated with a persistent identifier and a metadata description. We cannot claim thus that DOs are necessarily files, since they could also be constructs in databases for example.

An abstract definition of metadata says “MD is data about data”. In this document we use the term *metadata* as a keyword type of description of data objects. This concept of metadata is not at all new, it was introduced as cards when big libraries were being built. These cards typically combined creation with location information.

It is widely agreed – also across disciplines – that associating metadata with every DO is the only alternative to be able to support management, sharing and access of data in the Internet domain. Currently we see that this wide agreement is turned into strong requirements from funding agencies: projects that include the creation of data need to come up with a *data management plan* where it is described how the data created will be described, preserved and curated. Metadata in this restricted sense can be defined as “structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage and information resources” [NISO:2004].

1. Managing and Accessing Data

Metadata serves a number of important functions such as

- management of large data sets,
- associating access permissions with such data sets,
- discovery of digital objects and data sets,
- information about how to access data objects and sets,

- assisting in re-using data objects by covering context and provenance information, and
- finding appropriate tools for a given data object

The discussion about metadata initiated by librarians mainly focused on the discovery function. However, modern e-Research will require considering the other functions as equally important. New functions such as profile matching will certainly be required.

Due to the range of different functions some experts speak about different types of metadata descriptions such as structural MD, administrative MD, guide MD, preservation MD, technical MD, process MD, descriptive MD, etc. These categories are not standardized, very much dependent of the community using them and are subject to changes. In this document we will not use these terms since we did not find them helpful.

We then need to address the question how we can describe the characteristics of data so that the above-mentioned functions can be realized. The typical way is to define a number of meaningful keywords that can describe the properties of a digital object, its context and provenance. A few examples are typical keywords such as:

- *creator*: the name of the person(s) who created the object
- *country*: the country where an object was created
- *content_language*: the language an object is in
- *date*: the data when an object was created, modified, released, etc. – consider how this example demonstrates that a keyword *date* is obviously not sufficient to allow correct interpretations.
- *actor*: the person(s) actively involved in the data object
- *genre*: the genre the object can be categorized in
- *source*: an indicator of the lifecycle steps that led to this object – this is for example very important when processing video streams. One needs to know which kinds of codecs have been used, what kind of transformations have been applied etc. to do correct interpretations.

With each of these keywords some form of vocabulary or constraint can be associated. With respect to the category “country” for example one may want to associate the official list of nations as accepted by the UN. Such a list of possible values is called a *controlled vocabulary*. With “date” one may want to associate a certain form to be entered such as the US way of writing dates. Such syntactical limitations on the values are called *constraints*. However, for many fields such as “genre” there are no widely agreed vocabularies, i.e. one can only indicate a few typical options, but the list of values basically needs to be open.

2. Objects, Collections, Granularity

Digital Objects as introduced by [Kahn/Wilensky 2006] have a number of internal and external properties. These properties are stored separate from the object and describe the DO as a whole, which can excellently be done by metadata keywords. External properties can be described with typical keywords (categories) as mentioned above. The internal properties typically indicate the technical encoding scheme used, the structure of the object, the semantic used resp. covered, etc. Keyword type of metadata is not meant to contain this information itself, but it should point to an object that contains this information. It should be mentioned here that metadata descriptions are restricted DOs itself, i.e. they need to have an identity that can be used to refer

to them, but they are not described by metadata. Otherwise this would result in an unlimited recursive system.

Some format suggestions for files such as CHAT [<http://chilides.psy.cmu.edu/>] [MacWhinney 2000] and TEI [<http://www.tei-c.org/>] [TEI P5] suggest including so-called *header information* in the file. The widely agreed convention is that metadata needs to be separate, since then it can be used free of licenses, free of the large amounts of bytes the object itself may cover, to be combined to form all sorts of virtual collections, to merge metadata from various sources, or to update the metadata without changing the object. This is very important since updating the header information in a file means creating a new object requiring an own identity and thus version. Converters will allow extracting the header information from such files to generate the metadata.

DOs are related in many different ways as a whole or amongst its fragments. Here we only discuss examples where DOs are related as a whole. Typical examples are: the DO is a new version of an older one, the DO is a different presentation version, the DO is part of a series of DOs that was created at the same time and location, the DOs are about the same content_language, the DOs include the same actors, and there are many more possibilities of relations users want to express. Metadata systems should allow the creator, manager and end-user to form so-called virtual collections, i.e. they should support the user in aggregating the metadata descriptions of DOs even from various repositories to build collections fit for any kind of purpose such as writing a thesis on a collection of objects. Figure 2.1, “Building virtual collections by aggregating metadata descriptions” indicates this process: a user can aggregate metadata descriptions from one or more existing collections into a “basket” to create a collection. To build virtual collections the actual DOs are not moved, only the metadata descriptions that have pointers to the objects are being collected. Such a virtual collection can also be described by a metadata description, which will contain next to the typical metadata keywords describing its properties a long list of references pointing to the metadata descriptions of the objects included.

Figure 2.1. Building virtual collections by aggregating metadata descriptions



Conversely, there is a large debate what kind of granularity should be chosen to assign metadata that can be used in the above-mentioned ways. Many repositories still offer only metadata descriptions for whole collections without pointing to the descriptions of individual objects. Imagine a corpus of lower-saxonian German including variants spoken at the coastal areas from the North Sea and Baltic Sea and recorded over some time. We could give the whole corpus one metadata description to publish and register it. This would allow users to find this corpus and to work with it as a whole. But let's assume that an analysis work is directed to the question whether there are differences between male and female speakers in losing their capabilities of speaking the variants. There would be no chance based on metadata descriptions to make a simple query and group the whole collection into two or more sub-collections. Having a high granularity of metadata descriptions simplifies re-using a collection in particular in ways as they were not foreseen by their creators and thus supporting new research questions.

**Tip**

Thus we can conclude that it makes sense to associate each meaningful digital object with a metadata description to support identifying and re-combining them to address new research questions.

3. Types of Resources and Metadata Components

Metadata frameworks need to allow researchers to describe different resource types that occur in the area of linguistics ranging from raw material in form of texts, audio and video recordings, brain imaging, eye tracking etc. to derived data which can include completely different types such as lexica, results of statistical analysis in table form, etc. All these resource types can be created by researchers from different sub-disciplines in linguistics. Thus the heterogeneity of the resource types and the intentions of the researchers needs to be covered by a metadata framework that allows to use metadata for research questions and not just to discover useful resources by approximate semantics.

In 2000 two groups discovered that the suggestions coming from the library world were not sufficient to meet the researchers' needs:

1. In early 2000 the IMDI group [<http://www.mpi.nl/imdi/>] (widely European experts) decided to develop the IMDI metadata set that is structured, extendible and includes domain semantics to express the linguistic wishes.
2. In late 2000 the OLAC group, with its origins mainly in the US, decided to extend the Dublin Core [<http://dublincore.org/>] set by a few linguistically relevant categories to meet the most urgent needs, but also to remain simple.

Other suggestions were made in the linguistic domain such as ENABLER [<http://www.enabler-network.org/>], but since no tools supported them these suggestions were not used. Both IMDI and OLAC were used by various linguistic resource centers with different purposes in mind. However, both approaches suffered from some major deficits:

1. Despite possibilities to add extensions they both offered a limited set of categories – OLAC severely more restricted than IMDI.
2. Despite its greater expressiveness due to structure options IMDI as well as OLAC had a fixed schema, i.e. even if a creator only knew about four values for example he had to cope

with all requested input fields or when a new sub-discipline (e.g. sign language experts) wanted to use IMDI a new special profile had to be created and integrated.

It was obvious that only a flexible component model could overcome the limitations and give all researchers from the various sub-disciplines the possibility to create the profiles they would like to use and that are tailored for their intentions. It is obvious that syntax does not hamper interpretation if the meaning of the categories being used is widely independent of their structural embedding, i.e. these definitions have to be semantically narrow. We need to define categories such as *date of birth*, *date of creation*, *date of annotation*, etc. instead of semantically broad categories such as *date* where the interpretation is defined by its structural embedding. These considerations were the motivation to build CMDI:

- it should allow users to define their own components resulting in tailored profiles,
- the components need to make use of categories the definitions of which are registered in ISOcat (see Section 2, “ISOcat, a Data Category Registry”), and
- semantic interoperability and interpretability is guaranteed by fine-grained semantics.



Tip

Only flexible component models have the expressive power to cover the heterogeneity of a broad field in terms of resource types, variety of sub-disciplines and research intentions.

4. Lifecycle Management

Lifecycle management of data is becoming a very important issue, since proper lifecycle management will influence the accessibility and re-usability of data over years. Lifecycle management includes a number of aspects such as taking care of preserving identity, integrity and authenticity of data, of data curation, of creating metadata to capture contextual and provenance information, etc. Increasingly often data objects are not created by manual operations anymore, but by algorithms that automatically operate on existing data objects. For all creation acts – be it manual or automatic – it is of crucial importance to consider lifecycle management aspects from the beginning, i.e. in particular to create metadata.

In both cases it means extra work for the software developers or data creators, resulting in large delays of the creation of metadata descriptions. But [Beagrie 2001] showed convincingly that shifting the creation of metadata to a later point in time would be much more costly compared to doing it immediately.



Tip

We can only recommend starting with metadata considerations and with the creation of metadata descriptions as early as possible.

5. Existing MD sets

Here we want to briefly introduce a few relevant metadata standards and best practices respectively that are relevant for the area of linguistic resources and tools.

5.1. Dublin Core Metadata Initiative (DCMI)

The Dublin Core Metadata Initiative [<http://dublincore.org/>] was started in the library world to come up with a descriptor set that can be used to describe all kinds of web-resources. In total 15 categories have been defined by a worldwide harmonization process. Due to the need to describe the most different types of objects these 15 categories are semantically broadly defined and a generic terminology is being used. DCMI does not make statements about a specific schema to be used. Since for many applications these 15 categories were too broadly defined DCMI defined later the “qualified DC” categories, which have a narrower semantic scope.

DCMI has still much relevance in the world of digital libraries and is often used in areas where global searches are seen as being sufficient. The DC set is widely accepted for cross-disciplinary metadata aggregation although much information in general is being lost.

5.2. ISLE Metadata Initiative

Early in 2000 the ISLE (International Standards for Language Engineering) project [http://www.ilc.cnr.it/EAGLES/isle/ISLE_Home_Page.htm] was started to come up amongst others with a metadata standard more focusing on the world of multimedia and multimodal resources. After several discussions with the DCMI user group it was finally decided by the IMDI group to use a structured metadata set to have greater expressive power and to better model the descriptions of such complex resources or resource bundles as they occur in linguistics, to make use of linguistic terminology to support research questions, to leave space for extensions by user-defined key-value pairs and to allow data managers to use IMDI to organize and manage large data collections. A schema and tools were developed allowing interested people to use IMDI for real work. IMDI has been used by a number of projects and data centers worldwide; however, its outreach was limited.

5.3. Open Language Archive Community (OLAC)

In the autumn of 2000 the OLAC initiative [<http://www.language-archives.org/>] was made public with one of its main goals to extend the DCMI category set by four additional and linguistically meaningful categories (*linguistic field*, *linguistic type*, *discourse type*, *language code*). OLAC presented itself as a metadata service provider, i.e. an organization that will harvest metadata from linguistic data centers and support retrieval services. Thus the OLAC set is not meant as a way to organize and manage large collections and to support specific research questions.

It is widely agreed by linguistic data centers to produce OLAC compliant mappings and to allow OLAC to harvest all metadata records. Large variations in the granularity of the offered descriptions present a challenge for search engines as well as for users.

5.4. CHAT

The CHAT format was invented to support the CHILDES (Child Language Data Exchange System) program [<http://chilides.psy.cmu.edu/>] in creating and collecting much data in particular about how children are talking and interacting. The CHAT format is a pure ASCII-based flat text format that has defined so-called header categories that can be compared with typical metadata keywords, since it allows specifying the speakers, the language and other contextual aspects. CHAT allows describing also sub-sections, i.e. header type of information can occur everywhere in the annotation transcript. CHAT and with it the header categories is a

widely used format in developmental linguistics and beyond. Metadata and annotation data are thus merged in one file, so that metadata extraction is required to support the relevant functions.

5.5. Text Encoding Initiative (TEI)

The Text Encoding Initiative [<http://www.tei-c.org/>] recently presented their P5 version [<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/HD.html>] allowing users to describe a wide variety of resource types. Its expressional power is extensive since its structural specifications allow users to combine categories in almost unlimited ways. Also TEI has so-called header categories that are meant to describe the whole resource and these header descriptions are part of the total document structure, i.e. typical metadata categories and annotation categories can appear intertwined. Due its almost unlimited expressive power almost all applications of TEI are based on specific sub-schemas. Thus TEI resources appear in a large variety of flavors the interpretation of which requires knowledge about the specific schemas.

TEI is widely being used in some humanities disciplines. However the interpretation of the TEI files widely depends on the availability of specific schemas. For harvesting metadata from TEI files the specific sub-schema must be known and an extraction has to be done.

5.6. Component Metadata Initiative (CMDI)

The Component Metadata Initiative [<http://www.clarin.eu/cmdi>] brought together a large group of leading linguists from different sub-disciplines to define a metadata framework that is flexible enough to cover the different wishes from the various sub-disciplines and projects, but nevertheless has the expressive power to serve for the various functions mentioned above including those that are emerging in the e-Research scenario. As already indicated the core of CMDI was the definition of a set of categories the semantics of which are sufficiently specific to guarantee interoperability and interpretability. Substantial work was carried out by the group of linguists to define a robust set of categories which have been registered in ISOcat and which can be extended and altered if necessary. A flexible syntactical framework allows users to combine categories to components and profiles.

The CMDI framework will be explained in more detail below. Here we would like to summarize that CMDI can be seen as a flexible syntactical umbrella to include the metadata categories defined so far, to cover the needs of a wide variety of disciplines, the requirements posed by different usages and linguistic data types. In so far it is a big step ahead with respect to expressive power and coverage. However, communities of usage (such as OLAC etc) are not requested to change their practices as long as no additional functionalities are required.

6. The Component Metadata Initiative (CMDI)

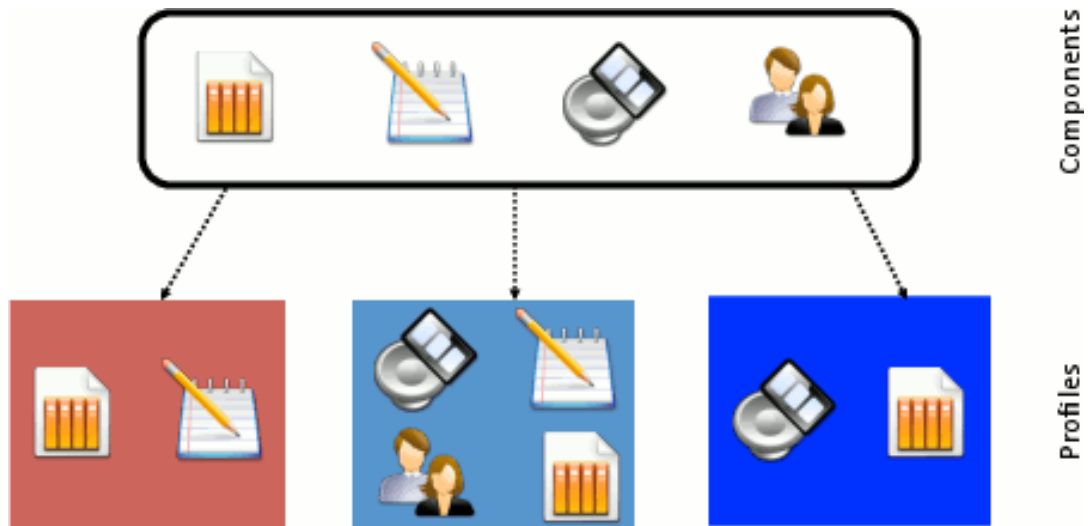
6.1. What is it?

Rather than a single metadata format, the Component Metadata Infrastructure [<http://www.clarin.eu/cmdi>] provides a framework to create and use self-defined metadata formats. It relies on a modular model of so-called *metadata components*, which can be assembled together, to improve reuse, interoperability and cooperation among metadata modelers. With a culinary analogy one could call it metadata “à la carte”: instead of choosing one completely predefined

schema you can select your preferred plates (*components*) and group them until you have a whole (*profile*) that suits your needs.

There is however an important difference with a restaurant. In case none of the proposed components fulfill the requirements, the user can always create a new one. Figure 2.2, “Profiles are made up of components” displays how a relatively small set of components (general metadata, metadata for textual resource, metadata for multimedia and metadata on persons) can be combined in tailored profiles.

Figure 2.2. Profiles are made up of components



6.2. Why yet another metadata format?

In the previous section a whole list of existing metadata formats was listed. One could ask why it is necessary to come up with a new metadata formalism, and why CLARIN thinks this will make a difference compared to choosing one of the existing formats. The answer is manifold:

- First of all CMDI is not just another format. It is much more: as a meta-model it provides a well-defined framework to define and use your own format. It also allows the user to integrate existing schemas (IMDI, OLAC) as components and thus offers interoperability to the existing base.
- No single metadata scheme could ever address all the needs of the heterogeneous community of humanities and social sciences researchers: they range from describing Greek texts on vases over gesture analysis in YouTube videos to noting down phonetic features of telephone recordings. Hence the need for a flexible solution.
- There is a clear need for semantically explicit metadata descriptions. Ambiguity could otherwise threaten the usefulness of metadata when many metadata descriptions, coming from a multitude of sources, are made searchable.

In the coming section we will go into more technical details of the CMDI model.

6.3. The CMDI model

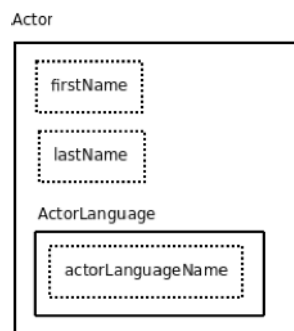
Work on CMDI started in 2008 in the context of the European CLARIN research infrastructure [<http://www.clarin.eu>]. Most existing metadata schemas for language resources seemed to be

too superficial (e.g. OLAC) or too much tailored towards specific research communities or use cases (e.g. IMDI).

CMDI addresses this by leaving it to the metadata modeler how a schema should look like. It is based on the use of metadata components making use of agreed and registered categories. These elementary building blocks contain one or more elements (also known as *fields*) describing a resource. For instance, an *actor* component can group elements like *first name*, *last name* and *sex*. A component can also contain one or more other components, allowing a lego brick approach, where many small components together form a larger unit. To continue with the actor example, such a component could include a sub-component *actor language*, containing a set of fields describing the *language(s)* a person can speak.

Figure 2.3, “A component describing an actor” shows a typical component describing an Actor [http://catalog.clarin.eu/ds/ComponentRegistry?item=clarin.eu:cr1:c_1274880881804], consisting of two elements (*firstName* and *lastName*) and an embedded ActorLanguageName component. You can explore the component in the CMDI component registry [http://catalog.clarin.eu/ds/ComponentRegistry?item=clarin.eu:cr1:c_1274880881804].

Figure 2.3. A component describing an actor



Ultimately a set of components will be grouped into a profile – this “master component” thus contains all fields in a structured way that can be used to describe a language resource.



The CMDI component registry

In order to promote the re-use and sharing of components and profiles, the CMDI component registry [<http://catalog.clarin.eu/ds/ComponentRegistry/#>] was created. A web application (see Figure 2.4, “The CMDI component registry”) allows metadata modelers to browse through all existing components and profiles and to create new ones, with the possibility to include existing components. The component registry is open to anyone to read components. Submitting new components can only be done by accredited experts to guarantee that only correct and proven components are ready for reuse by others.

Figure 2.4. The CMDI component registry

The screenshot shows the 'Clarin Component Browser' interface. At the top, there are buttons for 'Browse...', 'Edit...', and 'Import...'. The user is logged in as 'anonymous'. Below the buttons, there are tabs for 'Profiles' and 'Components'. The 'Components' tab is active, showing a table of components. The table has columns: Name, Group Name, Domain Name, Creator Name, Description, and Registration Date. The 'cmdi-language' component is highlighted. Below the table, there is a detailed view of the 'cmdi-language' component, showing its name, group name, description, element type, concept link, number of occurrences, and multilingual status.

Name	Group Name	Domain Name	Creator Name	Description	Registration Date
cmdi-annotat...	clarin		Patrick Duin	List of frequently used annotation formats	21 April 2010 16:...
cmdi-language	clarin		Patrick Duin	Component for describing a certain language (free name, ISO-639-3 code)	21 April 2010 16:...
cmdi-location	clarin		Patrick Duin	Component for describing a certain location (address, region, country, continent)	21 April 2010 16:...
cmdi-mimetype	clarin		Patrick Duin	List of frequently used mime types	21 April 2010 16:...
iso-3166-1-bl...	clarin		Patrick Duin	The list of ISO3166-1 country names, using 2 alphabetical characters. Based o...	21 April 2010 16:...
iso-639-1	clarin		Patrick Duin	The list of ISO-639-1 language codes	21 April 2010 16:...
iso-639-3	clarin		Patrick Duin	The list of ISO-639-3 language families. Based on:	21 April 2010 16:...
iso-639-5	clarin		Patrick Duin	The list of ISO-639-5 language families. Based on: http://en.wikipedia.org/wiki...	21 April 2010 16:...
iso-continent	clarin		Patrick Duin	Component listing all continents using their ISO code	21 April 2010 16:...
Operation	CLARIN core ...	Unknown	mwindhouwer	A Web Service operation	08 August 2011 1...

cmdi-language	
Name:	Language
Group Name:	clarin
Description:	Component for describing a certain language (free name, ISO-639-3 code)
Element:	LanguageName string
ConceptLink:	http://www.isocat.org/datcat/DC-2484
Number of occurrences:	1 - 1
Multilingual:	true
Component:	iso-639-3
Number of occurrences:	1 - 1

After creating or choosing a profile, the user can generate an XML W3C schema (also known as an XSD file) that contains a formal specification of the structure of the metadata descriptions that are to be created. This schema too can be accessed from the component registry, with a right click on the profile, choosing the “Download as XSD” option. From then on the schema can be used to check the formal correctness of the CMDI metadata descriptions. See Section 1.1, “Well-formedness and schema compliance” for more information on formal correctness checking.



CMDI data format

A CMDI metadata description (stored as XML file with the extension `.cmdi`) consists of three main parts:

- A fixed `Header`, containing information about the author of the file, the creation date, a reference to the unique profile code and a link to the metadata file itself.
- A fixed `Resources` section, containing links to the described resources or other CMDI metadata descriptions.
- A flexible `Components` section, containing all of the components that belong to the specific profile that was chosen as a basis. In our earlier example there would be one `Actor` component immediately under the `Components` tag.

A set of CMDI example files can be found on the CLARIN EU website [<http://www.clarin.eu/page/3312>].

6.4. Explicit semantics

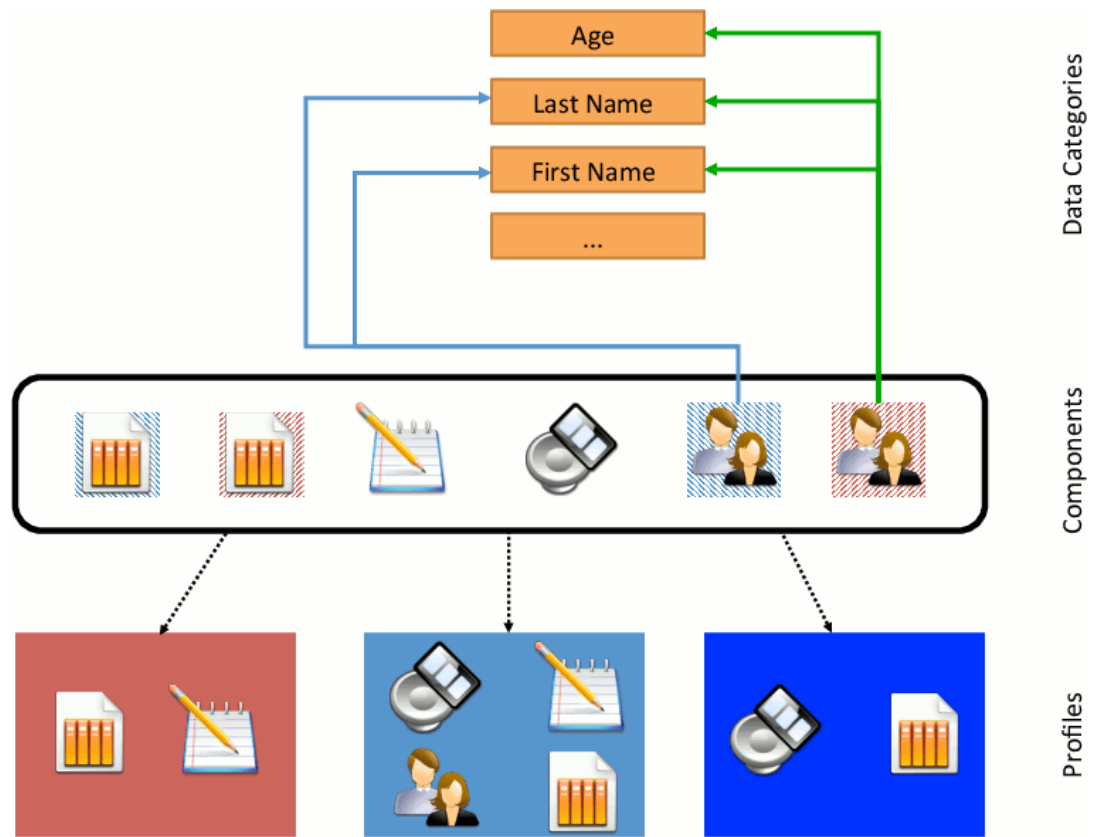
To avoid ambiguity and to achieve clear semantics when using metadata, the CMDI model has close ties to the ISOcat data category registry (see Section 2, “ISOcat, a Data Category Registry”). The model can also be easily extended to other widely agreed registries of data categories. Because the system relies on a (potentially large) set of components created by many different users, the risk is there that searching in the metadata descriptions becomes unfeasible as:

- an element might have different names within multiple components (e.g. *name* or *last name*),
- a user might not be aware that there are multiple components that contain a specific element, even when they are called the same (e.g. *name*), and
- some elements might have the same label (say *name*) but might mean something different (e.g. *project name* versus *person name*).

The solution (or at least a part of it) is in declaring what each element really means. When adding an element to a CMDI component the metadata modeler has to add a link to the ISOcat data category registry, where very detailed definitions are available. This link provides a persistent and unique identification of the intended semantics.

Consider the example mentioned above consisting of two components, both containing a description of an actor, where one refers to a person’s last name with the label “Name” and another one with the label “Last Name”. When both point to the data category `last name` [<http://www.isocat.org/datcat/DC-4195>], both the search machines and human users know what is precisely meant. It is the reference and not the used label that clarifies the intended semantics. This method is illustrated in Figure 2.5, “Declaring explicit semantics in CMDI with links to data categories”.

Figure 2.5. Declaring explicit semantics in CMDI with links to data categories



More information on CMDI and ISOcat can be found on a frequently asked questions page of the CLARIN EU website [<http://www.clarin.eu/faq/281>].

6.5. Procedure

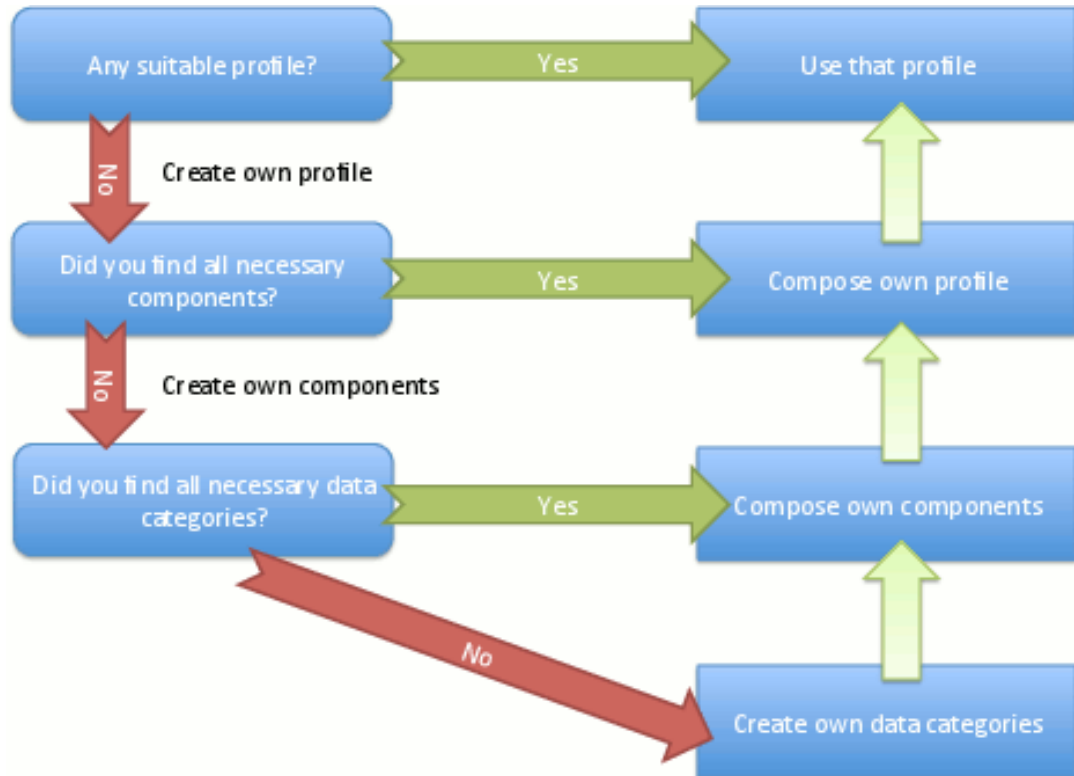
As the CMDI framework aims to enhance the reuse and sharing of metadata components, one of the important aspects in using it is finding out existing material (be it profiles, components or data categories) that could be used. Only if this is not the case one should create these from scratch.

In general, the procedure to identify relevant building blocks looks as follows (the diagram in Figure 2.6, “Procedure for CMDI metadata modeling” summarizes the whole procedure):

1. Search in the component registry for interesting profiles. If one of them matches the requirements, use this one.
2. If a profile matches more or less the requirements, check how to create a derivative that meets your requirements.
3. Otherwise, look in the Component Registry for useful components:
 - a. If you can create a profile out of existing components, do so.
 - b. Otherwise, create additional components:
 - i. Link to existing data categories when they match the semantics of the elements in your component.

- ii. Otherwise, create new data categories yourself and link to these from the new component. Consult Section 2, “ISOcat, a Data Category Registry” for more information on how to do this.

Figure 2.6. Procedure for CMDI metadata modeling

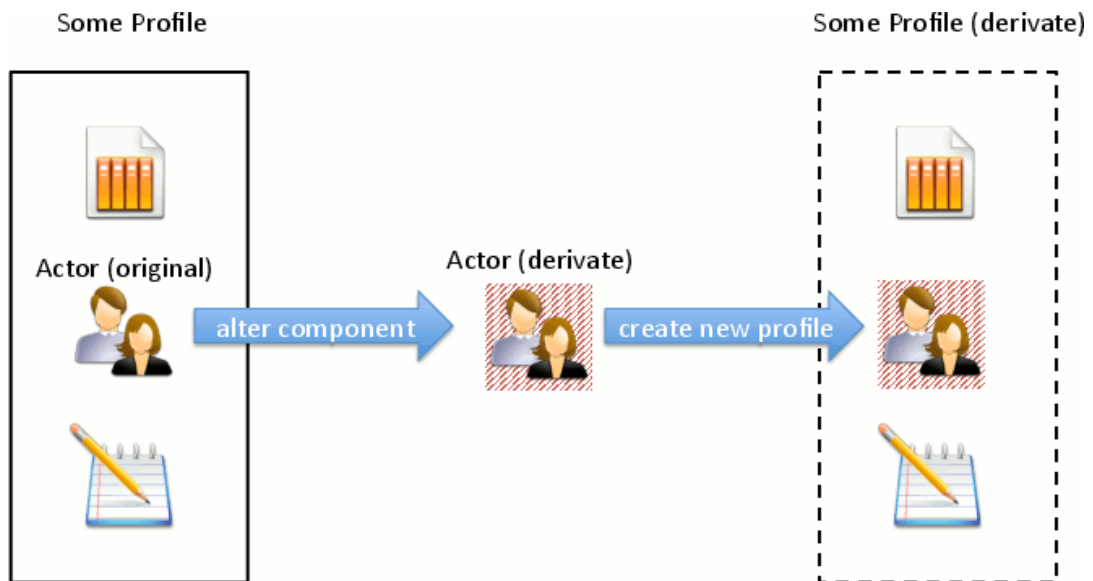


A practical example to CMDI profile creation

This section will contain a concise step-by-step guide starting an MD profile from scratch based on an existing resource.

6.6. Profile and component adaptation

Many users find that there is an existing profile or component (further on: component) that mostly fits their needs, but just needs some minor changes, e.g. adding an extra element or renaming an existing one. In such cases it is possible for expert metadata modelers to use an existing component, make the preferred changes and to store it separately in the component registry. It is important to realize that such derivative components are not any longer connected to the original. In case of components with embedded subcomponents, this means that even a small change in one of the lower-situated components results in the need to create alternate versions of the parent components, as these are built in a bottom-up manner. Thus creating a new component requires some care from the user. This way of working is illustrated in Figure 2.7, “Creating a derivative profile with an altered component”.

Figure 2.7. Creating a derivate profile with an altered component

6.7. Preferred components and profiles

With currently about 300 components and about 70 profiles available, the question arises if we can recommend some components. This is certainly the case, we strongly advise to use those components, which contain standardized vocabularies for language names, country names, continents, etc. This will greatly enhance the interoperability and the metadata quality. Some other components, like `cmdi-description` [http://catalog.clarin.eu/ds/ComponentRegistry?item=clarin.eu:cr1:c_1271859438118] are also general enough to be recommended. It should be noted that the list of recommended components and profiles [<http://www.clarin.eu/faq/3491>] that adhere to high quality standards (e.g. there are ISOcat links for all elements used) is constantly growing and therefore it is recommended to check the most-up-to-date information.

6.8. Converting existing metadata to CMDI

Quite often there is already some metadata available in a non-CMDI format. For certain frequently occurring conversions there are conversion methods available. Such a method includes:

- a corresponding CMDI profile that contains all of the fields of the original metadata format and
- a conversion script or stylesheet for the actual conversion.

For OLAC, DC, IMDI and TEI-headers these conversion methods are described in detail on the CLARIN EU website [<http://www.clarin.eu/faq/282>]. A similar conversion scheme for MetaShare [<http://www.meta-net.eu/meta-share>] metadata is planned but not yet available.

7. Aggregation

7.1. Metadata Harvesting

The wish to have a central catalog covering metadata aggregated from different repositories has led to the emergence of the Open Archives Initiative's protocol for metadata harvesting (OAI-PMH [<http://www.openarchives.org/pmh/>]) as a de-facto standard for gathering metadata into a single catalog – a process that is called metadata harvesting.

In the OAI-PMH model the world is divided in data providers, that offer metadata for harvesting, and service providers that harvest the metadata and offer discovery service to the world, for instance a central metadata catalog. The OAI-PMH protocol is fairly efficient and simple to implement. Each data center that is offering linguistic resources should offer metadata descriptions that can be harvested by service providers – for CLARIN centers it is mandatory to support the OAI-PMH protocol.

OAI-PMH requires that the metadata is in all cases also offered in DC format next to any other format such as for example CMDI. This allows that metadata from different disciplines can be harvested and put into one catalog although presumably much information will be lost by mapping all metadata to the rather simple and restricted DC set. When all harvested data providers have agreed to also provide metadata of another set than DC, it is of course possible to create a more useful catalog.

7.2. Metadata gathering and searching: the Virtual Language Observatory (VLO)

Gathering all CMDI metadata into one large basket does not make sense without an intuitive interface to search and explore the information that it contains. In the case of CLARIN the metadata provided by the centers in the CMDI format is harvested via OAI-PMH and then made accessible via the Virtual Language Observatory [<http://www.clarin.eu/vlo/>] (VLO). More details on the harvesting process employed by the VLO can be found on the CLARIN EU website [<http://www.clarin.eu/faq/279>]. Anyone else can also build a portal based on the same principles. However, it should be noticed that aggregating and mapping metadata from different service providers in general does involve much curation effort.

With the VLO's facet browser it is possible to quickly navigate through this constantly growing inventory of language resource (and tools) metadata. Using a full-text search one can quickly identify electronic and non-electronic sources of information. The results can also be refined step-by-step by specifying a particular language, collection, resource type, subject and so on. While the VLO is not a tool by itself that can directly answer research questions it allows any user with an Internet connection to efficiently search within a metadata catalogue, as to identify language resources and tools that might be helpful for research purposes. As such it can serve as leverage for the reuse of data sets and archived language material in general.

Suppose a multidisciplinary research team of historians, linguists and cultural psychologists is investigating sign language iconicity in the context of the fall of the Berlin wall. Figure 2.8, “Using the VLO to identify relevant language resources” and Figure 2.9, “Accessing the results of a VLO query” illustrate how the VLO can be used to explore some of the resources that might be relevant for such a study.

Figure 2.8. Using the VLO to identify relevant language resources



A full text metadata search on the word “Wende” returns 23 results (out of 179.000)

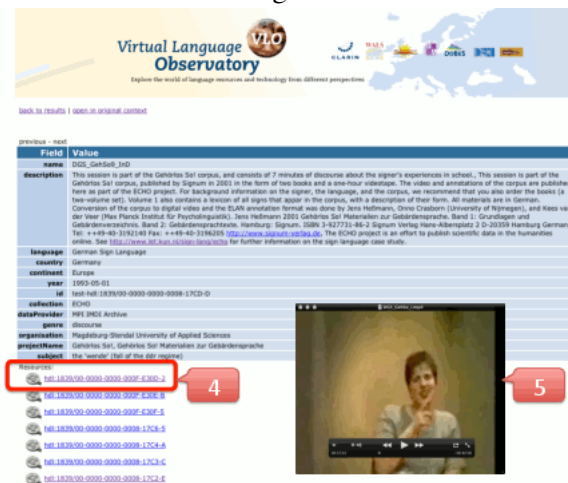


This set is narrowed down to 1 result when selecting only the results that are about German Sign Language

Figure 2.9. Accessing the results of a VLO query



The user clicks on this single result to view the details of the metadata



A user clicks on one of the metadata-described resources and can inspect it

8. Recommendations

As a conclusion to this metadata chapter we would like to give some advice about metadata creation in general:

- Always start as early as possible to collect and create metadata. Otherwise chances are high that information is lost or that fixing the incomplete metadata records afterwards will be very costly.
- Try to achieve a high but reasonable level of granularity.
- When using CMDI (which is required by CLARIN):
 - Try to reuse as much as possible. It should save you work and will enhance the interoperability.
 - Be aware that there are conversion methods in place for the most widely used formats – there is no need to reinvent the wheel.

Chapter 3. Resource annotations

Kerstin Eckart, Universität Stuttgart

To annotate a resource usually means to enhance it with various types of (linguistic) information [McEnery/Wilson 2001, page 32]. This is done by attaching some kind of information to parts of the resource and/or introduce relations between those parts which can again be *annotated* with information. Thereby the annotated information is always an interpretation of the data with respect to a particular understanding.

Annotating a resource can be done manually by one or more annotators, automatically by a tool or semi-automatically e.g. by manually correcting automatic annotations. The annotated information can be represented in terms of atomic or hierarchical tags, complete feature structures or simple feature value pairs. The whole annotation document is stored or visualized in a specific *representation format*.

The rules by which information is attributed to parts of the resource are captured in an *annotation scheme*, specifying e.g. guidelines, which should be used to inform and control the work of human annotators or correctors. [Schiller et al. 1999] for example specifies guidelines for annotating parts-of-speech. They also provide a finite set of category abbreviations, thus composing a *tagset* (the Stuttgart-Tübingen tagset for part-of-speech tagging, STTS). Other tagsets for part-of-speech and syntax annotation are utilized in the Penn Treebank (PTB) project [<http://www.cis.upenn.edu/~treebank/>], see for example [Santorini 1990] and [Bies et al. 1995].

Linguistic annotation schemes reflect linguistic theories or are tailored with respect to the investigation of a specific phenomenon. They are an essential part of the documentation which accompanies an annotated resource.

However, it is important to distinguish between the concepts of *guidelines*, tagsets and representation formats. While tagsets often evolve when elaborating the annotation guidelines and the tags tend to be abbreviations for the decisions made, they could definitely be replaced by other abbreviations, i.e., there could be more than one tagset part of an annotation scheme, e.g. tagsets differing in granularity. Some tagsets can be understood as *taxonomies*. In the abbreviations applied in STTS the letters of the tag denote more general information on the left and more specific information on the right: VVFIN is a main verb (VV) which is finite (FIN), VAFIN is a finite auxiliary verb. This structure can be helpful when queries are conducted via regular expressions.

Some annotations cannot make use of a predefined tagset, such as when annotating synonyms. The representation format again is distinct from the concepts of the guidelines and tagsets. It specifies how the annotation content is represented. One has to be aware that on the one hand annotations with the same content can be represented in very different ways and that on the other hand using the same representation format, i.e., an XML format like TIGER XML [<http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html>] for two annotations does not mean that the annotation content of the two is the same, or even similar.

Lastly, as stated by [McEnery/Wilson 2001, page 33] and [Leech 1993] one has to be aware, that any act of annotating a resource is also an act of interpretation, either of its structure or of its content. Therefore an annotation is never universal consensus. Moreover it has to be taken into account that the process of annotating is also likely to introduce errors, see Chapter 5, *Quality assurance*.

1. Aspects of annotations

In this section we introduce the different ways how to technically attach linguistic annotations to a resource, and which advantages, disadvantages and consequences come along with each of these. We show examples of different annotation styles and annotations from different linguistic layers, which are utilized by existing resources, current research projects and/or within the CLARIN-D infrastructure.

1.1. Inline vs. stand-off annotations

Annotations have mostly been attached to the related parts of the resource as *inline annotations* or *stand-off annotations*.

Inline annotations are directly included into the resource, thereby changing the primary data. Examples are token-based or phrase-based annotations which are directly attached to the related string with separators or structural tags. We use the term *token* to include punctuation and to avoid going into the definition of the notion *word* here, as it may vary in different resources.

Example 3.1, “LOB style inline annotation” shows an inline part-of-speech annotation in the encoding style of the tagged Lancaster-Oslo-Bergen (LOB) corpus [<http://khnt.hit.uib.no/icame/manuals/lobman/index.htm>], Example 3.2, “Penn Treebank style inline annotation” shows an inline annotation of part-of-speech and syntactic phrases using the bracketing style of the Penn Treebank. Both examples represent the sentence *Er fährt ein Auto.* (“He drives a car.”).

Example 3.1. LOB style inline annotation

```
Er_PPER fährt_VVFIN ein_ART Auto_NN ._$.
```

The tags included in this example and in Example 3.2, “Penn Treebank style inline annotation” are NN (common noun), PPER (personal pronoun, excluding reflexive pronouns), VVFIN (finite main verb), ART (determiner), \$. (sentence-end marker), S (sentence), NP (noun phrase). Sentence and tags have not been taken from the mentioned corpora, which contain English text and make use of other tagsets, but have been chosen for exemplification of the annotation styles only.

Example 3.2. Penn Treebank style inline annotation

```
(S
  (NP (PPER Er) )
  (VP (VVFIN fährt)
    (NP (ART ein) (NN Auto))
  )
  ($. .)
)
```

See Example 3.1, “LOB style inline annotation” for a description of the tags used here.

Stand-off annotations are stored separately from the primary data they refer to, see [Thompson/McKelvie 1997], thereby leaving the primary data untouched. References into primary data,

e.g. the original text to be annotated, or into other annotation layers denote the parts to which the annotations belong, see [Zinsmeister et al. 2008]. For these references different mechanisms can be used, often depending on the media type of the primary data.

Example 3.3, “PAULA annotation” shows a stand-off annotation making use of XPointers, see [Grosso et al. 2003], for reference into the primary data. This requires the original resource to be stored in a file which can be referenced by XPointers such as an XML file.

Example 3.3. PAULA annotation

```
<body>Er fährt ein Auto.</body>

<mark id="tok_1"
  xlink:href="#xpointer(string-range(//body,' ',1,2))"/>
<mark id="tok_2"
  xlink:href="#xpointer(string-range(//body,' ',4,5))"/>

<!-- ... -->

<feat xlink:href="#tok_1" value="stts.type_pos.xml#PPER"/>
<feat xlink:href="#tok_2" value="stts.type_pos.xml#VVFIN"/>

<!-- ... -->
```

The example is encoded in the PAULA format, see [Dipper 2005], version 1.1. Tokens are defined with respect to relative character positions. The first token starts in position 1 of the string and includes 2 characters etc., see [Zinsmeister 2010].

Stand-off annotation is also used when annotating other media types than text, e.g. audio files. In the WAVES format, for example, transcriptions and annotations are aligned by timestamps.

The Linguistic annotation framework (LAF, [ISO 24612:2012]) refers to primary data via an arbitrary number of anchors specifying medium-dependent positions, e.g. coordinates, frame indexes or pre- and post-character positions in text, video or other kinds of data. For an example see Section 2, “Exchange and combination of annotations”

There are also hybrid forms of referencing combining stand-off and inline approaches such as the XML-format TIGER XML. In this format the original text is segmented into tokens without references into the original resource. The annotation of the syntactic phrases is represented as a separate layer on top of the part-of-speech annotation of the tokens. In the annotation part for the syntactic phrases, the tokens are referred to by identifiers, e.g. s1_1. Example 3.4, “Tiger XML” shows an excerpt of a part-of-speech and syntax analysis of the example given above, this time represented in TIGER XML.

Example 3.4. Tiger XML

```
<s id="s1" >
  <graph root="s1_500" >
    <terminals>
      <t id="s1_1" word="Er" pos="PPER" />
      <t id="s1_2" word="fährt" pos="VVFIN" />
      <t id="s1_3" word="ein" pos="ART" />
      <t id="s1_4" word="Auto" pos="NN" />
```

```
        <t id="s1_5" word="." pos="\$. " />
    </terminals>
    <nonterminals>
        <nt id="s1_502" cat="NP" >
            <edge idref="s1_1" label="--" />
        </nt>
        <nt id="s1_503" cat="NP" >
            <edge idref="s1_3" label="--" />
            <edge idref="s1_4" label="--" />
        </nt>
        <!-- ... -->
    </nonterminals>
</graph>
</s>
```

For resources other than text, stand-off annotation with references into the original data is the only way to annotate at all, but annotators of textual resources have so far made use of inline annotations in many projects. While processing or querying stand-off annotation includes some sort of link or pointer resolution problem, the problem with inline annotation is that often the original resource cannot easily be recovered just by removing the annotation. Difficulties arise for example when information like the placement of whitespaces, line breaks or other formatting information has not been made explicit in the annotated resource. Another difficulty with inline formats are overlapping annotations, for example, when different annotation schemes are applied to the same data, or if formatting information such as page segmentations overlap with the annotation of linguistic units, e.g. sentences. Moreover it is more difficult to work in parallel with the same resource, as new annotation layers would have to be inserted into the same source or document.

Stand-off annotation provides for more sustainability and flexibility: each annotation layer is encapsulated and can coexist with alternative or even conflicting versions of the same type of annotation.

Although more and more projects prefer the usage of stand-off annotation for sustainability reasons, there are still cases where inline annotations are needed. Natural language processing tools need to take into account more (structural) information when working on stand-off data which may have an impact on processing time – especially if large amounts of data have to be processed.

While CLARIN-D recommends the use of stand-off annotations for newly annotated resources, existing resources making use of inline annotations can also be hosted by CLARIN-D center repositories. Annotations should be accompanied by thorough documentations and, where possible, follow well-established practices (e.g. TEI, PTB format etc).

An important representation format in CLARIN-D is the *text corpus format* (TCF, see [Heid et al. 2010]) as an intermediate format in web service chains. When entering a chain of annotation tools in the WebLicht platform (see Chapter 8, *Web services: Accessing and using linguistic tools*) the input is firstly converted into TCF, which is an example for a combined format utilized in CLARIN-D.

In TCF the input text and its annotations are stored in the same document (technically: one file). The segmented tokens are provided with identifiers (if they are not already present) stored in a separate section of the file. They do not contain explicit references into the input text. Higher layers of annotation, e.g. part-of-speech annotations, are also stored in separate sections of the

file and refer to the token identifiers. This way, input text, segmented text and annotations can be handled separately in the outcome of an annotation chain. It is important to provide the possibility to represent the original text as well as the token layer, so that the possible tool chains or future web services are not restricted by the input layers they can choose from. See Example 3.5, “TCF corpus format” for an illustration of the use of stand-off annotation in TCF.

Example 3.5. TCF corpus format

```
<TextCorpus xmlns="http://www.dspin.de/data/textcorpus"
  lang="de">
  <text>Er fährt ein Auto.</text>
  <tokens>
    <token ID="t1">Er</token>
    <token ID="t2">fährt</token>
    <token ID="t3">ein</token>
    <token ID="t4">Auto</token>
    <token ID="t5">.</token>
  </tokens>
  <sentences>
    <sentence ID="s1" tokenIDs="t1 t2 t3 t4 t5" />
  </sentences>
  <POStags tagset="STTS">
    <tag tokenIDs="t1">PPER</tag>
    <tag tokenIDs="t2">VVFIN</tag>
    <!-- ... -->
  </POStags>
</TextCorpus>
```

This example is encoded in TCF version 0.4.

1.2. Multi-layer annotation

As already demonstrated iabove, it is possible to have different layers of annotations attached to the same primary data. Example 3.6, “TCF corpus format, extended” shows an extension of the TCF example above with an additional layer containing annotations of base form(s) and a syntactic annotation layer, each encapsulated and referring to the token layer. In this case all annotation layers refer directly to the token layer, but there are also cases where one annotation layer refers to another one lying "beneath" it. In these cases, one annotation layer immediately depends on the other annotation layer.

Example 3.6. TCF corpus format, extended

```
<TextCorpus xmlns="http://www.dspin.de/data/textcorpus"
  lang="de">
  <text>Er fährt ein Auto.</text>
  <tokens>
    <token ID="t1">Er</token>
    <token ID="t2">fährt</token>
    <token ID="t3">ein</token>
    <token ID="t4">Auto</token>
    <token ID="t5">.</token>
  </tokens>
```



```
<sentences>
  <sentence ID="s1" tokenIDs="t1 t2 t3 t4 t5" />
</sentences>
<POStags tagset="STTS">
  <tag tokenIDs="t1">PPER</tag>
  <tag tokenIDs="t2">VVFİN</tag>
  <!-- ... -->
</POStags>
<lemmas>
  <lemma tokenIDs="t1">er</lemma>
  <lemma tokenIDs="t2">fahren</lemma>
  <!-- ... -->
</lemmas>
<parsing tagset="tigertb"><parse>
  <constituent cat="TOP">
    <constituent cat="S-TOP">
      <constituent cat="NP-SB">
        <constituent cat="PPER-HD-Nom"
          tokenIDs="t1"/>
      </constituent>
      <constituent cat="VVFİN-HD" tokenIDs="t2"/>
      <!-- ... -->
    </constituent>
    <!-- ... -->
  </constituent>
</parse></parsing>
</TextCorpus>
```

The tagsets utilized here are STTS for the part-of-speech annotation and the tags from the syntactic annotations of the TiGer treebank, see [Brants et al. 2002].

1.3. Relations between annotation types

Introducing different layers of annotations means to introduce relations, and sometimes also dependencies between the layers. Nearly each linguistic annotation depends on a layer that represents the results of a segmentation process. These segmentation layers therefore define the parts of the resource which can be annotated. The ability to include specific annotations depends on the granularity of the segmentation. For example, part-of-speech annotation can not be carried out if the smallest parts to add annotations to are sentences; syntactic trees refer to word-like terminals rather than to phonemes. If two single annotation layers refer to the same segmentation layer, they can be related to each other via the segmentation layer.

In multi-level annotation, annotation layers depend on the content and integrity of the layer they refer to. If a layer is changed in any way, e.g. due to manual corrections, a layer referring to it might become invalid, as the parts to which its annotations referred might have changed or be gone.

Similar problems arise, when primary data is corrected. An annotator may not harm the syntax when correcting misspellings in a treebank, but other annotators might have been annotating exactly those misspellings in the course of investigating types of misspellings in newspaper text. On the other hand, corrections might sometimes be needed to make the data processable in the first place, e.g. in the case of slips of the tongue in spoken text which might cause problems for a parser. Therefore it is advisable to either apply versioning mechanisms for the primary data or to

introduce a new layer for each modification of the primary data. As already existing annotation layers may not apply to the new version of the primary data or to the modification layer, it is important to explicitly specify the version or modification layer an annotation refers to. In the example on misspellings in newspaper texts, an additional layer could contain a normalized view of the primary data to include error corrections.

Like in representation formats, where stand-off annotation provides for higher flexibility, it can be helpful to keep each layer as self-sustaining as possible. Nevertheless, a specific layer of annotation often implies the existence of other annotation layers. E.g., in the annotation scheme of [Riester et al. 2010] for information status, the hierarchical structure directly relies on a constituent-based syntactic layer.

In automatic processing, dependencies also arise due to the tools which are utilized, see also Chapter 7, *Linguistic tools*.

As resources are to be used for more than one purpose it is always helpful to make the relations and dependencies explicit. In multi-layer stand-off annotation this should be done by including dependencies or versioning information into the metadata for each annotation layer. On top of that, before relying on another annotation layer for a new annotation one should check if the underlying annotation covers all of the resources needed as there are annotations which do not take every part of a resource into account, e.g. because they do not cover punctuation, or because some parts have not (yet) been annotated.

2. Exchange and combination of annotations

Nearly every resource (corpus or automatic annotation tool) comes with its own annotation scheme, one or more tagsets, and one or more possible representation formats. However, this multitude of formats, schemes, tagsets and underlying categorial distinctions is a serious obstacle to the further processing and querying of heterogeneous resources, which is a likely scenario within a research infrastructure like CLARIN-D.

In this section, we present an example of a generic exchange format for annotations, which can be used with respect to different representation formats, we refer to the data category registry from Chapter 1, *Concepts and data categories* to relate tagsets and we give an overview on how the transfer of annotation schemes between different concepts has been handled so far.

2.1. Representing and exchanging complete annotations: getting independent of a specific representation format

Due to a number of different representation formats for annotations, varying from basic inline formats to task-specifically tailored XML descriptions, the exchange of annotated data often poses a problem. Therefore, structurally generic representations, which do not implement a preference for a specific linguistic theory, are needed to serve as pivot formats in the conversion procedure from one specific representation into another one, such as formats proposed by TEI and ISO. For a complete overview of all standards exploited in CLARIN-D and their relations see the CLARIN-D standards guidance web page [<http://clarin.ids-mannheim.de/standards/>]. Since direct conversion often means to build a lot of single converters and on top

of that probably also loss of information, an exchange format should be able to keep as much information as possible from the input while only passing the relevant parts to the output format.

In the following section we describe one of those possible pivot formats, the *Linguistic annotation format* (LAF, [ISO 24612:2012], see also [Hinrichs/Vogel 2010]). LAF uses a simple graph structure as data model for the stand-off annotation layers which is generic enough to handle different annotations. Some examples for annotation layers such as part-of-speech and syntax are provided in the next sections. Other annotation layers can be represented as well. LAF is also an ISO standard and therefore fits well into the CLARIN objective of taking standards into account.

2.1.1. LAF – the Linguistic Annotation Framework

LAF is developed by the ISO technical committee 37 (ISO/TC 37/SC 4), and is related to other standards developed within this group such as [ISO 12620:2009] on specification and management of data categories for language resources and [ISO 24610-1:2006] on feature structure representation. LAF provides a data model to represent primary data and different kinds of linguistic annotations without commitment to one particular linguistic theory. It is based on stand-off annotations where each annotation layer may contain references to elements in other layers.

There is always at least one segmentation layer, which, as a result of a segmentation process, defines the parts of the resource that may be further annotated. As different annotations may refer to the resource with different granularity, it is also possible to have concurrent segmentation layers for the same primary data.

While trying to leave the original resource untouched, the encoding plays a crucial role for the references into the primary data. Positions in the primary data are for example defined in between byte sequences denoting the base unit of representation. These positions are referred to as *virtual nodes*. The byte sequence defines the minimal granularity for the parts to be annotated. For textual data (UTF-8 encoded by default), virtual nodes are located between each pair of adjacent characters of the primary data document. *Anchors* refer to these virtual nodes. Example 3.7, “LAF virtual nodes” shows the virtual nodes defined between the characters.

Example 3.7. LAF virtual nodes

	h		i		s			o		w		n			w		e		b		s		i		t		e	
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5													

Node numbers are to be read top-down. Anchors to node 568 and 575, e.g., denote the token *website*. Anchors to the nodes 567 and 568 denote a whitespace.

LAF stipulates an XML-serialization as pivot format, which currently focuses on textual data. Nevertheless, by utilizing the flexible anchor management for references into primary data, LAF is designed to handle other types of media as well, e.g. audio data referenced by timestamps.

2.1.2. GrAF – a graph-based XML-serialization for LAF

The XML-serialization of the LAF pivot format is called *Graph annotation format* (GrAF). It is based on graph data structures and includes nodes, edges, regions for referencing primary data,

and annotations which can consist of simple labels or complete feature structures (see [ISO 24610-1:2006]). Therefore annotations at different levels of complexity can be represented [Ide/Suderman 2007].

The following examples of a segmentation and some annotation documents are extracted from the MASC I Corpus [<http://www.anc.org/MASC/Home.html>]. MASC structure and annotation details are taken from the MASC documentation website [http://www.anc.org/MASC/MASC_Structure.html].

In a segmentation, regions of primary data are denoted by anchors. In Example 3.8, “LAF reference to virtual nodes” the anchors refer to the virtual nodes in example Example 3.7, “LAF virtual nodes”, so seg-r194 denotes the token *his*, seg-r196 the token *own* and seg-r198 the token *website* respectively.

Example 3.8. LAF reference to virtual nodes

```
<region xml:id="seg-r194" anchors="560 563"/>
<region xml:id="seg-r196" anchors="564 567"/>
<region xml:id="seg-r198" anchors="568 575"/>
```

In an annotation document, nodes, edges and annotations can also be specified. A terminal node, i.e., a node with a direct reference to the primary data, references a region with a link and the respective annotations reference the node or edge element they belong to. Non-consecutive parts of the primary data can be annotated by introducing a region for each part and referencing them conjointly in the graph structure layered over the segmentation. Example 3.9, “LAF part-of-speech annotation” reproduces a part-of-speech annotation from the Penn Treebank project of the region denoting *website*.

Example 3.9. LAF part-of-speech annotation

```
<node xml:id="ptb-n00198">
  <link targets="seg-r198"/>
</node>

<a label="tok" ref="ptb-n00198" as="PTB">
  <fs>
    <f name="msd" value="NN"/>
  </fs>
</a>
```

Edges in annotation documents denote their source node and target node with *from* and *to* attributes. The node referenced by the edge attributes can also be defined in another annotation document. In this case the annotation document containing the edge depends on the annotation document containing the referenced node. In Example 3.10, “LAF syntactic annotation” *his own website* constitutes a noun phrase, i.e., the category *NP* is annotated to the node with the identifier *ptb-n00195*. This syntactic annotation (from the Penn Treebank project) depends on the part-of-speech annotation in Example 3.9, “LAF part-of-speech annotation”. The node *ptb-n00198* from Example 3.9, “LAF part-of-speech annotation” is annotated as *NN*, and referenced as the target of the edge in Example 3.10, “LAF syntactic annotation”.

Example 3.10. LAF syntactic annotation

```
<node xml:id="ptb-n00195"/>
```

```
<a label="NP" ref="ptb-n00195" as="PTB">
  <fs>
    <f name="cat" value="NP"/>
  </fs>
</a>

<edge xml:id="ptb-e00192" from="ptb-n00195" to="ptb-n00198"/>
<!-- website -->
```

Example 3.11, “LAF event annotation” reproduces an annotation for events produced by researchers at Carnegie-Mellon University. This annotation excerpt refers to the example sentence: *He said companies [...] would be able to set up offices, employ staff and own equipment [...]*. It denotes a *setting up* event with two arguments.

Example 3.11. LAF event annotation

```
<region xml:id="ev-r4" anchors="894 900"/>

<node xml:id="ev-n4">
  <link targets="ev-r4"/>
</node>

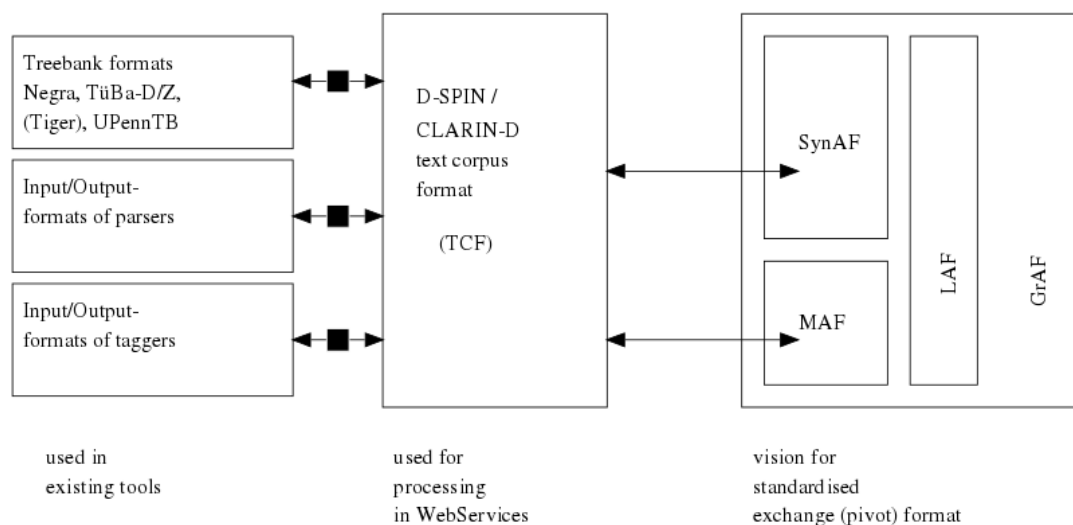
<a label="Setting Up" ref="ev-n4" as="xces">
  <fs>
    <f name="arg1" value="companies"/>
    <f name="arg2" value="offices"/>
  </fs>
</a>
```

We describe the LAF/GrAF framework here, because it can take different annotation layers into account. Within ISO technical committee 37 (ISO/TC 37/SC 4) there are also standards (and upcoming standards) related to specific annotation layers, such as the Syntactic annotation framework (SynAF, [ISO 24615:2010]).

Users can transform their data format into the LAF pivot format GrAF, and from GrAF into any other format for which an importer or exporter exists. There is ISO GrAF [<http://sourceforge.net/projects/iso-graf/>], an experimental Java API provided to access and manipulate GrAF annotations, and a renderer for the GraphViz visualization. [<http://www.graphviz.org/>]

2.1.3. The 3-layer-framework of representations to include, process and exchange

In terms of the WebLicht tool chains (see Section 3, “WebLicht – A service-oriented architecture for linguistic resources and tools”), internal representations of existing tools should not have to be changed in order to integrate them into the architecture. A 3-layer-framework, which differentiates between tool format, processing format and exchange format is proposed. Figure 3.1, “The 3-layer-framework” outlines the idea.

Figure 3.1. The 3-layer-framework

To make a web service out of an existing tool, the tool is encapsulated in a wrapper, see Chapter 8, *Web services: Accessing and using linguistic tools*. While the tools still work internally using their own representations, the wrapper provides an interface converting the output into a common processing format for the web service chains, e.g. TCF in the case of WebLicht. Note that a processing format has to fulfill other requirements than an exchange format, being as efficient as possible while staying as faithful to the original content as necessary. After the web service chain has been processed the data could again be converted into an exchange format taking standard formats such as LAF or TEI into account.

2.2. Introduction and monitoring of data categories: relating specific tagsets

The counterpart for a generic exchange format with respect to annotation representation is a data category registry for annotation content, i.e., for defining relationship between the categories used in the different tagsets. The problems which arise with this approach will be shortly discussed here. For a more detailed discussion see Chapter 1, *Concepts and data categories*, which describes the objectives of a data category registry and introduces ISOcat [<http://www.isocat.org/>] as the data category registry utilized in CLARIN-D.

In the case of different tagsets, e.g. STTS for annotating part-of-speech in German data or the Penn Treebank tagsets for part-of-speech and syntactic phrases with respect to (American) English, we might have to deal with a number of mismatches or “misunderstandings” due to different conceptions. Some examples are granularity, existence or duplication. Tagsets may differ in granularity. E.g. in STTS, there are two tags for nouns, one for common nouns and one for proper names, while in the Penn Treebank tagset nouns are additionally classified according to their number (four tags). If we compare different tagsets we may come upon cases where a certain category will be present in one tagset, but completely absent in another. Please note that we are not comparing linguistic concepts here, but sticking to the actual information annotated. Duplication means that there are the same tags, meaning something different or different tags meaning the same. For example, according to the granularity example above, NN is a common noun in STTS but a non-plural common noun in terms of the Penn Treebank tagset.

2.3. Handling different concepts: issues in transferring annotation schemes

While generic representation formats allow for an exchange of annotated resources and data category specifications help relating the semantics of single tags, much more needs to be considered when one annotation should be completely transformed into another one which relies on a different annotation scheme. This usually comes at the cost of either losing information or being forced to generate new information.

A prominent example is the conversion from constituency-based into dependency-based syntax. While on the one hand some information may potentially be lost in the process, on the other hand the heads in dependency-based annotation have to be explicitly specified in the transformation. As they cannot always be clearly identified, e.g. in English noun phrases containing compounds, additional procedures are needed. Head identification can then be conducted manually or by utilizing heuristics, see [Yamada/Matsumoto 2003], [Collins 1997], [Magerman 1994]. A heuristic approach may add errors to the annotation. However, as such conversions are often done in terms of shared tasks or statistical parsing approaches, a large amount of data is needed, which makes a manual approach rather impractical.

Another approach to large-scale mapping of annotation schemes considers ontologies of annotations such as in [Chiarcos 2008]. The ontologies again can be used to link to data category registries, e.g. ISOcat, see [Chiarcos 2010].

3. Recommendations

In this section only a few very general recommendations can be given, as the decision how to annotate a resource or which annotation to use highly depends on the data, annotators, available tools and on the task at hand.

The most general but very important recommendation is to provide metadata for each annotation of a resource. For already existing resources, not all of the relevant information might still be known or can be reconstructed, but as much as possible of the following information should be provided together with the annotation:

- information on how the annotation was created, i.e., manually, semi-automatically or automatically,
- the annotation guidelines, in case of a manual or semi-automatic annotation,
- the information which tools and tool versions have been applied in an automatic or semi-automatic annotation,
- the tagset and the explanations of the tags, preferably linked to a data category registry such as ISOcat,
- information about the quality of the annotations, e.g. inter-annotator agreement or scores of automatic processing tools, when applied to gold standards,
- explicit dependencies, e.g., if an annotation relies on other annotation layers or a specific version of the primary data,
- version information that identifies the existing annotation and allows for differentiation of versions

When creating a new set of annotations, it is also recommended to opt for sustainability and reusability, for example by choosing a state-of-the-art representation format and a common encoding such as UTF-8 for textual data. It is also recommended to choose a fine-grained basic segmentation that fits the annotation but also allows for other annotations to build on the same segmentation layer.

Chapter 4. Access to resources and tools – technical and legal issues

Erik Ketzan, IDS Mannheim

Ingmar Schuster, Universität Leipzig



editorial note

This chapter is currently being revised. Further information will be added.

In this chapter we will deal with legal and directly related technical issues of accessing language resources. You will learn how to access or obtain a resource once you have discovered it, e.g. via the Virtual language observatory (see Section 7, “Aggregation”).

Whether you are allowed to access a particular resource and what you are allowed to do with it depends on both your role in the academic system or elsewhere and on attributes of the resource which you want to obtain. Your rights with regard to a particular resource might be more or less restricted depending on your affiliation to a certain institution and your role in this institution (professor, student, fellow etc.). It might also be restricted by the attributes of the resource itself, in particular the restrictions that the owner or author of such a resource attributes to it. Even if you are at the “right” institution and in the “right” position, you might be allowed to use a resource personally, for your research, but not be allowed to change or redistribute it. You might not be allowed to use the resource (e.g. a corpus) as a whole, but you might be allowed to use a derivative work, e.g. a wordlist. The former aspects are dealt with in more detail in Section 1, “Single Sign-on access to the CLARIN-D infrastructure”, where we describe the CLARIN-D policies and schemes for the technical access of distributed resources. The latter aspect, i.e. legal and ethical restriction which are attributed to resources and their use, is dealt with in Section 2, “Legal Issues”.

1. Single Sign-on access to the CLARIN-D infrastructure

The *single sign-on* (SSO) infrastructure enables CLARIN-D users to access electronic resources from several institutions without applying for a large number of individual logins and the need to administer these logins. A single username/password combination, usually provided by the users home institution, makes sure that the scientific community gains access to certain distributed resources while the general public does not.

This is especially useful in the case of data with legal restrictions. These can include privacy issues in the case of experimental data from psycholinguistic experiments or copyright restrictions in the case of corpus and computational linguistics, ancient history, archaeology and other fields of research. In case a CLARIN-D user wants to grant others access to legally restricted resources, the SSO infrastructure will be of great use as well. Even the case where only certain groups in the scientific community are legally allowed to access a resource are handled by the SSO infrastructure. If for example participants in a psycholinguistic experiment agreed to have their data shared with PhD candidates and researchers, but not undergraduate students, the SSO infrastructure can ensure this legal restriction is enforced.

1.1. Gaining access to a resource

To gain access to resources, the CLARIN-D user's home institution has to maintain an identity provider (IdP) that stores username/password combinations (or other means of authentication information used with smartcards etc.) as well as additional attributes about a user. Most often this will be the Shibboleth [<http://shibboleth.net/>] software with attribute representation using SAML [<http://saml.xml.org/>] (security assertion markup language). The set of attributes typically comprises the affiliation status within the institution, email address or an anonymized unique identifier. To learn whether an institution runs an IdP and get their login information, users should ask local IT services.

In case the CLARIN-D user's home institution does not run a SAML IdP and will not be able to do so in the mid term, CLARIN-D has a fallback solution. Users can register with the CLARIN IdP [<http://www.clarin.eu/page/3398>] to access resources until their home institution has deployed their own IdP.

Once login information is provided either from a home institution or from the CLARIN IdP fallback solution, resources protected by the SSO infrastructure can be accessed. If the user is not legally allowed to access a resource (e.g. because the user is an undergraduate student while the resource can be viewed only by senior researchers) the hosting institution can deny access even for correct authentications.

1.2. Granting access to a resources

To protect resources that are distributed using the SSO infrastructure, the CLARIN-D user's home institution has to run a SAML *service provider*. A service provider (SP) can be configured according to the conditions which qualify a person to view a resource. These could be their home institution (e.g. researchers and undergraduates from University A can access the resource, undergraduates from University B cannot), the status within their institution (e.g. undergraduate student, senior researcher), or some other criteria. In the future, access can even be restricted to particular users. To learn whether an institution runs an SP and for steps to protect resources, users should ask their local IT service.

In case the CLARIN-D user's home institution does not run a SAML SP and will not be able to do so in the mid term, CLARIN-D centers can be contacted to explore the possibility of hosting and protecting resources with the server infrastructure they provide.

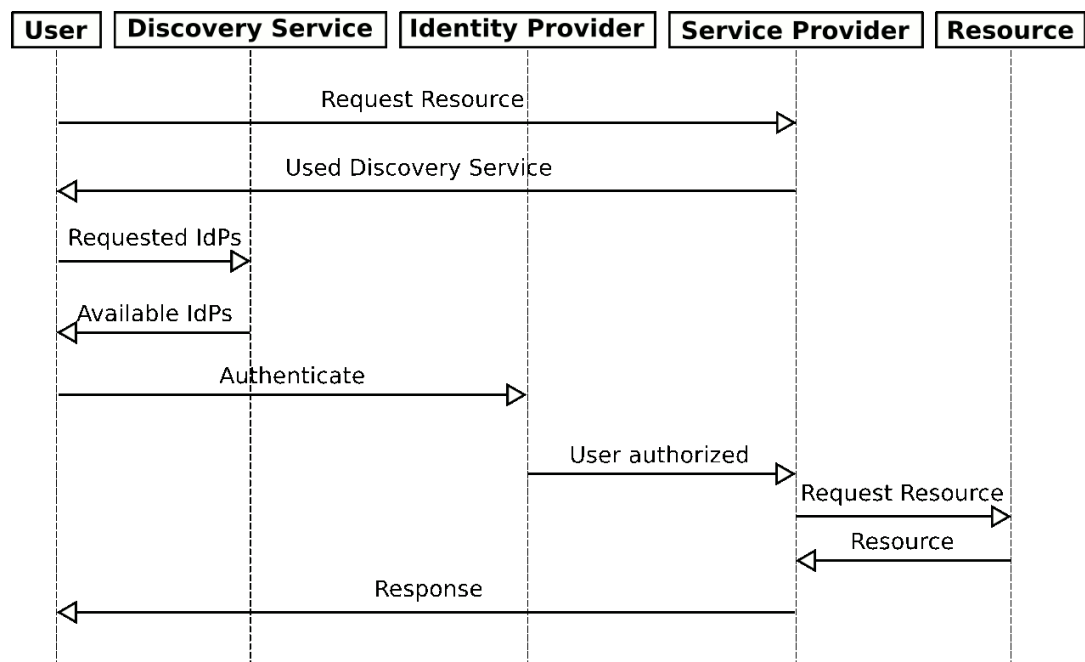
1.3. Technical details of the single sign-on infrastructure

The IdP and SP of an institution are expected to be part of the SAML federation built by the national research infrastructure organization. In Germany this is the Deutsches Forschungsnetz (DFN) and the DFN-AAI. It might make sense to additionally deploy a *discovery service* (DS) which assists users in choosing an appropriate IdP. Although the central CLARIN DS [<http://www.clarin.eu/page/3496>] can be used a DS at the home institution will come in handy in case the central service fails or has a malfunction.

Figure 4.1, “Authentication sequence” shows the communication flow running in the background when a user authenticates to access a resource. When a user request a resource from the SP, it responds with a list of accepted IdPs in the form of a DS. The user selects her home institution and authenticates against the home institution's IdP. If the authentication is

successful, the IdP tells the SP so and the resource can be accessed (provided the user authorized to use this resource).

Figure 4.1. Authentication sequence



2. Legal Issues

Both developers and users of CLARIN-D must contend with a variety of legal issues and challenges. The overriding message for now is that everyone should make legal decisions in communication with their institutions, directors, institutional lawyers, and the CLARIN-D legal help desk.

The goal of the legal help desk is to provide useful legal information to scientists so that they and their institutes may make better informed decisions. The help desk is constantly updating its legal information platform [<http://de.clarin.eu/en/training-helpdesk/legal-helpdesk.html>] that serves as an evolving best practices guide for researchers and developers. It covers issues relating to licenses, legal implications for data collection (e.g. speech, text, etc.), legal issues for resource creation (building corpora from various works, including online texts and online forums), and legal issues for tools (e.g. can I have them run in a data center?).

Additionally, the legal help desk provides extensive written memos, plus email and telephone support, on these and other cutting-edge issues in copyright, privacy law, and other areas that affect language technology.

Contact the legal help desk by email via its staff at:

- Erik Ketzan <mailto:ketzan@ids-mannheim.de>
- Pawel Kamocki <mailto:kamocki@ids-mannheim.de>

Chapter 5. Quality assurance

Heike Zinsmeister, Universität Stuttgart

One important goal of CLARIN-D is to support the reuse of language resources, in particular corpora, lexical resources and tools. Quality assurance is an essential pre-requisite for the reuse of resources. What does (high) quality mean with respect to language resources? First of all, it means that resources are described in a way that the users know what to expect from them, so that they can judge whether a particular resource is suitable for their purposes. The answer to the question, what to expect from a resource, touches several domains discussed thoroughly in other chapters of this book, most notably in Chapter 2, *Metadata* and Chapter 3, *Resource annotations*. In this chapter, we will mainly concentrate on the issue of inherent quality of a resource in terms of well-formedness, (linguistic) adequacy and consistency. After that we will conclude with a short section considering metadata.

1. Aspects of the quality of resources

1.1. Well-formedness and schema compliance

Corpora and lexical resources are considered well-formed if they conform to a particular defined format. This does not refer to linguistic concepts but the general document grammar as specified, for example, in an XML *document type definition* (DTD) or *schema*. The schema determines the structure of the document in terms of eligible markup. If an XML document conforms to such a specification, it is said to be *valid* with regard to this DTD or schema. One method of quality assurance is to make any such a specification available together with the resource.

1.2. Adequacy and consistency

A resource is *adequate* only with respect to a particular theory or a particular application. It is therefore essential to provide information about these. In the case of the linguistic annotation of corpora – and also to a certain extent for lexical resources – this is normally done by means of detailed guidelines that specify the tagset and define the (linguistic) concepts on which the tagset relies. Ideally, the guidelines also provide linguistic tests that human annotators can apply, and discussions of problematic cases, which make the annotation decisions comprehensible.

With respect to annotated resources, *consistency* “[...] means that the same linguistic phenomena are annotated in the same way, and similar or related phenomena must receive annotations that represent their similarity or relatedness if possible” [Zinsmeister et al. 2008, page 764]. Consistency is undermined by two major factors: ambiguity and vagueness in the data, on the one hand, and errors made by annotators, on the other hand. Errors made by annotation tools normally are of a consistent nature.

One should be aware that any annotation contains errors. In addition to errors performed by human annotators basically any tool which makes decisions on how to annotate a resource will make some mistakes and analyze certain instances of the data inappropriately. Therefore, the expectation is that some sort of *evaluation of the annotation quality* is provided together with the resource. For automatic annotations tools this quality information can be based on evaluating the tools on some *gold standard* – a resource that has been approved to be adequately annotated – and report standardized scores such as precision, recall, and F-measure (or other task-related measures as developed in shared tasks such as the CoNLL shared task initiative

[<http://ifarm.nl/signll/conll/>]. Nevertheless, one has to keep in mind, that also gold standards contain debatable decisions. Each processing step in course of creating a corpus is a case of interpretation. Even transcribing non-digitized texts is subject to interpretation if the quality of the master copy is poor, as it is sometimes the case with historical data, or if an originally hand-written scripts is unreadable.

For manual annotations it is recommended to let more than one annotator work on the same part of the result, calculating the inter-annotator agreement afterwards [Artstein/Poesio 2008]. The same holds for manual transcription of non-digitized text. In Digital Humanities, it is standard to employ the *double-keying* method in which two operators create two independent transcriptions, which are subsequently compared, see [Geyken et al. 2011].

From results of *inter-annotator agreement*, conclusions can be drawn on the ambiguity of either the guidelines, see [Zinsmeister et al. 2008], or the phenomenon to be annotated. The guidelines can be specified in some sort of bootstrapping approach, thereby also changing the tagsets, when the results of different annotators imply that two categories can not be clearly differentiated.

Next to trying to improve tools in inspecting their results, or by training them on tailored resources and adapting the features they use, there are also some mechanisms working on the annotated resources. For example an automatic processing software to find inconsistencies in text corpora taking part-of-speech tagging, constituent-based and dependency-based syntactic annotations into account has been developed within the DECCA project [<http://decca.osu.edu/>] [Boyd et al. 2008].

1.3. Metadata

All metadata describe the resource they belong to in some way or other and contribute to qualify it for reuse.

For assessing the inherent quality of a resource in particular it is relevant that the metadata keeps record of all processing steps the resource has undergone. This includes information about the method applied (e.g., in terms of guidelines) and the tools used. For all manual annotation steps, including transcription and correction, the metadata should provide information about inter-annotator agreement to specify the reliability of the resource as requested in Section 1.2, “Adequacy and consistency”. In the case that the resource is a tool itself, the metadata ideally also include gold standard evaluation or shared-task results for the same purpose. The latter is not yet part of standard metadata sets. Very often this kind of consistency and performance information is only indirectly provided in terms of reference to a publication, which includes a report on these measures. However, providing a reference to a publication on the resource is in itself part of high-quality metadata. It equips the user with a standard reference to be cited when the resource is used.

High-quality metadata for tools will specify the requirements on the input format of the data the tool is applied to, and it will also inform about the output format. In a similar vein, metadata for corpora and lexical resources might recommend tools that can be used to query, visualize or otherwise process the resource – if it is developed in a framework that offers such tools.

2. Recommendations

Resource providers should release the schema(ta) which formally describe(s) the format of their resource together with the resource itself. The formal account is ideally accompanied by

an informal description of the format or refers to well-known standards and practices (e.g. in the case that TEI is used).

Resource providers should add a description of a typical use of of the range of uses which the resource has been used for or is usable for.

Resource providers should be as explicit as possible with respect to potential errors and the expected rate of such errors in their resource. Hints to regular patterns of error might also be useful for the potential users. For example, if an automatically part-of-speech annotated corpus is known to not well distinguish two categories which are distributionally similar, a user can decide to ignore this distinction altogether and map both classes into one more general category.

Resource providers should describe the applied methods in the case of manual annotation of a result, e.g. how many annotators, the annotation manual used, rate of inter-annotator agreement, resolving methods in the case of disagreement etc.

Users of a resource should be as clear as possible about what they are looking for. If they are looking for a resource, they should consider

- whether a large set of data with potential errors is preferred, or a small data set which is as accurate as possible;
- whether an annotation with high granularity (i.e. many categories) and many potential errors is preferred or a coarse-grained annotation with a high rate of annotation accuracy.

Users of a resource should check whether the accompanying documentation, schema etc. meets their information needs. If this is not the case, they should come back to the resource provider or the distributor with a precise request for additional information.

Users of resources should take some time to find and compare different resources with respect to their specific requirements.

Part II. Linguistic resources and tools

Table of Contents

6. Types of resources	51
1. General recommendations	51
2. Text Corpora	52
2.1. Background	52
2.2. Text format	56
2.3. Metadata	61
2.4. Summary and Recommendations	61
3. Multimodal corpora	61
3.1. Examples of possible modes within corpora	62
3.2. Some background on audiovisual data formats	62
3.3. Recommendations	64
4. Lexical resources	65
4.1. Introduction	65
4.2. Common formats	66
4.3. Formats endorsed by CLARIN-D	71
7. Linguistic tools	73
1. Hierarchies of linguistic tools	73
2. Automatic and manual analysis tools	75
3. Technical issues in linguistic tool management	77
4. Automatic segmentation and annotation tools	77
4.1. Sentence splitters	78
4.2. Tokenizers	79
4.3. Part-of-speech taggers	81
4.4. Morphological analyzers and lemmatizers	82
4.5. Syntax	86
4.6. Word sense disambiguation (WSD)	90
4.7. Coreference resolution and anaphora	90
4.8. Named entity recognition (NER)	90
4.9. Sentence and word aligners	91
5. Manual annotation and analysis tools	92
5.1. Manual annotation tools	92
5.2. Annotated corpus access tools	93
6. Multimedia tools	93
7. Recommendations for CLARIN-D tool designers	95
8. Web services: Accessing and using linguistic tools	96
1. Web Services	96
2. Service-oriented architectures	98
3. WebLicht – A service-oriented architecture for linguistic resources and tools.....	98
3.1. Tool chains	99
3.2. Interoperability and the Text Corpus Format	100
3.3. Visualization	101
3.4. Metadata	101
3.5. Security	101
4. WebLicht usage scenarios	102
4.1. Quick annotation	102
4.2. Statistics	111
4.3. Geovisualization	113
5. Integrating existing linguistic tools into WebLicht	114

Chapter 6. Types of resources

Axel Herold, BBAW Berlin

In the following sections we will provide a brief overview of the most commonly encountered types of linguistic resources ranging from text corpora and multimodal corpora to lexical resources. In Chapter 7, *Linguistic tools* a broad range of automatic segmentation, annotation and analysis tools is presented that provide working implementations of the methods already touched upon in Part I, “Basic concepts” and for many tasks beyond. Finally, in Chapter 8, *Web services: Accessing and using linguistic tools* we demonstrate how resources and tools can be dynamically connected in general and within the CLARIN-D infrastructure in particular.

We assume that readers are already familiar with basic data modeling and representation standards such as the extensible markup language (XML) [<http://www.w3.org/XML/>] or different character encodings representing Unicode [<http://www.unicode.org/>] or older legacy encodings. These standards are of great importance well beyond the linguistic domain and not covered in this user guide.

1. General recommendations

For a successful technical integration of linguistic resources and tools into the CLARIN-D infrastructure the following requirements must be met:

- All data should be stored in one of a limited set of data formats. Throughout this user guide we discuss all data formats available within the CLARIN-D infrastructure. A list of data formats and their status within the European CLARIN project can be found on the CLARIN EU website [<http://www.clarin.eu/recommendations>].
- All tools and resources have to be associated with persistent identifiers.
- Tools and resources have to be associated with comprehensive metadata. All data categories used in the resource itself or in its metadata description should map to a data category in ISOcat, or other CLARIN supported data category registries, like the ISO-3166 registry for country codes [ISO 3166-1:2006], [ISO 3166-2:2007]. In certain cases, entries in RELcat (which is still under development) may also be used. See Chapter 2, *Metadata* for details.
- A *formal description* of the resource's underlying data model must be provided. It serves as a means of formal documentation of the resource. It will also be used for formal validation if the resource is processed by a CLARIN-D member.

See Section 1.1, “Well-formedness and schema compliance” for a detailed account on formal descriptions for validation. In the case of XML based resources XSD, Relax NG schemata and Schematron rules are preferred over document type definitions (DTD). Formal descriptions should be documented and should contain links to a data category registry for all datatypes and their values. Closed sets of value data categories must be explicitly enumerated.

- An *informal documentation* of the resource targeted for the CLARIN-D user community must be provided. In the simplest case this might be an already existing freely available electronic article or whitepaper. The documentation should be provided in an English version and preferably also in the subject language(s) of the resource. Versions in additional languages are welcome, too.

- In the case of resources in formats that are not supported by CLARIN-D, resource providers are encouraged to contact the CLARIN-D technical help desk [<http://www.clarin-d.de/en/training-helpdesk/technical-helpdesk.html>] to find support in transforming their resource into a CLARIN-D compatible format.

Integration of some resources and tools may lead to additional requirements. These are discussed in the specific section on the resource or tool type.

2. Text Corpora

Alexander Geyken, Susanne Haaf, BBAW Berlin
Marc Kupietz, Harald Lungen, Andreas Witt, IDS Mannheim

This section addresses corpus linguists and corpus technologists who would either like to provide existing text corpora to the CLARIN-D infrastructure or want to develop new corpora for the CLARIN-D infrastructure. The central part of this section explains the recommended format for CLARIN-D compliant text corpora on two levels, text data and metadata, and motivates the reasons for this decision. In addition, a background chapter introduces to corpus linguists or other scholars interested in corpora how such resources are addressed by the CLARIN-D-infrastructure.

Why contribute to the CLARIN-D infrastructure? Users, who provide the CLARIN-D-infrastructure with CLARIN-D compliant text corpora, can benefit from the following features:

- CLARIN-D compliant metadata can be harvested and are therefore searchable throughout the CLARIN-D infrastructure. The benefit is: your data are more visible and easier to find for other researchers and therefore more widely used. See Chapter 2, *Metadata* for more details.
- CLARIN-D compliant texts that are provided with sufficient intellectual property rights (IPR) can be stored and maintained in the repositories of CLARIN-D service centers. The benefit is: CLARIN-D handles the distribution and access to your data while respecting their particular IPR situation.
- CLARIN-D compliant corpora (i.e. their text structure) are interoperable in the CLARIN-D infrastructure; i.e. they are fully searchable, several views (HTML, TEI, text) can be generated automatically at the CLARIN-D service centers, and third those corpora can be further processed by the CLARIN-D tool chains (see Chapter 8, *Web services: Accessing and using linguistic tools*). The benefit is: you are able to receive support for the (further) linguistic analysis of your data, including a ready to use box of tools. You can mix and merge your data more easily with data which are provided by someone else.

2.1. Background

Readers who are familiar with corpus typology, corpus compilation, and corpus quality can skip this section and go directly to Section 2.2, “Text format”. The following section gives a brief overview and is not intended to replace introductory works like [McEnery/Wilson 2001] for English and [Lemnitzer/Zinsmeister 2010] and [Perkuhn et al. 2012] for German. We also recommend to take a look at the *Handbook of linguistics and communication science*, vol. 2, *Corpus linguistics* [Lüdeling/Kytö 2009].

2.1.1. Corpus typology

A common characterization of the term *corpus* is given by John Sinclair, a pioneer of corpus linguistics: “A corpus is a collection of naturally-occurring language text, chosen to characterize a state or variety of a language” [Sinclair 1991, page 171]. The terms *corpus*, *text corpus* and *electronic (text) corpus* are used interchangeably in this section to refer to a collection of texts in machine-readable form.

Corpora of written text consist of several *corpus items* (for a discussion about the size of a corpus see [Sinclair 2005]). Each corpus item consists either of text samples (such as the Bonner Frühneuhochdeutschkorpus [<http://www.korpora.org/Fnhhd/>]) or an entire document (such as most of the corpora of the IDS, the DWDS-Kernkorpus or the Deutsches Textarchiv [<http://www.deutschestextarchiv.de/>] (DTA, German Text Archive), in a machine-readable form, annotated on various linguistic levels and enhanced with metadata (see Chapter 2, *Metadata*). The primary data is a transcription of the raw data (or source data, e.g. when the raw data is a non-digital asset, such as a book). These primary data can be annotated on different linguistic levels (see Chapter 3, *Resource annotations* for a detailed discussion about annotations and Chapter 1, *Concepts and data categories* for an account on data categories) resulting in a set of related files. Usually each single annotation layer of the primary data results in a new XML instance which is stored in a separate but related file. Apart from the annotation files, metadata files have to be included as well.

One commonly distinguishes the following types of text corpora with regard to data collection and language modeling:

- balanced reference corpora, i.e. text collections that are balanced over text genres such as the British National Corpus [<http://www.natcorp.ox.ac.uk/>] (BNC) or the DWDS-Kernkorpus [<http://www.dwds.de/>] [Geyken 2007],
- opportunistic corpora, and corpus archives that are designed to act as primordial samples for user defined virtual corpora that are balanced or representative with respect to the intended basic population such as the Deutsches Referenzkorpus [<http://www.ids-mannheim.de/kl/projekte/korpora/>] (DeReKo, [Kupietz et al. 2010]),
- closed corpora, i.e. text collections that provide the entirety of texts from a former language period or a single author, if this amount is comparatively small and strictly limited such as for the Old German texts in Deutsch Diachron Digital (DDD) [<http://www.deutschdiachrondigital.de/>].
- open corpora, i.e. text collections corresponding to well documented subsets of text taken from the period of time or discourse. As opposed to closed corpora open corpora are compiled if the amount of text for a certain period of time or for a certain discourse is too large to be presented in its entirety.
- specialized corpora, i.e. text collections containing a specific linguistic variety or sublanguage (e.g. a discourse, a certain text type) whereas general corpora aim at providing a representative section of the entire language material of a specific period of time (e.g. Old/Middle/Early New/New High German).

One may also find the terms *synchronic* and *diachronic* corpora: synchronic corpora represent the language used at a specific point in time, whereas diachronic corpora document the evolution of language over a certain period of time.

2.1.2. Corpus compilation

This section addresses the question of data compilation and distinguishes between capturing data from non-digital and from digital sources.

Capturing data from non-digital sources

The sources on which the recognition of corpus texts is based may differ. In some cases, such text recognition sources are the original works (printed works or manuscripts) themselves. Nevertheless, for practical reasons, transcriptions will generally be based on some kind of copy from the original whereas the original sources are only consulted in case of doubt. Copies may be paper copies, microfiches or microfilms. In most cases, however, they will be available in a digital format. The best case scenario (when dealing with copies of text sources) would be to base the transcription on a coloured image with high resolution (scan or photography) taken directly from the original, which should be available in a lossless image format (e.g. TIFF). The image quality and therefore the text recognition accuracy decreases the less of the named conditions apply, e.g.

- in case the source images are grey or bitonal scans,
- in case the source images are based on microfilms or microfiches, or,
- in case of insufficient image resolution of the source images.

High quality transcriptions of the text are a necessary prerequisite for the correctness and completeness of retrieval results. There are different methods of text recognition:

Optical character recognition (OCR)

With this method texts are recognized by optical character recognition software. While OCR may be suitable for modern texts in Antiqua performance decreases considerably with the increasing age of the text sources. Therefore, texts have to be proofread and manually corrected several times in order to eliminate recognition errors. The performance of the OCR may also be influenced positively by preparing the scanned images in advance, e.g. by labeling zones on a page with information about whether they contain text or images, whether text is written in columns etc.

In addition, OCR software may be trained for a particular kind of text source (e.g. sources printed at the same time and by the same printer). With training, OCR performance may be increased considerably. However, since training is quite costly, it is only practicable if there is a considerable amount of homogeneous corpus texts to be transcribed. It is too time consuming if the texts are of varying typefaces and structures [Tanner et al. 2009], [Nartker et al. 2003], [Furrer/Volk 2011], [Holley 2009].

Manual transcription

Manual transcription by one person with several proofreading run-throughs: With this method high accuracy rates may be achieved, especially if there are different correctors who are trained to work with historical texts. However, this method is time consuming, as texts have to be transcribed very carefully and corrected several times, in order to get good accuracy rates.

Double keying:

The double keying method requires two typists transcribing the same text, thus producing two independent versions of a text transcription. These two versions are then compared to one another and differences are evaluated by a third person. Given, that two typists are

unlikely to make the same mistakes, this method is highly accurate even for older historical texts. However, this transcription method is quite costly as well, since two versions of a text have to be produced. Training of the typists might improve the transcription accuracy especially for older texts with difficult typefaces [Haaf et al. forthcoming].

For text recognition, guidelines should be defined to regulate how to deal with special phenomena within the source text, where to apply normalizations or, in which cases of doubt to follow the source text. The text recognition guidelines of the DTA project [<http://www.deutschestextarchiv.de/doku/richtlinien>] (in German) are a good example.

Capturing data from digital sources

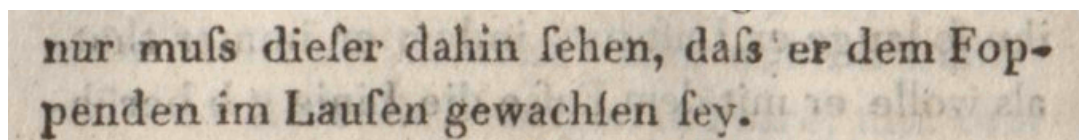
Written language data may already be digitized and come in different formats, e.g. as plain text, office software documents or in PDF files. If so, text data has to be converted into a common standardised format, which allows for further processing.

Spoken language lies in the intersection of written corpora and multimodal corpora since their transcriptions are provided in a written form whereas their underlying data correspond to digital audio or video recordings. For more technical details see Section 3, “Multimodal corpora”.

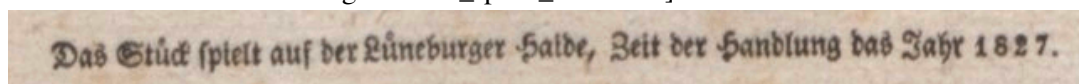
2.1.3. Corpus quality and quality control

Corpus quality can be assessed on two levels: transcription quality and correctness of annotations. Both should be dealt with in the data caption phase rather than in the correction phase. One example for a problem that might arise during data capturing are similar letters, where each interpretation leads to a correct word. This is a typical error that is very hard to detect after the transcription phase. Some instances of common errors of this class are shown in Figure 6.1, “Transcription errors leading to valid word forms”.

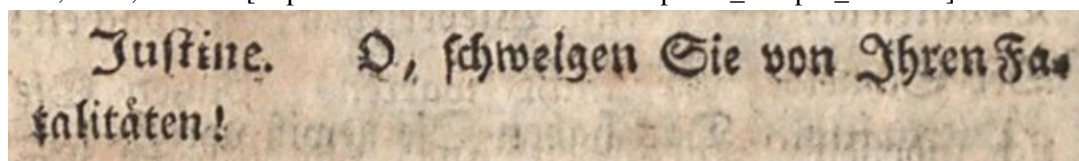
Figure 6.1. Transcription errors leading to valid word forms



“Laufen” (to delouse) vs. “Laufen” (to run), in: Johann Christoph Friedrich Guts Muths: Spiele zur Übung und Erholung des Körpers und Geistes. Schnepfenthal, 1796, IMG 270 [http://www.deutschestextarchiv.de/gutsmuths_spiele_1796/270]



“Haide” (heath) vs. “Halde” (dump), in: August von Platen: Der romantische Ödipus Stuttgart u. a., 1829, IMG 10 [http://www.deutschestextarchiv.de/platen_oedipus_1829/10]



“schweigen” (to remain silent) vs. “schwelgen” (to wallow), in: Friedrich Wilhelm Gotter: Die Erbschleicher. Leipzig, 1789, IMG 78 [http://www.deutschestextarchiv.de/gotter_erbschleicher_1789/78]

Similarly, misinterpretations on the annotation level are possible.

The Deutsches Textarchiv provides a detailed workflow of how to deal with possible annotation problems in the pre-caption phase [<http://www.deutschestextarchiv.de/doku/workflow>] and of how detailed transcription guidelines can help to avoid transcription errors.

For more detailed information on quality control issues consult Chapter 5, *Quality assurance*.

2.2. Text format

TEI-P5 (see Excursus: Text Encoding Initiative – TEI-P5) is the recommended standard to be used for CLARIN-D. TEI-P5 is a widely used and very flexible standard that is adoptable for a large variety of text types. Due to this flexibility of the “full” TEI-P5 tag set (`tei_all`) TEI-P5-compliant corpora are generally not interoperable per se. For a discussion of some of the problems arising from this fact see [Unsworth 2011] and [Geyken et al. 2012].



Excursus: Text Encoding Initiative – TEI-P5

For the semantic structuring of texts the TEI, a consortium of international scholars and researchers founded in 1994, provides a modular, theory-neutral annotation tag set accompanied by an elaborated description. The TEI guidelines are freely available and there is a wide range of tools (both open source and commercial) that can process TEI-annotated data. All these benefits make the TEI a good starting point for annotating linguistic corpora. P5, the current version of the TEI guidelines, was officially released in 2007.

The TEI offers solutions for the structuring of texts, considering as many demands of digital editions as possible. Corresponding to the purpose of XML the TEI encoding standard focuses on the semantic rather than the formal structuring of texts. Text structuring may remain on a rather superficial level or may take place on a deeper text level and even contain the addition of editorial information as comments or text critical remarks. The TEI P5 tag set consists of a limited number of elements, each of which encompasses a selection of specific attributes. On the level of attribute values only some recommendations are given. Apart from that users are free to use value names, which they consider to be the most suitable in their context.

This way, the application of the TEI tagset results in a broad variety of possible annotations, from which users may choose the most suitable ones for the annotation of the different and specific phenomena, which occur in the texts they are dealing with. However, this desirably wide range of annotation solutions leads to the problem, that for the treatment of one phenomenon different annotation approaches (each of which is accordant to the TEI) are possible, depending e.g. on the degree of specificity of the annotation. For instance it is possible to tag the name of a person with either the element `name` or the element `persName`. If the reading of a character is insecure, it may be annotated with a `supplied` or an `unclear` element. Therefore, projects need to narrow down the TEI tagset to their specific needs in order to avoid incoherence of annotation and therefore incompatibility across text collections of their texts on a structural level.

For this purpose it is possible to adjust the TEI schema to the needs of a specific project. The easiest way to do so is to create an ODD (short for: “one document does it all”) document, in which the designated adjustments to the TEI schema are specified. ODD is a special TEI P5 format which allows for customizations of the

TEI schema, such as excluding modules, elements, or attributes (element or class wise) from the schema, or defining particular attribute values. It is also possible to add new elements to a given tagset.

For the creation of a specific ODD file the web application Roma [<http://www.tei-c.org/Roma/>] may be used. Roma helps the user to define the project's specifications step by step. The resulting ODD file may be saved and, if necessary, modified further according to chapter 22 of the TEI guidelines [<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/TD.html#TDmodules>]. Based on the final ODD file a schema (RelaxNG, RelaxNG compact, XML Schema etc.) may then be generated using Roma. The TEI provides a tutorial on how to use Roma [http://www.tei-c.org/Guidelines/Customization/use_roma.xml].

Note, that the TEI itself offers several specifications of the TEI guidelines as recommendations for specific annotation purposes [<http://www.tei-c.org/Guidelines/Customization/>], due to the fact that there are prototypical phenomena which should be dealt with in a common standard way, such as TEI-Tite [http://www.tei-c.org/release/doc/tei-p5-exemplars/html/tei_tite.doc.html], TEI-Lite [<http://www.tei-c.org/Guidelines/Customization/Lite/>] or the Best practices for TEI in Libraries [<http://www.tei-c.org/SIG/Libraries/teinlibraries/main-driver.html>]. For the annotation of German texts which were originally published in print the DTABf offers a TEI specification, that is suitable for the different kinds of text structures found in printed texts.

Among the partners of CLARIN-D there are two formats that are widely used for a very large number of documents: the DTA base format [www.deutschestextarchiv.de/doku/basisformat] (DTABf) at the BBAW and IDS-XCES at the IDS. A TEI P5 conformant document grammar for IDS-XCES, called I5, is currently under development [Lüngen/Sperberg-McQueen 2012]. These formats are very likely to be supported in the future since software used by many thousand scholars is tailored to them. However, both partners intend to unify the formats in the coming months in such a way that a common TEI-P5 compliant format can be provided for CLARIN-D. In the meantime the CLARIN-D policy recommends both of the two different but TEI-P5 compliant schemas with less variation but more semantic control. They are discussed in more detail below focussing especially on the requirements and benefits for the integration into the CLARIN-D infrastructure.

As a consequence of the remarks made above, there are two main levels of compatibility with the CLARIN-D infrastructure:

1. Text corpora that are compliant with “full” TEI-P5: Those corpora will be interoperable with other resources in the CLARIN-D infrastructure on four sub-levels:
 - a. CLARIN-D compliant Metadata (CMDI profiles) can be generated from TEI-P5 headers (even though they might be underspecified).
 - b. Corpora can be sorted in the repository of the CLARIN service centers of either BBAW or IDS.
 - c. Corpora formats can be transformed into the TCF format (see Section 3.2, “Interoperability and the Text Corpus Format”), hence postprocessing of these documents in any of the web service based CLARIN-D processing pipelines (Weblicht) is possible.
 - d. Texts in TEI-P5 are searchable via the federated search component of CLARIN-D.

2. Text corpora that are fully compatible with one of the TEI-P5 subsets of CLARIN-D, namely DTABf or IDS-XCES: In addition to the four levels of interoperability above, documents in these formats share additional benefits with regard to the CLARIN-D infrastructure. See the DTA base format (DTABf) and IDS-XCES technical information in the boxes below for details and a discussion on how deviating TEI-P5 schemas can be integrated into the recommended schemas.



DTA base format (DTABf)

The DTA base format [<http://www.deutschestextarchiv.de/doku/basisformat>] draws on the works of the Deutsches Textarchiv [<http://www.deutschestextarchiv.de/>] (DTA) of the BBAW where a large reference corpus for a large variety of (kinds of) texts is currently compiled. The tagset of DTABf is completely compliant to the TEI P5 guidelines, i.e. no new elements or attributes were added to the TEI P5 tagset. It consists of about 80 TEI P5 elements needed for the basic formal and semantic structuring of the DTA reference corpus, plus another 25 additional elements used for metadata structuring in the TEI header. The purpose of DTABf is to gain coherence on the annotation level (i.e. similar structural phenomena should be annotated similarly), given the heterogeneity of text material as published in the DTA over time (1650–1900) and text types (fiction, functional, and scientific texts).

DTABf attempts to meet the criteria of interoperability mentioned by Unsworth in that it “focuses on non-controversial structural aspects of the text and on establishing a high quality transcription of that text” [Unsworth 2011]. Therefore, the goal of DTABf format is to provide as much expressiveness as necessary by being as precise as possible. For example DTABf is restrictive not only considering the selection of TEI elements but also with respect to attribute-value pairs and allows only a limited set of values for a given attribute. Unlike initiatives such as TEI-A [Pytlik Zillig 2009] the goal of DTABf is not to build a schema that validates as many cross collections as possible but to convert resources from other corpora so as to keep the structural variation as small as possible.

The necessity of a common standardized format for the annotation of printed texts seem to be opposed to the fact, that different projects usually have different needs as for how a corpus may be exploited. Therefore annotation practices vary according to the variable queries on a certain corpus. This problem may be addressed by defining different levels of text annotation, that represent different text structuring depths. The TEI recommendations for the encoding of language corpora [<http://www.tei-c.org/release/doc/tei-p5-doc/en/html/CC.html>] foresee four different levels of annotation defining required, recommended, optional, and proscribed elements.

DTABf consists of such annotation levels, which serve as classes subsuming and by that categorizing all available DTABf elements:

Required (level 1)

These elements are mandatory for the basic semantic structuring of corpus texts.

Recommended (level 2)

These elements are recommended for the semantic structuring of corpus texts and are systematically used in the DTA corpus.

Optional (level 3)

These elements do not need to be considered for text annotation. Level 3 elements are not (yet) part of the DTA guidelines and are therefore not used systematically in the texts of the DTA corpus. They are, however, compatible with the DTA schema.

Proscribed (level 4)

These elements were explicitly excluded from the DTA guidelines. They should be avoided in favor of the solutions offered in the DTA guidelines.

See the tabular overview of DTABf `text` elements and their corresponding levels [http://www.deutschestextarchiv.de/doku/basisformat_table].

DTABf is continuously adapted to the requirements of external corpus projects. These extensions are mainly community-based. Up to now more than ten external projects have contributed to the extension of DTABf. For a range of projects, their TEI based schemas were converted to DTABf [<http://www.deutschestextarchiv.de/dtae>], including:

- Blumenbach Online Edition [<http://www.blumenbach-online.de/>]
- some projects at the Max-Planck-Institut für Bildungsforschung [<http://www.mpib-berlin.mpg.de/de/institut/bibliothek/projekte>]
- Dingers Polytechnisches Journal [<http://dinglr.de>]

In addition, a CLARIN-D corpus curation project [<http://www.clarin-d.de/de/fachspezifische-arbeitsgruppen/f-ag-1-deutsche-philologie/kurationsprojekt-1.html>] by the F-AG 1 (“Deutsche Philologie”, German philology) with the project partners BBAW (coordinator), HAB, IDS, and University of Gießen pursues the goal of converting external corpora, which were created using many different formats, into DTABf and integrating those corpora into the CLARIN-D infrastructure. This project started in September 2012. It will be extensively documented and might serve as an example of good practice for your own work.



IDS-XCES

IDS-XCES [<http://www.ids-mannheim.de/kl/projekte/korpora/textmodell.html>] is an adaptation of the XCES corpus encoding standard [<http://www.xces.org/>] (see [Ide et al. 2000]). It has been the format for the semantic and structural annotation of the texts in the corpus archive of the German Reference Corpus DEREKO at the Institute for the German Language in Mannheim (IDS) since 2006. The XCES standard had been introduced in 2000 as a re-definition of the SGML-based corpus encoding standard CES [Ide 1998] in XML. CES, in turn, had been based on the likewise SGML-based P3 version of the TEI guidelines as an effort to define a subset of TEI P3 elements and attributes suitable for corpus annotation.

Subsequently, XCES had not been kept fully compatible with the XML-based successors of P3, i.e. TEI P4 and P5, so that as of today, it contains a few elements and attributes not in the TEI P5 guidelines. IDS-XCES is a modified version of XCES, in which some content models of XCES have been redefined and some

new elements and attributes have been introduced according to the needs of the DEREKO corpus descriptions. In doing so, the additional attributes and elements and revised content models have been drawn mostly from the TEI P4 and P5 guidelines so that as a result, IDS-XCES is to a large extent compatible with TEI-P5. Currently, a TEI P5-conformat document grammar for IDS-XCES, dubbed I5, is being customized, which will be applicable to the existing IDS-XCES annotated corpora without the need to convert them [Lüngen/Sperberg-McQueen 2012].

IDS-XCES comprises 167 elements and numerous attributes to encode metadata, in particular all bibliographic metadata, and the features of the corpus and text structure, including hierarchical structure. One major principle of IDS-XCES is the tripartite structuring of an archive into corpus, document, and text. In DEREKO, a corpus for instance comprises all editions of one year of a daily newspaper and contains a number of documents in the form of the daily editions. A document, in turn, contains a number of texts, which in the case of newspapers are the single articles, commentaries and the like. In IDS-XCES, corpora with texts of all text types are structured according to this basic scheme.

IDS-XCES is also designed to encode the basic sentence segmentation. Elements for further linguistic annotations such as part-of-speech tagging, however, are not incorporated in IDS-XCES. Instead, they should be provided as XML standoff annotations pointing into the texts marked up in IDS-XCES. This allows for the provision of any number of concurring linguistic annotations.

IDS-XCES serves as the internal representation format for the corpus access platform COSMAS II [<http://www.ids-mannheim.de/cosmas2/>] at the IDS. This means that texts marked up according to IDS-XCES can be integrated in COSMAS II and thus be queried and analysed via its web interface. Moreover, all annotation features of IDS-XCES can potentially be exploited to define virtual subcorpora, as within DEREKO's primordial sample design [Kupietz et al. 2010]. The DEREKO corpus archive is used by linguistic researchers at the IDS and institutions around the world via COSMAS II. With over 5 billion word tokens, it is the largest archive of contemporary written German. It contains newspaper articles, scientific texts, fiction, and a wide variety of other text types.

Text corpora compliant to either DTABf or IDS-XCES gain the following benefits:

- Texts can be stored in the repositories of the CLARIN-D service centres.
- Texts are searchable via the search engines and the federated content search interface.
- Texts can be made available through corpus analysis software like COSMAS II (IDS) or DDC (BBAW).
- (HTML, TEI, text) views for web presentation can be generated automatically at the CLARIN-D service centres.
- Texts can be converted automatically into TCF (see Section 3.2, “Interoperability and the Text Corpus Format”), the entry point for the CLARIN-D toolchains (see Chapter 8, *Web services: Accessing and using linguistic tools*). These conversion routines will be available by the end of 2012.
- Metadata compliant to the CMDI metadata format can be provided (see Chapter 2, *Metadata*).

A full list of the commonalities and differences between IDS-XCES and DTABf as well as steps towards their unification is in preparation and will be made publicly available after completion.

There are types of corpora which are not covered by the TEI schemas described above, cases where even the TEI guidelines do not suffice, namely texts of computer-mediated communication (CMC). CMC is a genre which falls in between written and spoken communication. Some features of it are not yet expressed appropriately in the TEI guidelines. This concerns the macrostructural level of the “documents” (i.e. threads and logfiles) as well as microstructural elements (e.g. emoticons and addressing terms). In such a case we suggest extensions and modifications to the TEI guidelines. See [Beißwenger et al. 2012] for a detailed account of CMC specific TEI customization.

2.3. Metadata

The recommended metadata format is CMDI. Data in CMDI format can be directly made available through the CLARIN-D infrastructure. For a detailed description of the CMDI framework see Chapter 2, *Metadata*.

Editing tools provided within CLARIN help with the specification of CMDI-formats that are suitable for the necessities of certain projects. However, to reduce the effort of preparing new CMDI formats for individual projects, it is planned to provide services that help with the provision of CMDI metadata for texts encoded according to DTABf or IDS-XCES. Future plans foresee the provision of a web form within CLARIN-D, where metadata can be recorded and subsequently exported in a structured way according to the DTABf and IDS-XCES metadata headers, or CMDI. In addition, conversion routines will be provided, which allow for the conversion of DTABf and IDS-XCES metadata into CMDI.

2.4. Summary and Recommendations

For a suitable integration into the CLARIN-D infrastructure text corpora should be provided with an explicit statement of intellectual property rights and terms and restrictions of usage. This is typically documented through a license. See Chapter 4, *Access to resources and tools – technical and legal issues* for more information on legal aspects in corpus distribution.

The annotation of the text structure should be based on the P5 guidelines of the TEI. The recommended text corpus format is the unified document scheme currently developed within CLARIN-D. Until the finalization of this annotation scheme we recommend to use either DTABf or IDS-XCES. For both original formats lossless converters will be supplied.

The use of any of these three formats assures the interoperability of corpus texts on several levels: full text search, web display of the texts and the possibility to process the texts by the CLARIN-D tool chain. Some examples of corpora already integrated into the CLARIN-D infrastructure [<http://www.deutschestextarchiv.de/dtae>] can be found on the DTA website.

3. Multimodal corpora

Dieter van Uytvanck, MPI for Psycholinguistics Nijmegen



editorial note

This section is currently being rewritten.

3.1. Examples of possible modes within corpora

- Audio: Spontaneous conversation, interview with one or more interviewees, experiment (structured, following a plan), cultural events, celebrations without particular speaker.
- Video: videos corresponding to above listed points, fieldtrip interviews, cultural events and experiments; sign language, gesture analysis: conversation, story telling, structured experiment (performing tasks according to some schedule, answering questions).
- Text: transcription, translation, deeper linguistic annotations (describing morphosyntactic properties, actions in the video, including human interaction, emotions, reactions, comments, etc.).
- MRI data recorder during an experiment.
- Eye-tracking data – direction of the gaze, pupils' size, focused area.
- Hand movement – virtual glove recording the position of the hand and all the joints.
- Motion capture data – part or full body with numerous markers attached.

3.2. Some background on audiovisual data formats

There are no strict standards of audio, video or annotations within the CLARIN-D network. Each center has developed their standards and solutions with respect to the needs of their users. Nevertheless, usability of the resources and tools by any other potential users was always desired. Therefore, popular and open standards are enforced.

When speaking about formats for audiovisual data, one needs to make a distinction between the container format and the actual encoding of the media streams. The container is used as a wrapper around one or more streams of audio and/or video data and typically some metadata about these streams. Many container formats can contain audio and/or video streams in a variety of encodings, although there are also “single coding formats” that only support one type of encoding. Examples of container formats are: WAV or AIFF for audio; AVI, Matroska (MKV) or QuickTime (MOV) for video.

Audio and video streams can be encoded and decoded by means of a “codec”, a piece of software that can read and write audio or video data according to a certain encoding scheme. Often these encoding schemes make use of data compression in order to reduce storage or bandwidth requirements. Data compression can be lossless, i.e. completely reversible such as in the case of a zip file, or lossy, in which case information is thrown away that can no longer be recovered. Most audio and video compression algorithms make use of weaknesses of human perception in order to throw away information that is less noticeable to us.

It depends a lot on the research questions and on the long-term preservation goals whether lossy compression of audiovisual data is an issue or not. A lot of research questions in linguistics for example can be answered perfectly well when using mp3-compressed audio. The question is then whether this data should serve other purposes as well and whether it needs to be preserved for the long term. Certain phonetic analyses, for example, could be affected by the omissions introduced by the mp3 compression algorithm. Converting from one lossy

compressed encoding to another typically introduces artifacts (“data” that has its origin not in the depicted real world event or state, but by data manipulation algorithms, such as the curly paper effect on grey areas in a Xerox copy of a Xerox copy), so after several conversions the quality of the recordings will decrease. Since file formats and encodings typically have a limited life span, one should in principle store data in uncompressed form if interpretability in the long run is a goal. The current situation however is such that this is easily feasible for audio recordings but nearly impossible for video recordings. Video recording equipment that falls within the typical humanities research budgets does not record in uncompressed form and even if it did, the storage requirements would be still too demanding for today’s storage prices.

Compression of video data can be done within each video image (intra-frame compression, comparable to JPEG compression of still images) as well as between consecutive images (inter-frame compression). Since two consecutive images in a video signal typically share a lot of content, some compression algorithms make use of that by only storing the differences rather than each full image. If the data rate is sufficiently high and there is not a lot of motion in the video, the intra-frame compression is hardly noticeable, but with lower data rates and fast motion, some artifacts can occur (blocks appear in the image). The MiniDV tape-based format only used intra-frame compression, but most consumer and semi-professional camcorders today use MPEG2 or MPEG4/H.264 compressed video, which also uses inter-frame compression.

The quality of audio and video recordings depends on many more factors than just the file format and encoding (microphone, lens, or imaging sensor quality, correct settings and operation of the equipment, good recording circumstances, etc.), but these are beyond the scope of this survey. When looking at uncompressed digital audio, there are basically two parameters that determine the maximum quality that can technically be obtained: the bit rate and the sampling frequency. The bit rate determines the maximum possible dynamic range of the recording (difference between the loudest and softest possible sound) and the sampling frequency determines the highest possible frequency component that can be represented in the signal (half the sampling frequency, e.g. 22.05 kHz for a signal recorded at sampling frequency of 44.1 kHz).

When looking at uncompressed digital video, there are four basic parameters that determine the theoretical maximum quality that can be obtained: the spatial image resolution (number of pixels in each image) the temporal resolution (number of images per second), the colour depth (how many different colour values can a single pixel have) and the dynamic range (how many different light intensity values can a single pixel have). In common video hardware we have seen a shift in the spatial resolution of the image from Standard Definition to High Definition, which contains up to 4 times as many pixels. The temporal resolution on typical video recording hardware is limited by the video standards being used in the particular region, e.g. 25 frames per second for the PAL standard and 30 frames per second for the NTSC standard. Some recent hardware can record double the standard number of frames per second, and there are specialised high-speed video cameras that can record thousands of frames per second. Most video recording hardware reduces the colour information that is stored by making use of a technique called “Chroma Subsampling”. Since the human eye has less spatial sensitivity for colour information than for light intensity, the colour information is not stored for every single pixel but for groups of 2 or 4 pixels. Colour information in a digital video signal is typically represented as a 24-bit value (8 bits per primary colour) resulting in over 16 million possible different colours, which is more than can be distinguished by the human eye. The dynamic range of the human eye is much wider than the dynamic range of current video and display technology, even when not considering dynamic adaptations that can occur within the human eye over time (pupil dilation or constriction). This is already a limitation of the imaging sensor technology and therefore not in the first place an issue of the digital representation in current common video formats.

3.3. Recommendations

3.3.1. General recommendations

Data producers should search for information about the limitations of certain audio-visual data representations and select formats that offer sufficient resolution in all dimensions that are relevant for the research questions that they or future users might have. At the same time they should inform themselves about any formats recommended or required by the data repository at which they would like to archive their material.

Data repositories that preserve research data for the long term should select a limited set of audio-visual formats as archival formats, based on the following criteria:

- suitability to represent the recorded events in sufficient detail in all relevant dimensions
- suitability to represent the recorded events in sufficient detail in all relevant dimensions
- suitability to convert to a different format without (too much) loss of information
- long-term perspective of the format
- openness of the format
- general acceptability of the format within the research domain (in case the criteria above are already met and one needs to choose between formats that are in principle suitable as archival formats).

For the time being, in practice the choice for certain video formats will be a compromise of some kind since not all criteria can be fulfilled. Derived working formats can typically be generated for practical usability if the archival format itself is not usable.

3.3.2. Practical recommendations given the current state of technology

Audio recordings: Use uncompressed linear PCM audio at sufficient bit rates and sampling frequencies for the recorded material. For speech this would be 16 bit 22 kHz as a minimum, for field recordings where the environment needs to be accurately represented this would be 16 bit 44.1 kHz, for highly dynamic music this would be 24 bit 44.1 kHz.

Video recordings: For most purposes, video formats produced by the higher end of today's consumer camcorders (MPEG2 or MPEG4/H.264 at high bit rates) are sufficient in quality and can be stored in their original form until storage space is cheap enough to convert the material to a lossless format, if long-term preservation is a requirement.

- standard definition video size (720 × 576, 704 × 480), MPEG-2 compression up to 9.8 Mbit/s (usually around 3.5 Mbit/s) – Fieldtrip recordings, field interviews, cultural events
- high definition video (1280 × 720, 1920 × 1080), H.264/MPEG-4 AVC compression up to 48 Mbit/s (usually around 9 Mbit/s) – detailed analyses of gestures, eye gaze and facial expressions (although this also depends a lot on the framing and distance when filming)

For special collections one could already consider to store the material in a lossless compressed format, such as currently the MJPEG2000 format. Keep in mind, however, that all consumer camcorders today record audio in compressed form.

4. Lexical resources

Axel Herold, BBAW Berlin

In this section you will learn how you can integrate a particular lexical resource into the CLARIN-D infrastructure. You will be guided through some widely accepted standards for various types of lexical resources, for which we will introduce prototypical examples. With Toolbox/MDF we will introduce an example of an environment for the recording of lexical entries.

This section gives only a rather broad overview of the main concepts connected with lexical resources and is not intended to replace general and introductory works, e.g. [Svensen 2009] and, for German, [Kunze/Lemnitzer 2007] and [Engelberg/Lemnitzer 2009].

Since the handling of lexical resources in the CLARIN-D infrastructure is not yet as mature as for example with text corpora, we will give you some recommendations of how to best prepare your resource in order to make integration as smooth as possible.

4.1. Introduction

Lexical resources are collections of *lexical items*, typically together with linguistic information and/or classification of these items. Commonly encountered types of lexical items are words, multi-word units and morphemes. Individual items of a lexical resource need not be necessarily of the same type. They are represented by a *lemma*. A lexical item and the information related to it together form a *lexical entry*. Other common general terms used for lexical resources are

- *dictionary*, mostly referring to reference works compiled to be used by humans directly, and
- *lexicon*, used in a more technical sense mainly as integral part of complex natural language processing (NLP) applications such as language/speech analysis and production systems and thus often not used by humans directly.

Exact definitions of these terms vary slightly across different scientific communities.

Types of lexical resources can be established based on various criteria. The following list is not exhaustive. It rather serves to illustrate the conceptual diversity among lexical resources with regard to their internal structure and the information that they provide:

- The *subject* of a lexical resource determines the information that is recorded in its entries. There are no inherent restrictions as to what information may be supplied. Morphological, semantic, phonological and etymological information is commonly found in lexical resources among numerous other types.
- A lexical resource can be construed as an unordered set or an ordered list of lexical entries but it may also have a more complex internal structure such as a hierarchical tree (see Section 4.2.3, “WordNet and similar resources”). This relation between lexical entries is called the resource's *macro structure*.
- Lexical items are generally considered to have an internal hierarchical structure termed *micro structure*. Thus lexical resources can be classified according to the depth of their micro

structure. The minimal structural depth is found in lemma lists that only serve to identify or enumerate lexical items. There is no inherent restriction to the maximum depth of a lexical item's internal structure.

- *Human vs. machine readable*: This dimension characterizes the intended main addressee of the resource. The content of human readable dictionaries can be readily read by humans in its primary representation (e.g. black ink on white paper or a rendered image on a computer screen). In machine readable dictionaries (*MRD*) the data is accessible for computers by means of a fixed set of encodings together with a scheme supporting the identification of individual units of information. These units need not necessarily be of linguistic or lexicographic relevance, i.e. they do not need to represent the lexical entries' micro structure. A plain text representation of a lexical resource as a stream of individual characters is already a most primitive form of MRD.

Human and machine readability are not complementary features. As long as lexical data is provided in a plain text encoding an electronic lexical resource is both human and machine readable. For a more detailed account of the overlap between human and machine readability see the discussion of *views* in Section 4.2.1, "Text encoding initiative".

- The distinction between *monolingual* and *multilingual* resources focusses on the number of subject languages with regard to the lexical items of the resource. In a monolingual lexicon lexical items systematically occur in only one language while in a multilingual lexicon lexical items in one language are systematically connected with equivalent lexical items in at least one other language.
- Lexical information given in a lexicon can be recorded in different modes of communication (see also Section 3, "Multimodal corpora"). While most lexicons contain only written data some record speech data or gesture representations. At the time of writing there are no lexical resources provided by CLARIN-D that contain data other than written text.

Generally, the micro structure of lexical entries and the relations among the entries are the key features of a lexical resource and consequently have to be accounted for in the data modeling.

4.2. Common formats

This section constitutes a brief overview of the most commonly encountered types and formats of electronic lexical resources and the data formats they are encoded in. The sections on TEI, LMF and wordnets describe widespread general purpose formats; Section 4.2.4, "Toolbox/MDF" illustrates the broad spectrum of existing lexical models by sketching resources built for rather narrow domains.

4.2.1. Text encoding initiative

Many dictionaries in the Text encoding initiative's (TEI) [<http://www.tei-c.org/>] markup format are results of retrodigitization efforts due to the TEI's strong focus on printed text representation. Although it is entirely possible to develop born-digital dictionaries in TEI we are not aware of any such enterprise.

[TEI P5] provides a detailed description of the TEI markup format for electronic text of various kinds (chapter 9 specifically deals with the representation of dictionary data.).

The key to understanding TEI for dictionaries is the distinction between different views that can be encoded for a given dictionary:

1. The *typographic view* is described by [TEI P5] as “the two-dimensional printed page”, i.e. a view on the textual data that allows the reconstruction of the page layout including all artefacts that are due to the print medium like line breaks.
2. In the *editorial view* the text is represented as a “one-dimensional sequence of tokens”, i.e. a token stream without specific media-dependent typographic information.
3. Finally, the *lexical view* represents the structured lexicographical information in the dictionary regardless of its typographic or textual form in print. This is also a way of enriching the dictionary with lexical data that is not present in the primary (printed) text.

A purely lexical representation of a dictionary's textual content can be perceived as a database model of that data.

In a concrete instance these views need not necessarily be separated on a structural level although it is strongly recommended to keep them apart. This way the modeling of conflicting hierarchies across different views can be avoided. The TEI guidelines advise encoders to encode only one view directly using the primary XML element structure and provide information pertaining to another view by other devices like XML attributes or empty elements (e.g. *milestone*).

In the context of linguistic processing within the CLARIN-D workflow, the lexical view (the lexicographically informed structure and classification of the textual content) is of primary interest.

Example 6.1, “TEI encoded lexical entry for *Bahnhof* (train station), lexical view.” shows an example of a TEI dictionary article in the lexical view for the German word *Bahnhof* (“train station”) with a rather modest level of lexicographic details (wordform, grammatical properties, senses, definitions, quotations). The TEI is a highly configurable framework and the level of details for the lexicographical view can be considerably increased, e.g. by means of customizing `@type` or `@value` attributes and the set of permitted values.

Every TEI dictionary file contains an obligatory metadata header (see Section 5.5, “Text Encoding Initiative (TEI)” in Chapter 2, *Metadata*). Within the `text` section a highly developed vocabulary for lexical resources allows for the detailed annotation of a broad range of lexicographic elements.

Example 6.1. TEI encoded lexical entry for *Bahnhof* (train station), lexical view.

```
<entry xml:id="a_1">
  <form>
    <orth>Bahnhof</orth>
  </form>
  <gramGrp>
    <pos value="N" />
    <gen value="masculine" />
  </gramGrp>
  <sense>
    <def>...</def>
    <cit>
      <quote>der Zug fährt in den Bahnhof ein</quote>
    </cit>
  </sense>
</entry>
```

```

    </sense>
    <!-- ... -->
    <sense>
        <def>...</def>
    </sense>
</entry>

```

In Example 6.1, “TEI encoded lexical entry for *Bahnhof* (train station), lexical view.”, the `entry` element is the basic unit of information which contains clearly form based (`form`) and semantic information (`sense`). The `gramGrp` element comprises grammatical information separate from the form description. This is not necessarily the case and may be encoded differently e.g. by subsuming `gramGrp` under `form` and thus relating the grammatical features to the concrete formal realization of the lemma. For existing print dictionaries the explicit relations between (micro)structural elements cannot always be reliably reconstructed or may be inconsistent throughout the resource. To account for this situation the TEI guidelines are very liberal with respect to permitted structures and often allow direct and indirect recursive inclusion of elements. For this reason encoding projects exploiting the guidelines resort to customization, i.e. project specific constrains and extensions for the generically permitted structures and possible values.

To tighten the TEI element semantics beyond the informal descriptions provided by the guidelines explicit references to the ISOcat data category registry (see Chapter 3, *Resource annotations*) can be established via `@dcr:*` attributes in the document instance. See Example 6.2, “Part of a TEI dictionary entry with references to ISOcat” for an example. To reduce the verbosity of directly linking each instance of a data category to the registry directly, the connection should preferably be maintained via `equiv` elements in the *ODD* file (“One document does it all”), a combination of schema fragments and documentation based on the TEI’s `tagdocs` module which is typically used for customizing the TEI schema.

Example 6.2. Part of a TEI dictionary entry with references to ISOcat

```

<gramGrp>
  <pos value="N"
    dcr:datcat="http://www.isocat.org/datcat/DC-1345"
    dcr:valueDatcat="http://www.isocat.org/datcat/DC-1256" />
</gramGrp>

```

4.2.2. Lexical markup format

Unlike the general TEI model, the Lexical markup format [<http://www.lexicalmarkupframework.org/>] (LMF, [ISO 24613:2008]) focusses exclusively on the lexical representation of the data (equivalent to TEI’s lexical view) because the design goal for LMF was the creation of a detailed *meta model* for all types of NLP lexicons, i.e. electronic lexical databases. In LMF, the reference to a data category registry is mandatory. This guarantees semantic unambiguity for the categories used within this framework.

The framework consists of a generic core package accompanied by a set of more domain specific extensions (morphology, morphological patterns, multiword expressions, syntax, semantics, multilingual notations, constraint expression, machine readable dictionaries).

There is a strict division between form related and semantic information on the level of individual entries. Unlike in the TEI guidelines recursion is kept at a minimum. In the core package, only *Sense* allows for recursion. LMF provides a generic feature structure

representation (*feat* class) which enables the modeling of data and annotations for LMF elements.

Example 6.3. Part of an LMF encoded lexicon entry based on the LMF core and morphology packages.

```
<LexicalEntry>
  <feat att="partOfSpeech" val="noun" />
  <feat att="gender" val="masculine" />
  <Lemma>
    <feat att="writtenForm" val="Bahnhof" />
  </Lemma>
  <WordForm>
    <feat att="writtenForm" val="Bahnhof" />
    <feat att="grammaticalNumber" val="singular" />
  </WordForm>
  <WordForm>
    <feat att="writtenForm" val="Bahnhöfe" />
    <feat att="grammaticalNumber" val="plural" />
  </WordForm>
  <Sense id="s1">
  </Sense>
  <!-- ... -->
  <Sense id="sn">
  </Sense>
</LexicalEntry>
```

Compared to the vanilla TEI guidelines the LMF meta-model is much tighter designed due to its much narrower target domain, namely NLP applications. This condensed focus makes it a good choice for resource harmonization and alignment for electronic lexicographic resources in complex research infrastructures such as CLARIN-D. A number of research projects have already developed and exploited XML serializations of LMF mainly for the purpose of data exchange, including RELISH [<http://www.mpi.nl/RELISH>], KYOTO [<http://www.kyoto-project.eu/>] and LIRICS [<http://lirics.loria.fr/>].

4.2.3. WordNet and similar resources

The Princeton WordNet [<http://wordnet.princeton.edu/>] and conceptionally similar databases for other languages than English are probably the most often exploited lexical resources within the NLP community. See e.g. [Fellbaum 1998] for some prototypical applications of wordnets.

In a wordnet, *synsets* (synonym sets) are the building blocks of the resource. Synsets represent mental concepts and can be linguistically expressed by one or more lexical word forms. These word forms are perceived as synonymous in the context of the synset; they share the same meaning that is represented by the synset. It is possible for a synset not to have a linguistic expression instantiating it in any given language, though (*lexical gap*). A linguistic expression on the other hand can appear in more than one synset. In many wordnets additional information is provided for a synset to make it more accessible for humans, most notably semantic paraphrases (*glosses*) or (corpus) quotations.

Among synsets different *conceptual relations* can be established which leads to a net-like conceptional structure (hence the name *wordnet*). If only *hyponymy* (*is_a*) and its reverse relation *hyperonymy* are considered the conceptional structure becomes tree-like and thus represents a (partial) *conceptual hierarchy*. Wordnets differ with respect to the enforcement

of a tree structure for these relations. Another conceptional relation that is often found is *meronymy* (`part_of`), possibly further divided into sub-relations. Different wordnets may maintain different sets of conceptual relations.

While conceptional relations are established among synsets, *lexical relations* can be established among the linguistic expressions. These relations may be relatively broad and underspecified with respect to the grammatical processes involved (`derived_from`) but also very fine grained lexical relation systems can be implemented.

While the Princeton WordNet traditionally uses so called lexicographer files (a proprietary plain text database system spread across a number of files), and alternatively a Prolog version of the database, most other wordnet projects have adopted proprietary XML serializations as their distribution format. The Princeton WordNet project is currently also planning on a transition to an XML serialization.

4.2.4. Toolbox/MDF

The so called “standard format” used by MDF (Multi-Dictionary Formatter, part of SIL's Toolbox [<http://www.sil.org/computing/toolbox/>], a software suite that is popular among field linguists) is a set of field marker and value pairs (i.e. feature-value-pairs) that is linearized into a plain text file. MDF provides an implicit default hierarchy of its field markers that can be redefined by the user. This effectively allows the choice between the creation of form (i.e. part-of-speech) oriented or sense oriented entries (cf. Example 6.4, “MDF encoded sense oriented Iwaidja lexicon entry”).

The MDF standard format is primarily used for the generation of print dictionaries. It can be mapped onto LMF provided form and sense descriptions within entries are cleanly separated (see [Ringersma/Drude/Kemp-Snijders 2010]). There is also a native XML serialization for the standard format available (Lexicon interchange format, LIFT [<http://code.google.com/p/lift-standard/>]).

Example 6.4. MDF encoded sense oriented Iwaidja lexicon entry

```
\lx alabanja
\sn 1
\ps n
\de beach hibiscus.Rope for harpoons and tying up canoes
is made from this tree species, and the timber is used to
make |fv{larrwa} smoking pipes
\ge hibiscus
\re hibiscus, beach
\rfs 205,410; IE 84
\sd plant
\sd material
\rf Iwa05.Feb2
\xv alabanja alhurdu
\xe hibiscus string/rope
\sn 2
\ps n
\de short-finned batfish
\ge short-finned batfish
\re batfish, short-finned
\sc Zabidius novaemaculatus
\sd animal
```

```
\sd fish
\rf Iwaidja Fish Names.xls
\so MELP project elicitation
\eb SH
\dt 19/Dec/2006
```

Note the \sn (sense number) and \ps (part-of-speech) field markers.

The example was taken from the [Ringersma/Drude/Kemp-Snijders 2010] presentation.

4.3. Formats endorsed by CLARIN-D

CLARIN-D directly supports LMF serializations and its own internal pivot format TCF. Due to the high structural diversity among lexical resources in TEI or Toolbox/MDF formats it is not feasible to maintain general purpose conversion tools for transformation into CLARIN-D compatible resources. However, a clearing center for lexical resources is operated by BBAW that provides guidance and support for the creation of project specific conversion tools. You can contact the clearing center via e-mail at <mailto:dwds@dwds.de>.

We encourage users to either contact the clearing center for lexical resources or provide an LMF serialisation of their data if that is already available.

Depending on the research question it may also be appropriate to use a dictionary's text or parts of it as a text corpus, e.g. as a collection of all quoted usage examples. If a lexical resource is intended to be used that way as opposed to a lexical database the recommendations of Section 2, "Text Corpora" apply.



Existing LMF serializations

LMF is intended to enable the mapping of all existing (NLP) lexicons onto a common model and LMF compliant resources are directly supported by CLARIN-D.

CLARIN-D provides and maintains a (partial) mapping from LMF based serializations to its internal pivot format TCF.



Converting proprietary formats to LMF

To transform a proprietary format into an LMF serialisation the following steps have to be taken:

- identify all micro- and macro-structural elements and possible annotations of the resource,
- map these categories to their LMF representation,
- determine the corresponding ISOCat categories (or create them in case they do not already exist), and
- create a schema description for the LMF serialisation with references to the ISOCat categories. For details as to how to create a schema description see Chapter 5, *Quality assurance*.

Following this approach the Princeton WordNet together with the Dutch, Italian, Spanish, Basque and Japanese wordnets were successfully transformed into LMF

serializations in the course of the KYOTO project [<http://www.kyoto-project.eu/>]. Alignment across the wordnets was also modeled in LMF demonstrating the suitability of this framework for representing this subset of lexical resources. Tools and data are available on the project's homepage [http://xmlgroup.iit.cnr.it/kyoto/index.php?option=com_content&view=section&id=28&Itemid=131].



TCF pivot format

Support for lexical resources via TCF – the CLARIN-D internal representation for linguistic data – is still in its infancy. The TCF lexicon module provides means for representing word form based result lists for queries on textual corpora (see Section 2, “Text Corpora”) or lexical resources. It follows the stand-off annotation paradigm and currently implements the layers *lemmas* (mandatory), *POStags*, *frequencies*, and *word-relations* (all optional). See Example 6.5, “TCF representation for lexical items” for an example instance.

Example 6.5. TCF representation for lexical items

```
<Lexicon xmlns="http://www.dspin.de/data/lexicon"
  lang="de">
  <lemmas>
    <lemma ID="l1">halten</lemma>
    <lemma ID="l2">Vortrag</lemma>
  </lemmas>
  <POStags tagset="basic">
    <tag lemID="l1">Verb</tag>
    <tag lemID="l2">Noun</tag>
  </POStags>
  <frequencies>
    <frequency lemID="l1">1257</frequency>
    <frequency lemID="l2">193</frequency>
  </frequencies>
  <word-relations>
    <word-relation type="syntactic relation"
      func="verb+direct-object" freq="13">
      <term lemID="l1"/><term lemID="l2"/>
      <sig measure="MI">13.58</sig>
    </word-relation>
  </word-relations>
</Lexicon>
```

The example was taken from [Przepiórkowski 2011].

The technical description of the TCF lexicon model is available at the CLARIN-D website [<http://www.clarin-d.de/index.php/en/component/content/article/53-tutorials/20>].

We do not recommend the direct provision of TCF based versions of lexical resources as TCF is an ever evolving pivot format meant for data representation solely within the CLARIN-D infrastructure. Currently, it is not a suitable common model for lexical resources in the broad sense intended by LMF. The TCF format is tailor-made with the needs of NLP tool chains in mind and will therefore be subject to changes when additional needs for processing within CLARIN-D arise.

Chapter 7. Linguistic tools

Scott Martens, Kathrin Beck, Thomas Zastrow, Universität Tübingen

Computational linguistic tools are programs that perform operations on linguistic data, i.e. analyses, transformations or other tasks that add to or change language data, or that assist people in performing such tasks. In this section we provide an introduction to the general classes of linguistic tools and what purposes they serve. It is intended to provide computer programmers, technicians, and humanities researchers outside of computational linguistics with a background for understanding linguistic tools in order to better use the CLARIN-D infrastructure and identify how it can meet their needs. This means discussing some notions from linguistics that are specifically relevant to understanding language processing tools.

The CLARIN-D project is focused on providing an infrastructure for the maintenance and processing of language resources, in which natural language processing (NLP) and computational linguistics (CL) play prominent roles. CLARIN-D supports many different tools of very different kinds, without regard for the scientific and theoretical claims behind those tools. Individual linguistic tools and resources may be based on specific linguistic schools or theoretical claims, but the CLARIN-D infrastructure is neutral with respect to those theories.

Many computational linguistic tools are extensions of pre-computer techniques used to analyze language. Tokenization, part-of-speech tagging, parsing and word sense disambiguation, as well as many others, all have roots in the pre-computer world, some going back thousands of years. Computational tools automate these long-standing analytical techniques, often imperfectly but still productively.

Other tools, in contrast, are exclusively motivated by the requirements of computer processing of language. Sentence-splitters, bilingual corpus aligners, and named entity recognition, among others, are things that only make sense in the context of computers and have little immediate connection to general linguistics but may be very important for computer applications.

Linguistic tools can also encompass programs designed to enhance and facilitate access to digital language data. Some of these are extensions of pre-computer techniques like indexing and concordancing (i.e. preparing a sorted list of words or other elements in a text, along with their immediate context, or all of the locations in the text where they occur, or both). But linguistic tools can also include more recent developments like search engines, some of which are already sensitive to the linguistic annotation of primary data. Textual information retrieval is a large field and in this user guide we will discuss only search and retrieval tools specialized for or based on linguistic analysis.

1. Hierarchies of linguistic tools

Linguistic tools are often interdependent and frequently incorporate some elements of linguistic theory. Modern linguistics draws on traditions of *structuralism*, a school of thought in the humanities and social sciences dating to the early 20th century. Structuralism emphasized the study of phenomena as hierarchal systems of elements, organized into different levels of analysis, each with their own units, rules, and methodologies. In general, linguists organize their theories in ways that show structuralist influences, although many disclaim any attachment to structuralism. Linguists disagree about what levels of analysis exist, what units are appropriate to each level, to what degree different languages might have differently organized systems of

units and levels, and how those levels interrelate. However, hierarchal systems of units and levels of analysis are a part of almost all linguistic theories and are very clearly reflected in the practices of computational linguists.

Linguistic tools are usually categorized by the level of analysis they perform, and different tools may operate at different levels and over different units. There are often hierarchal interdependencies between tools. A tool used to perform analysis at one level often requires, as input, the results of an analysis at a lower level.

Figure 7.1. A hierarchy of levels of linguistic analysis

	Discipline	Units and categories	Tools
“Higher” levels of analysis ↑	Pragmatics	Discourse type	Emotion analyzers
	Discourse theory	Genre	Rhetorical coherency tools
	Rhetoric	Speech act category	Named entity recognizers
“Lower” levels of analysis ↓	Speech act theory	Emotion	
	Semantics	Predicate Logical representation Word sense	Word sense disambiguators Semantic role analyzers
	Syntax	Sentence Phrase Word	Syntactic Parsers Chunkers
	Morphology and lexical analysis	Word Prefix and suffix Grammatical gender Grammatical number Conjugation Declension	Tokenizers Stemmers Lemmatizers Part-of-speech taggers
	Phonetics and phonology	Sound Phoneme Syllable Intonational category	Speech recognition tools Spectrograms/Sonograms Speech segmentation tools

Figure 7.1, “A hierarchy of levels of linguistic analysis” is a very simplified hierarchy of linguistic units, sub-disciplines and tools – which is why “higher” and “lower” are in quotes. It does not provide a complete picture of linguistics and it is not necessarily representative of any specific linguistic school, nor should any implication about the complexity or importance of particular levels of analysis be taken from this scheme. However, it provides an outline and a reference framework for understanding the way hierarchal dependencies between levels of analysis affect linguistic theories and tools: Higher levels of analysis generally depend on lower ones.

Syntactic analysis like parsing usually requires words to be clearly delineated and part-of-speech tagging or morphological analysis to be performed first. This means, in practice, that texts must be tokenized, their sentences clearly separated from each other, and their morphological properties analyzed before parsing can begin. In the same way, semantic analysis is often dependent on identifying the syntactic relationships between words and other elements, and inputs to semantic analysis tools are often the outputs of parsers.

However, this simplistic picture has many important exceptions. Lower level phenomena often have dependencies on higher level ones. Correctly identifying the part-of-speech, lemmas,

and morphological categories of words may depend on a syntactic analysis. Morphological and syntactic analysis can affect phonetic analysis: Without information from higher in the hierarchy, it can be impossible to tell the difference between “I recognize speech” and “I wreck a nice beach” [Lieberman et al. 2005]. Even speech recognition – one of the lowest level tasks – depends strongly on knowledge of the semantic and pragmatic context of speech.

Furthermore, there is no level of analysis for which all linguists agree on a single standard set of units of analysis or annotation scheme. For example, the Penn Treebank [<http://www.cis.upenn.edu/~treebank/>] and the British National Corpus [<http://www.natcorp.ox.ac.uk/>] use different part-of-speech tags. Different tools will have radically different inputs and outputs depending on the theoretical traditions and commitments of their developers.

Most tools are also language specific. There are few functional generalizations between languages that can be used to develop single tools that apply to multiple languages. Different countries with different languages often have very different indigenous traditions of linguistic analysis, and different linguistic theories are popular in different places, so it is not possible to assume that tools doing the same task for different languages will necessarily be very similar in inputs or outputs.

Corpus and computational linguists most often work with written texts, partly because written texts do not usually require phonetic analysis and are easy to find in large quantities, and partly because computational tools are much easier to obtain for written language than speech or other forms of communication. This chapter will not discuss speech recognition and phonetic analysis tools suitable for dealing directly with speech because very few of them are currently part of the CLARIN-D infrastructure, although some of the multimedia tools described here are suitable for some kinds of phonetic analysis. Most tools assume that their inputs are written language data in specified data storage formats.

Furthermore, at the highest levels of analysis, tools are very specialized and standardization is rare, so few tools for very high linguistic levels are discussed here. CLARIN-D is, however, committed to support for all varieties of linguistic tools, and expects to provide more resources at all levels of analysis as the project develops.

2. Automatic and manual analysis tools

Some computational linguistic tools exist just to provide linguists with interfaces to stored language data. The earliest digital corpus tools were made to search in texts and display the results in a compact way, like the widespread KWIC (*key word in context*) tools that date back to the 1970s.

However, many current linguistic tools are used to produce an *annotated resource*. Annotation involves the addition of detailed information to a linguistic resource about its contents. For example, a corpus in which each word is accompanied by a part-of-speech tag, or a phonetic transcription, or in which all named entities are clearly marked, is an annotated corpus. Linguistic data in which the syntactic relationships between words are marked is usually called a *treebank*. The addition of annotations can make texts more accessible and usable for linguistic research, and may be required for further processing. Corpus annotation is discussed in greater depth in Chapter 3, *Resource annotations*.

Automatic annotation tools add detailed information to language data on the basis of procedures written into the software, without human intervention other than to run the program. Automatic

annotation is sometimes performed by following rules set out by programmers and linguists, but most often, annotation programs are at least partly based on machine learning algorithms that are trained using manually annotated examples.

The vast majority of automatic annotation tools for linguistic data today are based on statistical machine learning principles of some kind. This approach to problem-solving in computer science goes back to the early 1950s when they were first applied to computer programs for playing board games. Arthur Samuel's work at IBM on programs to play checkers is usually credited as the first non-trivial work in machine learning. Although there were some very promising early results, there was very little progress in machine learning from the mid 1960s to the early 1990s – the period called the “AI Winter”. There was also very little use of statistical methods in linguistics or natural language processing during that period, for reasons that linguists still debate. However, a series of theoretical breakthroughs followed by very strong practical demonstrations of the value of statistical analysis in linguistics turned the tide in the 1990s, and now automatic linguistic annotation is overwhelmingly based on statistical machine learning techniques. The history of natural language processing is discussed in part in [Jurafsky/Martin 2009], the standard textbook for NLP, including the explosive growth of machine learning and statistical and empirical methods in this field since the early 1990s. The history of natural language processing overlaps heavily with the histories of linguistics, artificial intelligence and computers, all of which are fast changing fields. Although there is a broad but brief treatment of the history of artificial intelligence in the first chapter of [Russel/Norvig 2009] that covers much of what is described in this section, no up-to-date, general history of the field is available.

Machine learning is a complex topic, but from a practical standpoint, many linguistic annotation applications *learn* to annotate texts from manually annotated linguistic data. This process of machine learning is usually called *training*. It means that a linguist prepares a manually annotated corpus and then a computer program processes this data and learns how to replicate automatically the linguist's manual annotation on new corpora. These kinds of programs can often be adapted to new languages and annotation schemes if they are provided with appropriate annotated data to learn from.

Linguistic tools that use machine learning generally save information about the tasks they are trained to perform in files that are separate from the program itself. Using these tools may require specifying the location of that “learned” information.

Automated annotation processes – whether based on machine learning or rules – almost always have some rate of error, and before using any automatic annotation tool, it is important to consider its error rate and how those errors will affect whatever further purpose annotated corpora are used for, see also Chapter 5, *Quality assurance* .

Where possible, researchers prefer manually annotated resources with fewer errors. Fewer errors does not mean *no errors*. Although we often treat manually annotated data as an objectively correct standard, we do so because, first, we expect human errors to be random and unbiased when compared to the systematic errors made by imperfect software; and second, because we have no other means of constructing “correct” annotations than to have trained people make them. Highly reliable manually annotated resources are, naturally, more expensive to construct, rarer and smaller in size than automatically annotated data, but they are essential for the development of automated annotation tools and are necessary whenever the desired annotation procedure either has not yet been automated or cannot be automated. Various tools exist to make it easier for researchers to annotate language data, some of which are described in Section 5.1, “Manual annotation tools” and Section 6, “Multimedia tools”.

3. Technical issues in linguistic tool management

Many of the most vexing technical issues in using linguistic tools are common problems in computer application development. Data can be stored and transmitted in an array of incompatible formats, generated by different applications, and either there are no standard formats, or too many de-facto standards, or standards compliance is poor. Linguistic tool designers are rarely concerned with those kinds of issues or specialized in resolving them. Part of CLARIN-D's mandate, in developing infrastructure for language research, is overcoming these technical problems.

Many tools accept and produce only *plain text* data, sometimes with specified delimiters and internal structures accessible to ordinary text editors. Some tools require each word to be on a separate line, others require each sentence on a line. Some will use comma- or tab-delimited text files to encode annotation in their output, or require them as input. These encodings are often historically rooted in the data storage formats of early language corpora. A growing number of tools use XML for either input or output format.

Character encoding formats are an important technical issue when using linguistic tools. Most languages use some characters other than 7-bit ASCII (the standard letters and punctuation used in English) and there are different standards for characters in different languages and operating systems. Unicode and its most widespread implementation UTF-8 are increasingly used for written language data because they encode nearly all characters from nearly all modern languages. But not all tools support Unicode. Full Unicode compatibility has only recently become available on most operating systems and programming frameworks, and many non-Unicode linguistic tools are still in use. Furthermore, the Unicode standard supports so many different characters that simple assumptions about texts – like what characters constitute punctuation and spaces between words – may differ between Unicode texts in ways that are incompatible with some tools. For example, there are currently more than 20 different whitespace characters in Unicode, five dashes and many more characters that look much like dashes, as well as multiple variants for many common kinds of punctuation. In addition, many alphabetic letters appear in a number of variants in Unicode, some ligatures can be encoded as individual Unicode characters, and many other variations exist that are correctly human readable but anomalous for computer processing.

One of the major axes of difference between various annotation formats is *inline* or *stand-off* annotation. Inline annotation mixes the data and annotation in a single file or data structure. Stand-off annotation means storing annotations separately, either in a different file, or in some other way apart from language data, with a reference scheme to connect the two. See Chapter 3, *Resource annotations* for more information about annotation formats.

4. Automatic segmentation and annotation tools

The tools described in this section operate without very much direct user interaction, producing an annotated or segmented resource as its output. Many of them require training and may be available already trained for some tasks, or be available for users to train to suit their needs.

Those tools which are presently available through WebLicht – CLARIN-D's web service-based linguistic workflow and tool execution environment – are marked with a small icon:



The tools listed here are not an exhaustive list of WebLicht-accessible tools, and as the service grows, more tools will be integrated. For a current and comprehensive list of tools available through WebLicht, please log in to the WebLicht website [<https://weblicht.sfs.uni-tuebingen.de/>].

Sentence splitting and tokenization are usually understood as a way of segmenting texts rather than transforming them or adding feature information. Each segment, be it a sentence or a token, corresponds to a particular sequence of lower level elements (tokens or characters) that forms, for the purposes of further research or processing, a single unit. Segmenting digital texts can be complicated, depending on the language of the text and the linguistic considerations that go into processing it.

4.1. Sentence splitters

Sentence splitters, sometimes called *sentence segmenters*, split text up into individual sentences with unambiguous delimiters.

Recognizing sentence boundaries in texts sounds very easy, but it can be a complex problem in practice. Sentences are not clearly defined in general linguistics, and sentence-splitting programs are driven by the punctuation of texts and the practical concerns of computational linguistics, not by linguistic theory.

Punctuation dates back a very long time, at least to the 9th century BCE. The Meša Stele – an inscribed stone found in modern Jordan describing the military campaigns of the Moabite king Meša – is the oldest attestation of different punctuation marks to indicate word separation and grammatical phrases [Compston1919], [Martens 2011]. Until modern times though, not all written languages used punctuation. The idea of dividing written texts into individual sentences using some form of punctuation is an invention of 16th century Italian printers and did not reach some parts of the world until the mid-20th century. This makes it very difficult to develop historical language corpora compatible with tools based on modern punctuation. Adding punctuation to old texts is time-consuming and researchers of different schools will not always agree on where the punctuation belongs.

In many languages – including most European languages – sentence delimiting punctuation has multiple functions other than just marking sentences. The period often marks abbreviations as well as being used to write ordinal numbers or to split large numerical expressions in groups of three digits. Sentences can also end with a wide variety of punctuation other than the period. Question marks, exclamation marks, ellipses (*dropped words*), colons, semi-colons and a variety of other markers must have their purpose in specific contexts correctly identified before they can be confidently considered sentence delimiters. Additional problems arise with quotes, URLs and proper nouns that incorporate non-standard punctuation. Furthermore, most texts contain errors and inconsistencies of punctuation that simple algorithms cannot easily identify or correct.

Sentence splitters are often integrated into tokenizers, but some separate tools are available including:

MX Terminator [<ftp://ftp.cis.upenn.edu/pub/adwait/jmx/>]

A splitter for English that can be trained for other languages.

Stanford ssplit [<http://nlp.stanford.edu/software/corenlp.shtml>]

A splitter for English, but quite effective in other languages that use similar punctuation.

OpenNLP Sentence Detection [<http://opennlp.apache.org/>] 

A splitter for English that can be trained for other languages.

4.2. Tokenizers

A token is a unit of language similar to a word but not quite the same. In computational linguistics it is often more practical to discuss tokens instead of words, since a token encompasses many linguistically irrelevant and / or defective elements found in actual texts (numbers, abbreviations, punctuation, etc.) and avoids many of the complex theoretical considerations involved in talking about words.

In modern times, most languages have writing systems derived from ancient languages used in the Middle East and by traders on the Mediterranean Sea starting about 3000 years ago. The Latin, Greek, Cyrillic, Hebrew and Arabic alphabets are all derived from a common ancient source – a variety of Phoenician widely used in trade and diplomacy – and most historians think that the writing systems of India and Central Asia come from the same origin. See [Fischer 2005] and [Schmandt-Besserat 1992] for fuller histories of writing.

All of the writing systems derived from ancient Phoenician use letters that correspond to specific sounds. When words are written with letters that represent sounds, words can only be distinguished from each other if set apart in some way, or if readers slowly sound the letters out to figure out where the pauses and breaks are. The first languages to systematically use letters to represent sounds usually separated text into word-like units with a mark of some kind – generally a bar (“|”) or a double-dot mark much like a colon (“:”). However, these marks were used inconsistently and many languages with alphabets stopped using explicit markers over time. Latin, Greek, Hebrew and the languages of India were not written with any consistent word marker for many centuries. Whitespaces between words were introduced in western Europe in the 12th century, probably invented by monks in Britain or Ireland, and spread slowly to other countries and languages [Saenger 1997]. Since the late 19th century, most languages – all but a few in Pacific Asia – have been written with regular spaces between words.

For those languages, much but far from all of the work of tokenizing digital texts is performed by whitespace characters and punctuation. The simplest tokenizers just split the text up by looking for whitespace, and then separate punctuation from the ends and beginnings of words. But the way those spaces are used is not the same in all languages, and relying exclusively on spaces to identify tokens does not, in practice, work very well. Tokenization can be very complicated because *tokens do not always match the locations of spaces*.

Compound words exist in many languages and often require more complex processing. The German word *Telekommunikationsvorratsdatenspeicherung* (“telecommunications data retention”) is one case, but English has linguistically similar compounds like *low-budget* and *first-class*. Tokenizers – or sometimes similar tools that may be called *decompounders* or *compound splitters* – are often expected to split such compounds up, or are expected to only split some of them up. Whether or not compounds should be split may depend on further stages of processing. For syntactic parsing of German, for example, it is often undesirable because compounds are treated syntactically and morphologically like a single word, i.e., plurals and declinations only change the end of the whole compound and parsers can recognize parts-of-speech from the final part of the compound. In French, the opposite is often true and compounds like *arc-en-ciel* (“rainbow”) and cannot be treated as single words because pluralization modifies the middle of the compound (*arcs-en-ciel*). For

information retrieval, in contrast, German words are almost always compounded, because a search for *Vorratsdatenspeicherung* (“data retention”) should match documents containing *Telekommunikationsvorratsdatenspeicherung*. In French, however, no one would want *arc* (“arch”, “arc”, or “bow”) or *ciel* (“sky” or “heaven”) to match *arc-en-ciel*. English has examples of both kinds of compounds, with words like *houseboats* that behave like German compounds, but also *parts-of-speech*, which behave like French ones.

In other cases, something best treated as a single token may appear in text as multiple words with spaces, like *New York*. There are also ambiguous compounds, where they may sometimes appear as separate words and sometimes not. *Egg beater*, *egg-beater* and *eggbeater* are all possible in English and mean the same thing.

Short phrases that are composed of multiple words separated by spaces may also sometimes be best analyzed as a single word, like the phrase *by and large* or *pain in the neck* in English. These are called *multi-word terms* and may overlap with what people usually call *idioms*.

Contractions like *I'm* and *don't* also pose problems, since many higher level analytical tools like part-of-speech taggers and parsers may require them to be broken up, and many linguistic theories treat them as more than one word for grammatical purposes.

Phrasal verbs in English and separable verbs in German are another category of problem for tokenizers, since these are often best treated as single words, but are separated into parts that may not appear next to each other in texts. For example:

1. When we love others, we naturally want to talk about them, we want to *show* them *off*, like emotional trophies. (Alexander McCall Smith, *Friends, Lovers, Chocolate*)
2. Die Liebe im Menschen *spricht* das rechte Wort *aus*. (“People's love utters the right word.”, Ferdinand Ebner, *Schriften*, vol. 2.)

Verbs like *to show off* in English and *aussprechen* in German often require treatment as single words, but in the examples above, appear not only as separate words, but with other words between their parts. Simple programs that just look for certain kinds of characters cannot identify these structures as tokens.

Which compounds, if any, should be split, and which multi-part entities should be processed as a single token, depends on the language of the text and the purpose of processing. Consistent tokenization is generally related to identifying lexical entities that can be looked up in some lexical resource and this can require very complex processing for ordinary texts. Its purpose is to simplify and remove irregularities from the data for the benefit of further processing. Since the identification of basic units in text must precede almost all kinds of further processing, tokenization is the first or nearly the first thing done for any linguistic processing task.

Additional problems can arise in some languages. Of major modern languages, only Chinese, Japanese and Korean currently use writing systems not thought to be derived from a common Middle Eastern ancestor, and they do not systematically mark words in ordinary texts with spaces or any other visible marker. Several southeast Asian languages – Thai, Lao, Khmer and some other less widely spoken languages – still use no spaces today or use them rarely and inconsistently despite having writing systems derived from those used in India. Vietnamese – which is written with a version of the Latin alphabet today, but used to be written like Chinese – places spaces between every syllable, so that even though it uses spaces, they are of little value in tokenization. Tokenization in these languages is a very complex process that can involve large dictionaries and sophisticated machine learning procedures.

As mentioned in the previous section, tokenization is often combined with sentence-splitting in a single tool. Some examples for implementations of tokenizers are:

OpenNLP tokenizer [<http://opennlp.apache.org/>]



A tokenizer for English and German that includes an optional sentence splitter (see Section 4.1, “Sentence splitters”).

Stuttgart tokenizer [<http://www.ims.uni-stuttgart.de/>]



A tokenizer for German, English, French, Italian, Czech, Slovenian, and Hungarian that includes a sentence splitter (see Section 4.1, “Sentence splitters”).

Stanford tokenizer [<http://nlp.stanford.edu/software/corenlp.shtml>]



A tokenizer for English text (part of the Stanford CoreNLP tool).

4.3. Part-of-speech taggers

Part-of-speech taggers (PoS taggers) are programs that take tokenized text as input and associate a *part-of-speech tag* (PoS tag) with each token. A PoS tagger uses a specific, closed set of parts-of-speech – usually called a *tagset* in computational linguistics. Different taggers for different languages will routinely have different, sometimes radically different, tagsets or part-of-speech systems. For some languages, however, de-facto standards exist in the sense that most part-of-speech taggers use the same tagset. In German, for example, the STTS tagset is very widespread, but in English several different tagsets are in regular use, like the Penn Treebank tagset and several versions of the CLAWS tagset.

A part-of-speech is a category that abstracts some of the properties of words or tokens. For example, in the sentence *The dog ate dinner* there are other words we can substitute for *dog* and still have a correct sentence, words like *cat* or *man*. Those words have some common properties and belong to a common category of words. PoS schemes are designed to capture those kinds of similarities. Words with the same PoS are in some sense similar in their use, meaning, or function.

Parts-of-speech have been independently invented at least three times in the distant past. They are documented to the 5th century BC in Greece, approximately for the same period in India, and from the 2nd century AD in China. There is no evidence to suggest any of these three inventions was copied from other cultures. The origins of parts-of-speech are described in greater detail in [Martens 2011]. The 2nd century BCE Greek grammar text *The Art of Grammar* outlined a system of nine PoS categories that became very influential in European languages: nouns, verbs, participles, articles, pronouns, prepositions, adverbs, and conjunctions, with proper nouns as a subcategory of nouns. Most PoS systems in use today have been influenced by that scheme.

Modern linguists no longer think of parts-of-speech as a fixed, short list of categories that is the same for all languages. They do not agree about whether or not any of those categories are universal, or about which categories apply to which specific words, contexts and languages. Different linguistic theories, different languages, and different approaches to annotation use different PoS schemes.

Tagsets also differ in the level of detail they provide. A modern corpus PoS scheme, like the CLAWS tagset used for the British National Corpus, can go far beyond the classical nine parts-of-speech and make dozens of fine distinctions. CLAWS version 7 [<http://www.natcorp.ox.ac.uk/docs/c7spec.html>] has 22 different parts-of-speech for nouns alone. Complex tagsets are usually organized hierarchically, to reflect commonalities between different classes of words.

Examples of widely used tagsets include STTS for German [Schiller et al. 1999], the Penn Treebank Tagset for English [Santorini 1990], and the CLAWS tagset for English [Garside et al. 1997]. Most PoS tagsets were devised for specific corpora, and are often inspired by older corpora and PoS schemes. PoS taggers today are almost all tools that use machine learning and have been specifically trained for the language and tagset they use. They can usually be retrained for new tagsets and languages.

PoS taggers almost always expect tokenized texts as input, and it is important that the tokens in texts match the ones the PoS tagger was trained to recognize. As a result, it is important to make sure that the tokenizer used to preprocess texts matches the one used to create the training data for the PoS tagger.

Another important factor in the development of PoS taggers is their handling of *out-of-vocabulary words*. A significant number of tokens in any large text will not be recognized by the tagger, no matter how large a dictionary they have or how much training data was used. PoS taggers may simply output a special “unknown” tag, or may guess what the right PoS should be given the remaining context. For some languages, especially those with complex systems of prefixes and suffixes for words, PoS taggers may use morphological analyses to try to find the right tag.

Implementations of PoS taggers include:

TreeTagger [<http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/>]



A PoS tagger for German, English, French, Italian, Dutch, Spanish, Bulgarian, Russian, Greek, Portuguese, Chinese, Swahili, Latin, Estonian and old French, and trainable for many others.

OpenNLP tagger [<http://opennlp.apache.org/>]



A PoS tagger for English and German distributed as part of the Apache OpenNLP toolkit.

Stanford PoS tagger [<http://nlp.stanford.edu/software/corenlp.shtml>]



A PoS tagger for English distributed as part of the Stanford Core NLP toolkit.

Brill tagger [<http://cst.dk/download/uk/index.html>]

A PoS tagger for English, Danish, Dutch and Norwegian (Nynorsk).

4.4. Morphological analyzers and lemmatizers

Morphology is the study of how words and phrases change in form depending on their meaning, function and context. Morphological tools sometimes overlap in their functions with PoS taggers and tokenizers.

Because linguists do not always agree on what is and is not a word, different linguists may disagree on what phenomena are part of morphology, and which ones are part of syntax, phonetics, or other parts of linguistics. Generally, morphological phenomena are considered either inflectional or derivational depending on their role in a language.

4.4.1. Inflectional morphology

Inflectional morphology is the way words are required to change by the rules of a language depending on their syntactic role or meaning. In most European languages, many words have to change form depending on the way they are used.

How verbs change in form is traditionally called *conjugation*. In English, present tense verbs have to be conjugated depending on their subjects. So we say *I play*, *you play*, *we play* and *they play* but *he plays* and *she plays*. This is called agreement, and we say that in English, present tense verbs must agree with their subjects, because the properties of whatever the subject of the verb is determine the form the verb takes. But in the past tense, English verbs have to take a special form to indicate that they refer to the past, but do not have to agree. We say *I played* and *he played*.

Languages can have very complex schemes that determine the forms of verbs, reflecting very fine distinctions of meaning and requiring agreement with many different features of their subjects, objects, modifiers or any other part of the context in which they appear. These distinctions are sometimes expressed by using additional words (usually called *auxiliaries* or sometimes *helper words*), and sometimes by prefixes, suffixes, or other changes to words. Many languages also combine schemes to produce potentially unlimited variations in the forms of verbs.

Nouns in English change form to reflect their number: *one dog* but *several dogs*. A few words in English also change form based on the gender of the people they refer to, like *actor* and *actress*. These are rare in English but common in most other European languages. In most European languages, all nouns have an inherent grammatical gender and any other word referring to them may be required to change form to reflect that gender.

In many languages, nouns also undergo much more complex changes to reflect their grammatical function. This is traditionally called *declension* or a *case*. In the German sentence *Das Auto des Lehrers ist grün* (“The teacher’s car is green”), the word *Lehrer* (“teacher”) is changed to *Lehrers* because it is being used to say whose car is meant.

Agreement is often present between nouns and words that are connected to them grammatically. In German, nouns undergo few changes in form when declined, but articles and adjectives used with them do. Articles and adjectives in languages with grammatical gender categories usually must also change form to reflect the gender of the nouns they refer to. Pronouns, in most European languages, also must agree with the linguistic properties of the things they refer to as well as being declined or transformed by their context in other ways.

The comparative and superlative forms of adjectives – *safe*, *safer*, *safest* is one example – are also usually thought of as inflectional morphology.

Some languages have very complex inflectional morphologies. Among European languages, Finnish and Hungarian are known to be particularly complex, with many forms for each verb and noun, and French is known for its complex rules of agreement. Others are very simple. English nouns only vary between singular and plural, and even pronouns and irregular verbs like *to be* and *to have* usually have no more than a handful of specific forms. Some languages (mostly not spoken in Europe) are not thought of as having any inflectional morphological variation at all.

Just because a word has been inflected does not mean it is a different word. In English, *dog* and *dogs* are not different words just because of the *-s* added to indicate the plural. For each surface form, there is a base word that it refers to, independently of its morphology. This underlying abstraction is called its *lemma*, from a word the ancient Greeks used to indicate the “substance” of a word. Sometimes, it is called the *canonical form* of a word, and indicates the spelling you would find in a dictionary (see Section 4, “Lexical resources”).

One source of ambiguity that lemmatizers and morphological analyzers may be expected to resolve is that a particular token may be an inflected form of more than one lemma. In English,

for example, the word *tear* can refer to the noun meaning “a secretion of water from someone’s eyes”, or to the verb that means “to pull something apart”, among other possible meanings. Seen in isolation, *tear* and *tears* could refer to either, but *tearing* usually can only refer to a verb. Most – practically all – languages have similar cases, i.e. German *die Schale* meaning either *the bowl* (singular) or *the scarves* (plural). Disambiguation can involve many sorts of information other than just the forms of the words, and may use complex statistical information and machine learning. Many annotated corpora systematically disambiguate lemmas as part of their mark-up.

Inflectional morphology in European languages most often means changing the endings of words, either by adding to them or by modifying them in some way, but it can involve changing any part of a word. In English, some verbs are inflected by changing one or more of their vowels, like *break*, *broke*, and *broken*. In German many common verbs – called the *strong verbs* – are inflected by changing the middle of the word. In Arabic and Hebrew, all nouns and verbs and many other words are inflected by inserting, deleting, and changing the vowels in the middle of the word. In the Bantu languages of Africa, words are inflected by adding prefixes and changing the beginnings of words. Other languages indicate inflection by inserting whole syllables in the middle of words, repeating parts of words, or almost any other imaginable variation. Many languages use more than one way of doing inflection.

4.4.2. Derivational morphology

Derivational morphology is the process of making a word from another word, usually changing its form while also changing its meaning or grammatical function in some way. This can mean adding prefixes or suffixes to words, like the way English constructs the noun *happiness* and the adverb *unhappily* from the adjective *happy*. These kinds of processes are often used to change the part-of-speech or syntactic functions of words – making a verb out of a noun, or an adjective out of an adverb, etc. But sometimes they are used only to change the meaning of a word, like adding the prefix *un-* in both English and German, which may negate or invert the meaning of a word in some way but does not change its grammatical properties or part-of-speech.

As with inflectional morphology, languages may use almost any kind of variation to derive new words from old ones, not just prefixes and suffixes. There is no simple, fixed line to separate derivational morphology from inflectional morphology, and individual linguists will sometimes disagree about how to categorize individual word formation processes. A few will disagree about whether there is any meaningful distinction between inflection and derivation at all.

One common derivational process found in many different languages is to make compound words. German famously creates very long words this way, and English has many compound constructs that are sometimes written as one word, or with a hyphen, or as separate words that people understand to have a single common meaning. The opposite process – splitting a single word into multiple parts – also exists in some languages. Phrasal verbs in English and separable verbs in German are two examples (see Section 4.2, “Tokenizers”), and a morphological analyzer may have to identify those constructions and handle them appropriately.

4.4.3. Stemmers

One of the oldest and simplest tools for computational morphological analysis is the *stemmer*. The term *stem* refers to the part of a word that is left over when inflectional and derivational prefixes and suffixes have been stripped from a word, and the parts of words left when a compound word has been split into its parts. Many words share a common “stem” like the German words *sehen*, *absehen*, *absehbar* and *Sehhilfe*, which all share the stem *seh*. This stem usually reflects a common meaning.


The ability to reduce groups of similar words to a common form corresponding to a common meaning made stemmers attractive for information retrieval applications. Stemmers are algorithmically very simple and typically use a catalog of regular expressions – simple patterns of letters – for identifying and stripping inflectional and derivative elements. They are not very linguistically sophisticated, and miss many kinds of morphological variation. Whenever possible, more sophisticated tools should be used.

However, stemmers are relatively easy to make for new languages and are often used when better tools are unavailable. The Porter stemmer is the classical easy-to-implement algorithm for stemming [Porter 1980].

4.4.4. Lemmatizers

Lemmatizers are programs that take tokenized texts as input and return a set of lemmas or canonical forms. They usually use a combination of rules for decomposing words and a dictionary, sometimes along with statistical rules and machine learning to disambiguate homonyms. Some lemmatizers also preserve some word class information, like noting that a word ends in an *-s* that was removed, or an *-ing* – usually just enough to reconstruct the original token, but not as much as a full morphological analysis.

Implementations of lemmatizers:

SMOR [<http://wiki.ims.uni-stuttgart.de/extern/StatNLPResources>] 
SMOR is a stand-alone lemmatizer for German that also includes an optional morphological analysis module (see Section 4.4.5, “Morphological analyzers”).

RACAI lemmatizer [<http://www.racai.ro/>]
The RACAI lemmatizer works for Romanian, English and French texts.

MorphAdorner [<http://morphadorner.northwestern.edu/>]
MorphAdorner is a lemmatizer for English written in the Java programming language.


4.4.5. Morphological analyzers


Full morphological analyzers take tokenized text as input and return complete information about the inflectional categories each token belongs to, as well as their lemma. They are often combined with PoS taggers and sometimes with syntactic parsers, because a full analysis of the morphological category of a word usually touches on syntax and almost always involves categorizing the word by its part-of-speech.

Some analyzers may also provide information about derivational morphology and break up compounds into constituent parts.

High-quality morphological analyzers almost always use large databases of words and rules of composition and decomposition. Many also employ machine learning techniques and have been trained on manually analyzed data. Developing comprehensive morphological analyzers is very challenging, especially if derivational phenomena are to be analyzed.

Implementations of morphological analyzers:

RFTagger [<http://www.ims.uni-stuttgart.de/projekte/corplex/RFTagger/>] 
RFTagger assigns fine-grained part-of-speech tags based on a morphological analysis. It works for German, Czech, Slovene, and Hungarian data, and can be trained for other languages.

SMOR [<http://wiki.ims.uni-stuttgart.de/extern/StatNLPResources>] 
Contains a morphological analyzer for German.

Morpha [<http://www.informatics.sussex.ac.uk/research/groups/nlp/carroll/morph.html>]
A morphological analyzer for English.

4.5. Syntax

Syntax is the study of the connections between parts of sentences. It is intended to account for the meaningful aspects of the ordering of words and phrases in language. The principles that determine which words and phrases are connected, how they are connected, and what effect that has on the ordering of the parts of sentences are called a *grammar*.

There are many different theories of syntax and ideas about how syntax works. Individual linguists are usually attached to particular schools of thought depending on where they were educated, what languages and problems they work with, and to a large degree their personal preferences.

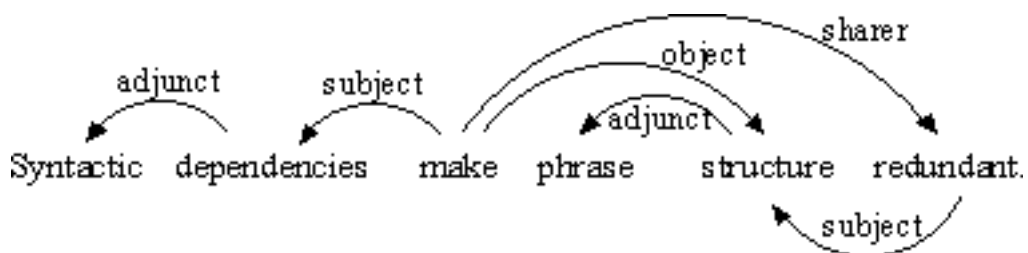
Most theories of syntax fall into two broad categories that reflect different histories, priorities and theories about how language works: *dependency grammars* and *constituency grammars*. The divide between these two approaches dates back to their respective origins in the late 1950s, and the debate between them is still active. Both schools of thought represent the connections within sentences as trees or directed graphs, and both schools agree that representing those connections requires that implicit structural information is made explicit. Relationships between words cannot be trivially represented as a sequence of tokens, and designing software that recognizes those relationships in texts is a challenging problem.

4.5.1. Dependency grammar

In dependency grammars, connections are usually between words or tokens, and the edges that join them have labels from a small set of connection types determined by some theory of grammar. Figure 7.2, “A dependency analysis” is a dependency analysis from a particular dependency grammar theory.

Dependency grammars tend to be popular among people working with languages in which word order is very flexible and words are subject to complex morphological agreement rules. It has a very strong tradition in Eastern Europe, but is also present elsewhere to varying degrees in different countries.

Figure 7.2. A dependency analysis



A dependency analysis of the sentence *Syntactic dependencies make phrase structure redundant.* This analysis uses the *Word Grammar* framework and is taken from the Word Grammar website [<http://www.phon.ucl.ac.uk/home/dick/WG/WG4PG/intro.htm>].

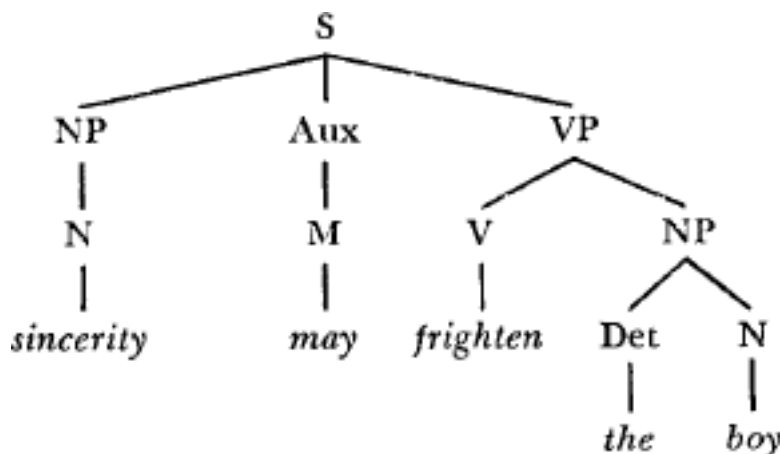
4.5.2. Constituency grammar

Constituency grammars view the connections between words as a hierarchical relationship between phrases. They break sentences up into parts, usually but not always continuous ones, and then break each part up into smaller parts, until they reach the level of individual tokens. The trees drawn to demonstrate constituency grammars reflect this structure. Edges in constituency grammars are not usually labeled.

Constituency grammar draws heavily on the theory of formal languages in computer science, but tends to use formal grammar in conjunction with other notions to better account for phenomena in human language. For example, there are prominent theories of syntax that include formalized procedures for tree rewriting, type hierarchies and higher-order logics, among other features. Few linguists currently believe the theory of formal languages alone can account for syntax.

As a school of thought, constituency grammar is historically associated with the work of Noam Chomsky and the American linguistic tradition. It tends to be popular among linguists working in languages like English, in which morphological agreement is not very important and word orders are relatively strictly fixed. It is very strong in English-speaking countries, but is also present in much of western Europe and to varying degrees in other parts of the world. See Figure 7.3, “A constituency analysis” for an example of a constituency analysis of an English sentence.

Figure 7.3. A constituency analysis



A constituency analysis of the sentence *Sincerity may frighten the boy*. This analysis is taken from [Chomsky1965].

The labels on edges in Figure 7.2, “A dependency analysis” and on tree nodes in Figure 7.3, “A constituency analysis” form a part of the specific syntactic theories from which these examples are drawn. Fuller explanations for them and their meanings are to be found in the referenced texts. Specific syntactic parsers and linguistic tools may use entirely different labels based on different ideas about language, even when the general form of the syntactic representations they generate resemble the examples here.

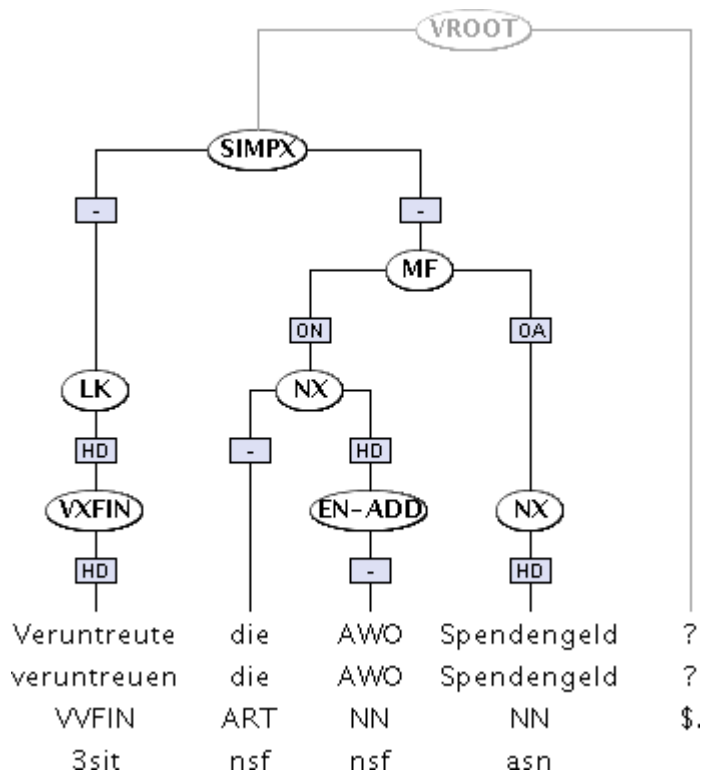
4.5.3. Hybrid grammars

Among computational linguists and people working in natural language processing, there is a growing tendency to use hybrid grammars that combine elements of both the constituency and dependency traditions. These grammars take advantage of a property of constituency grammars

called *headedness*. In many constituency frameworks, the phrases identified by the grammar have a single constituent that is designated as its *head*. A constituency analysis where all phrases have heads, and where all edges have labels, is broadly equivalent to a dependency analysis.

Figure 7.4, “A hybrid syntactic analysis” is a tree from the TüBa-D/Z treebank of German [http://www.sfs.uni-tuebingen.de/en/ascl/resources/corpora/tuebadz.html], and its labels are explained in [Hinrichs 2004]. This treebank uses a hybrid analysis of sentences, containing constituents that often have heads and edges with dependency labels. As with many hybrid analyses, not all constituents have heads, and some edges have empty labels, so it is not completely compatible with a strict dependency framework. However, the added information in the edges also makes it incompatible with a strict constituency framework. This kind of syncretic approach tends to find more acceptance in corpus and computational linguistics than in theoretical linguistics.

Figure 7.4. A hybrid syntactic analysis



A hybrid syntactic analysis of the German sentence *Veruntreute die AWO Spendengeld?* Display provided by the TIGERSearch [http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/] application [Lezius 2002].

4.5.4. Syntactic parsers

Syntactic parsing is the process (either automated or manual) of performing syntactic analysis. A computer program that parses natural language is called a *parser*. Superficially, the process of parsing natural language resembles parsing in computer science – the way a computer makes sense of the commands users type at command lines, or enter as source code for programs – and computing has borrowed much of its vocabulary for parsing from linguistics. But in practice, natural language parsing is a very different undertaking.

The oldest natural language parsers were constructed using the same kinds of finite-state grammars and recognition rules as parsing computer commands, but the diversity, complexity

and ambiguity of human language makes those kinds of parsers difficult to construct and prone to failure. The most accurate and robust parsers today incorporate statistical principles to some degree, and have often been trained from manually parsed texts using machine learning techniques.

Parsers usually require some preprocessing of the input text, although some are integrated applications that perform preprocessing internally. Generally, the input to a parser must have sentences clearly delimited and must usually be tokenized. Many common parsers work with a PoS-tagger as a preprocessor, and the output of the PoS-tagger must use the tags expected by the parser. When parsers are trained using machine learning, the preprocessing must match the preprocessing used for the training data.

The output of a parser is never simply plain text – the data must be structured in a way that encodes the connections between words that are not next to each other.

Implementations of parsers:

Berkeley parser [<http://nlp.cs.berkeley.edu/>] 

A constituency parser for English, German and many other languages. This parser optionally creates hybrid dependency output.

OpenNLP parser [<http://opennlp.apache.org/>] 

A constituency parser for English.

BitPar [<http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/BitPar.html>]

A constituency parser for German and English.

Bohnet parser [<http://code.google.com/p/mate-tools/>]

A dependency parser for English, German, Spanish and Chinese that is part of the Mate tools.

Alpino parser [<http://www.let.rug.nl/vannoord/alp/Alpino/>]

A hybrid parser for Dutch.

4.5.5. Chunkers

High-quality parsers are very complex programs that are very difficult to construct and may require very powerful computers to run or may process texts very slowly. For many languages there are no good automatic parsers at all.

Chunkers, also known as *shallow parsers*, are a more lightweight solution [Abney1991]. They provide a partial and simplified constituency parse, often simply breaking sentences up into clauses by looking for certain kinds of words that typically indicate the beginning of a phrase. They are much simpler to write, more robust, and much less resource-intensive to run than full parsers, and are available in many languages.

Implementations of chunkers:

RACAI chunker [<http://www.racai.ro/>] 

A chunker for Romanian and English.

Illinois chunker [http://cogcomp.cs.illinois.edu/page/software_view/13]

A chunker for English written in the Java programming language.

4.6. Word sense disambiguation (WSD)

Words can have more than one meaning. For example, the word *glass* can refer to a “drinking glass” and to the “material substance glass”. People can usually figure out, given the context of a word, which of a word's many meanings are intended, but this is not so easy for a computer. The automatic identification of the correct meaning of a word in context (sometimes called its *sense*) is called *word-sense disambiguation* (WSD).

Automatic WSD usually uses a combination of a digitized dictionary – a database of words and possible meanings – and information about the contexts in which words are likely to take particular meanings. Programs that do this often employ machine learning techniques and training corpora. Inputs to WSD programs are usually tokenized texts, but sometimes PoS tagging and even parsing may be required before disambiguation.

An implementation of a WSD tool:

UKB [<http://ixa2.si.ehu.es/ukb/>]

A graph based WSD and word sense similarity toolkit for English.

4.7. Coreference resolution and anaphora

Coreferences occur when two or more expressions refer to the same thing. Usually, linguists talk about coreferences only in those cases where the syntactic and morphological rules of the language do not make it clear that those expressions necessarily refer to the same thing. Instead, speakers have to use their memory of what has already been said, and their knowledge of the real world context of communication, to determine which words refer to the same thing and which ones do not.

An *anaphoric expression* is a particular case of coreference that can be resolved by identifying a previous expression that refers to the same thing. For example, in the sentence *The car stalled and it never started again* the word *it* refers to the car. But in the sentence *The car hit a truck and it never started again* it is not clear whether *it* refers to the car or the truck. In the sentences *John was out with Dave when he saw Mary. He thought Mary saw him, but she ignored him completely.* it is not clear whether any instance of *he* and *him* refers to John or Dave.

These kinds of ambiguities are what coreference resolution addresses. They can be very important in many NLP applications, like information retrieval and machine translation. Few automatic tools exist to resolve these kinds of problems and most use some form of machine learning.

Implementations for coreference resolution are:

BART (Beautiful Anaphora Resolution Toolkit) [<http://www.bart-coref.org/>]

A machine learning tool for coreference resolution. Currently supports English, Italian and German but may be trainable for other languages.

Stuttgart Coreference Resolver [<http://www.bart-coref.org/>]

A coreference resolver designed for the CoNLL 2012 coreference shared task [<http://conll.cemantix.org/2012/call-for-participation.html>].

4.8. Named entity recognition (NER)

Named entities are a generalization of the idea of a proper noun. They refer to names of people, places, brand names, non-generic things, and sometimes to highly subject-specific

terms, among many other possibilities. There is no fixed limit to what constitutes a named entity, but these kinds of highly specific usages form a large share of the words in texts that are not in dictionaries and not correctly recognized by linguistic tools. They can be very important for information retrieval, machine translation, topic identification and many other tasks in computational linguistics.

NER tools can be based on rules, on statistical methods and machine learning algorithms, or on combinations of those methods. The rules they apply are sometimes very *ad hoc* – like looking for sequences of two or more capitalized words – and do not generally follow from any organized linguistic theory. Large databases of names of people, places and things often form a part of an NER tool.

NER tools sometimes also try to classify the elements they find. Determining whether a particular phrase refers to a person, a place, a company name or other things can be important for research or further processing.

A special application of NER is *geovisualization* – the identification of place names and their locations. Knowing that a named entity refers to a particular place can make it much easier for computers to disambiguate other references. A reference to *Bismarck* near a reference to a place in Germany likely refers to the 19th century German politician Otto von Bismarck, but near a reference to North Dakota, it likely refers to the small American city of Bismarck.

Implementations of NER tools:

German NER [http://www.nlpado.de/~sebastian/software/ner_german.shtml]



A NER tool for German based on the Stanford Named Entity Recognizer.

Stanford NER [<http://nlp.stanford.edu/software/corenlp.shtml>] 

A NER tool for English distributed as part of the Stanford Core NLP toolkit.

SemiNER [<http://www.lsv.uni-saarland.de/personalPages/gchrupala/seminer.html>]

A NER tool for German trained from large corpora and Wikipedia data.

4.9. Sentence and word aligners

Aligners are tools mostly used with bilingual corpora – two corpora in different languages where one is a translation of the other, or both are translations from a single source. Statistical machine translation programs use aligned corpora to learn how to translate one language into another.

Alignments can be at different levels. Sentence aligners match the sentences in the two corpora, while word aligners try to align individual words. These programs are usually statistical in nature, and are typically language independent.

Implementations of aligners:

GIZA++ [<http://code.google.com/p/giza-pp/>]

A word-level aligner designed to work for all language pairs.

Gargantua [<http://gargantua.sourceforge.net/>]

A sentence-level aligner designed to work for all language pairs, that works well when the two texts are not sentence-for-sentence translations.

5. Manual annotation and analysis tools

Automatic annotation is not the end goal of most computational linguistics – linguistic tools are used to construct new resources and do linguistic research. Manual annotation and analysis tools touch on topics that have little place in automatic annotation technology: Visualization, usability, and human factors in linguistic analysis.

5.1. Manual annotation tools

The earliest linguistic corpora were annotated by adding information to the raw text manually. Human annotation is still performed, sometimes to correct imperfect automatic annotation software, and sometimes because no adequate automatic software exists. Human-corrected annotation, which is presumed to have fewer errors than automatic annotation, is used to create *gold standard* corpora, which are very important for creating, training and evaluating automatic annotation tools. See Chapter 5, *Quality assurance*, but see also Section 2, “Automatic and manual analysis tools”.

Annotating corpora manually is slow work and one of the roles of linguistic tools is making that task easier.

Manual annotation tools may involve one or more levels of analysis. Many specialized tools are oriented towards particular kinds of annotation and research corpora, while others are very general and applicable to many different linguistic theories and kinds of analysis.

Implementations of manual annotation tools:

Annotate [<http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/annotate.html>]

Annotate is a corpus annotation tool that is freely available for research purposes. It allows linguists to annotate the syntactic structure of sentences either manually or with the help of external tools like taggers and parsers, and see the results in an easy-to-read format on screen. Annotated corpora are stored in a sharable relational database and there is a permissions control scheme for users, granting them individual read and/or write access to each project. Multiple corpora can be hosted in a single Annotate installation.

Annotate works with input encoded in several versions of the NeGra corpus [<http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/>] format. Annotation tags can be specified for different levels of analysis, i.e. morphological tags, PoS tags, syntactic tags, etc.

SALTO [<http://www.coli.uni-saarland.de/projects/salsa/page.php?id=software>]

SALTO is a graphical tool for manual annotation of treebanks. It has been designed to work with syntactic trees and frame semantics annotation, but has also proven useful for other kinds of annotation.

Data is imported into SALTO in either the TIGER XML format or in its own SALSA XML format [Erk/Pado 2004]. Nevertheless as the TIGERSearch-related tool TIGERRegistry can convert a number of different treebank formats into TIGER XML, this makes it possible to use many different input sources. The output format is SALSA XML, an extension to TIGER XML. The added annotations can not be displayed or edited by TIGER tools.

5.2. Annotated corpus access tools

Searching and retrieving the contents of annotated corpora is one of the major supports that linguistic tools bring to general linguistic research. Relatively few specialized tools exist for this purpose, and many linguists use simple text tools like *grep* [Bell1979] to do research. Nonetheless, a number of applications are available for search and retrieval specifically with annotated language data.

Implementations of an annotated corpus access tools:

TIGERSearch [<http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/>]

TIGERSearch is a particularly widely used application for searching in treebanks. Users can import existing corpora into TIGERSearch and then query them using a lightweight search language developed specifically for TIGERSearch.

The results of queries are displayed visually and can be easily inspected. Matching patterns are clearly highlighted, so very small structures are quickly identified even in large trees. Results can be exported either as images (JPG, TIF, SVG, etc.) or as XML files. The statistics export module can be used to compute the frequency distribution for specific constructs in the query.


TIGERSearch uses the TIGER-XML format to encode corpora, but import filters are available for other popular formats such as Penn Treebank, NeGra, Susanne, among others.

ANNIS2 [<http://www.sfb632.uni-potsdam.de/d1/annis/>]

ANNIS2 is a versatile web browser-based search and visualization architecture for complex multilevel linguistic corpora with diverse types of annotation.

DDC-Concordance [<http://www.ddc-concordance.org/>]

DDC-Concordance is a tool for searching corpora with and without morphological markup. It is cross-lingual, but lemma searching is limited to English, German and Russian. Client APIs are available for Perl, Python, C and PHP.

Corpus Query Processor (CQP) [<http://cwb.sourceforge.net/>] 

CQP is a query engine for annotated corpora. It supports a range of annotation types and queries, including regular expressions and matches over words that are not next to each other, along with functions for marshalling results for convenient viewing.

6. Multimedia tools

Multimedia tools cover a wide variety of programs – mostly not specialized for linguistics – used to store, annotate, search and edit video and audio data. We will discuss only a few tools designed for use in linguistics:

Elan [<http://www.lat-mpi.eu/tools/elan/>]

ELAN is a tool for the creation of complex annotations in video and audio resources. With ELAN, a user can add an unlimited number of free-form annotations to audio and/or video data. An annotation can be, for instance, a sentence, word or morpheme, or a gloss, a comment, a translation, or a tag or description of any feature observed in the media. Annotations can be created on multiple layers, called tiers, that can be hierarchically interconnected and can correspond to different levels of linguistic analysis. Users can also mark and annotate gestures.

The annotation is always created and stored in files which are separate from the multi-media files. The textual content of annotations is always represented in Unicode and the annotation files are in an Elan-specific XML format. Annotation can be imported from and exported to a variety of other formats, including Shoebox/Toolbox, CHAT, Transcriber (import only), Praat and comma or tab-delimited text files. Export is also possible to interlinear text, HTML, SMIL and subtitle text.

Lexus [<http://www.lat-mpi.eu/tools/lexus>]

Lexus is a web-based tool for creating and editing multimedia lexical databases. A lexical entry in Lexus can describe different linguistic aspects of a word, like its part-of-speech, along with dictionary-style information like examples, as well as encyclopedic and ethnographic information. In Lexus, lexical entries can also contain images, sounds, and video files, to illustrate meanings or as an example of the usage of the word.

Lexus-built lexica make it possible to link linguistic and cultural concepts together in a way which conventional electronic language resources cannot easily manage. In addition, Lexus supports structural linguistic dependencies between words at all levels of analysis.

Lexus is of primary interest for language documentation projects since it offers the possibility to not just create a digital dictionary or thesaurus, but an entire multimedia encyclopedic lexicon. It also supports working collaboratively from different locations through a web-based interface.

WaveSurfer [<http://www.speech.kth.se/wavesurfer/>]

WaveSurfer is a tool for manually annotating sound files. It provides different visualizations of audio data – waveform or spectrogram display – and enables pitch contour and formant calculation and visualization. It supports different file formats for import and export, including WAVES or TIMIT. It is also possible to specify a tagset for annotation labels. Tagsets are saved in human readable form, and can be manually modified with a simple text editor. Annotation labels can be queried and replayed to the user.

WaveSurfer does not support hierarchical annotations. Also, no automatic transcriptions are provided, but results of automatic transcription processes in one of the supported formats can be imported and manipulated by the tool.

EXMARaLDA [<http://www.exmaralda.org/>]

EXMARaLDA (Extensible Markup Language for Discourse Annotation) is a system of data formats and tools for the computer assisted transcription and annotation of spoken language, and for the construction and analysis of spoken language corpora.

The EXMARaLDA Partitur Editor is a tool for inputting, editing and outputting transcriptions in partitur (musical score) notation. The EXMARaLDA Corpus-Manager is designed to assemble transcripts created with the EXMARaLDA Partiture-Editor with their corresponding recordings into corpora and enrich them with metadata. Metadata can be about speakers, communications (settings), recordings and the actual transcripts. The EXMARaLDA query tool EXAKT (EXMARaLDA Analysis and Concordancing Tool) is a tool for searching transcribed and annotated phenomena in an EXMARaLDA corpus.

All EXMARaLDA data is stored in Unicode-compliant XML files. EXMARaLDA data can be transformed into a number of widely used presentation formats and supports several important transcription systems (HIAT, GAT, CHAT, DIDA).

WebMAUS [<https://webapp.phonetik.uni-muenchen.de/BASWebServices/>]

The CLARIN-D center at the Bavarian Archive for speech Signals (BAS) [<http://www.bas.uni-muenchen.de/Bas/>] has made available several new web services that

incorporate the functionality of the MAUS (Munich AUtomatic Segmentation System) [<http://www.phonetik.uni-muenchen.de/Bas/BasProjectseng.html#MAUS>] segmentation tool. MAUS allows the fully automatic segmentation of speech recordings, given some form of written transcript. In a nutshell MAUS transforms the written text into a sequence of canonical phonemes, then produces a hypothesis model of possible pronunciation variants based on this canonical form, and finally decodes the speech signal into the most likely variant together with the optimal segmentation into phonemic and word units.

MAUS was developed in the late 1990s and has been maintained by BAS since then. To make it easier to use, several web services are now available that allow scientists to use MAUS over the Internet without the hassle of installation.

7. Recommendations for CLARIN-D tool designers

CLARIN-D provides an infrastructure for distributing linguistic resources, including tools for annotation, visualization and creation of language data. For tool owners and developers interested in taking advantage of CLARIN-D's infrastructure, there are some important recommendations to follow:

- For tools in the form of a downloadable and executable application, providing a compact delivery format, installation instructions and documentation is very strongly recommended. Contact the CLARIN-D technical help desk [<http://de.clarin.eu/en/training-helpdesk/technical-helpdesk.html>] for assistance in the process.
- For tools in the form of web-services, the functions and data structures that the tool uses should be clearly documented.
- All tools should be made available via a CLARIN-D center or, if they are hosted elsewhere, have accurate metadata available to CLARIN-D (see Chapter 2, *Metadata*).

Chapter 8. Web services: Accessing and using linguistic tools

Thomas Zastrow, Kathrin Beck, Scott Martens, Universität Tübingen

The previous chapter discusses the diversity of linguistic tools, a consequence of the diversity of human languages and linguistic theories, as well as their hierarchal interdependency. This chapter discusses web services and how they are used to organize complex systems of linguistic tools in a way that makes them as accessible as possible to users. The WebLicht system, a service-oriented architecture and execution environment for linguistic research built and hosted by CLARIN-D, has been designed make this interdependent system of resources available through standard Internet browsers with a minimum of user training and technical knowledge.

1. Web Services

Web services are a response to a number of problems that have emerged in the development of computers and in the way people use computers. Computers were, initially, a very expensive and rare tool used only for highly specialized, mostly scientific applications. Now, they are the core tool of almost all academic and intellectual work. In much of the world, owning a computer or smartphone with access to the Internet is something most literate people can take for granted. This change in computer users and uses forced the developers of computer tools to change their practices many times, and web services are a recent iteration of this process of adaptation.

The first computers, built in the 1940s and early 1950s, could only run one program at a time, were not interconnected, and had very little ability to store information for long periods of time. Individual users entered the programs and data they wanted a computer to process from stored media – punch cards and later disks and tapes – and then waited for the computer to complete the program and output the result on paper, cards, disks, or some other media.

From very early on, computer engineers looked at ways to save users time and trouble and improve the efficiency of computers. Beginning in the mid-1950s, IBM sold hard drives so that data could be stored internally in computers and accessed when wanted. Data no longer had to be inputted each time users wanted to do something, and output could be directed to the same internal storage for further processing and analysis.

In the early 1960s, engineers began connecting computers so they could exchange data, first as part of the US air defense system and then in the SABRE air travel reservation system. And, starting in the mid-1960s, computers started to support multiple users at the same time, each with their own screen and keyboard, running different processes on different data on the same computer without interfering with each other.

As computers became more widespread, more powerful, and were expected to perform more complex and demanding tasks, this networked multi-user model replaced the older picture of computers as a single isolated machine with a single user. Each user had an individual screen and keyboard, but used them to access any number of powerful computers, which could be located anywhere from the next room to thousands of kilometres away, tied together by sophisticated computer-to-computer connections. By 1970, this model had evolved into the earliest form of the Internet and the first versions of the still widespread Unix operating system.

Networked, multi-user computers created a number of new practical problems: When a networked computers store and run different programs that depend on each other, they need to have common ways of communicating and structuring data so that they can interact correctly. And, large networks with many users require security systems to control access. Many of the particularities of Unix-style operating systems were implemented to address those problems, and the standards for computer communication and data interchange on the Internet target precisely those issues.

The introduction of personal computers in the 1980s in some ways represented a step backwards in computer engineering: Home and office computers were once again isolated from each other, usually lacking any internal storage, and only able to run one process at a time, directed by a single user who would have to wait for processes to finish. Developing software for these widespread desktop computers meant confronting all over again many of the problems that standard protocols and interfaces were originally developed to solve.

In addition, the desktop computer revolution introduced a whole new problem to computer engineering: Personal computer users are usually not technical experts and often have little or no access to technical experts. Installing, running and maintaining software on personal computers can be very challenging even for experts and this inhibited the release of complex software.

In the 1990s, desktop computers became much more powerful and were increasingly networked and connected to the Internet, but the solutions that had been developed for high-performance computers were not so easy to reimplement for desktop computers. High-performance computer systems have professional administrators to maintain them, and until roughly 1990, nearly all Internet-accessible computers were owned by governments, military contractors, big companies, and universities. New standards and solutions could be implemented quickly by a combination of bureaucratic directives and professional agreement among system operators. System administrators operated as an informal guild, in constant communication with each other over the Internet, documenting and distributing highly specific knowledge about problems and solutions often before computer vendors were even aware of them. For personal computers, this approach was impossible. Users are not computer professionals and cannot be expected to be constantly on top of the latest highly specialized technical knowledge.

Web services are one, partial solution to these problems. The Internet is built on a set of stable, mature protocols for computer-to-computer interaction. Any computer that connects to the Internet, by definition, supports those common protocols. The Internet has, therefore, become a reliable mechanism of providing services to practically all computers.

A web service is any computational service that is provided via the Internet using a standard interface, accessible to different applications and computing platforms. To use a web service, a computer needs to only support the standard Internet protocols. Providing a web service means never having to distribute a tool or other resource at all. The service itself is hosted on a server under the provider's control, and does not need to ever be ported or adapted to different environments.

Web services resolve a number of the problems highlighted above:

- No special software to install other than the ones already needed to access the Internet.
- Fewer problems with system compatibility, since programs are written to run on distant servers under the control of developers.
- Bypassing the limitations of desktop computers that lack memory or storage space.

These benefits, along with widespread high-speed Internet access, have made web services an increasingly preferred solution to the problem of making digital resources and tools available to diverse users.

2. Service-oriented architectures

A service-oriented architecture (SOA) is a system that brings together web services to provide a single system, offering users a single interface to multiple interoperable tools. It is not simply a program or a server but a way of finding, fitting together and presenting to users web services and tools. It is an entire system of data standards, interfaces and practices.

Service-oriented architectures have a number of common features that are generally present:

- Multiple services are orchestrated at runtime, when users request them.
- Services are not coupled together in a rigid, pre-determined way.
- Services are discoverable when users want to employ them.
- Services are indexed in an accessible registry.
- Services have standardized interfaces, so that they can interoperate and exchange data.
- Services are reusable in different environments.
- Services are distributed, running on different servers, connected by the Internet or other networking scheme.
- Users have access to all services supported by the architecture through a single documented programming interface, and a single, widely supported front end like a web browser.

3. WebLicht – A service-oriented architecture for linguistic resources and tools

WebLicht (Web-based Linguistic Chaining Tool) [<https://weblicht.sfs.uni-tuebingen.de/>] is a web application built on service-oriented architecture principles that offers users access to linguistic resources. It incorporates a variety of new technologies to provide a single, user-friendly interface to a growing collection of tools and corpora. WebLicht is accessible from any Internet-connected computer with a recent web browser. Figure 8.1, “The WebLicht user interface, displayed in a browser” is a screenshot of the WebLicht user interface, displayed in an ordinary web browser. No special software needs to be installed on user's computers and WebLicht has no particular system requirements beyond those of standard web browser software.

WebLicht services can also be accessed directly from user-created and third party applications that support the required protocols, although at present there are no such applications. This is one of the principles of web service development and service-oriented architectures.

This section will describe the WebLicht system and its functioning.

Figure 8.1. The WebLicht user interface, displayed in a browser



3.1. Tool chains

WebLicht is organized around the notion of a *tool chain*, a succession of tools which

- work in a sequence on the same data,
- take the output of the preceding tool as their input,
- add information to these input data in a cumulative way, and
- do not alter either the primary data nor the data which were added by preceding tools.

Many linguistic tools perform specific and well-defined tasks, and for any particular task, there may be a variety of tools that do the same task but differ in underlying algorithm, basis in linguistic theory, or target language. The hierarchical nature of linguistic processing (see Chapter 7, *Linguistic tools*) means many tools rely on lower level tools to function. The “chain of tools” concept follows from this situation, and WebLicht offers users a robust and user-friendly way to assemble these small tools into processes.

For example, to parse a corpus, the user first uploads a corpus using their browser or selects a corpus already stored by WebLicht, if necessary selecting a format converter as the first tool in the processing chain to import it into the TCF format used internally by WebLicht (see Section 3.2, “Interoperability and the Text Corpus Format”). Currently, WebLicht provides format converters for plain text, RTF and Microsoft Word input.

From the menu, users then select tools one after another, organizing them into a processing chain that performs the required steps to parse the corpus: Tokenizing, then lemmatizing, performing part-of-speech tagging, morphological analysis, and parsing. WebLicht verifies that the input requirements for each tool are satisfied by the previous tools in the chain. For example, it ensures that the tagset used for part-of-speech tagging matches the tag set required as input for a parser. WebLicht’s internal processing format guarantees that the output of each tool requires no additional formal conversion to be compatible with the next tool in the chain. For additional

examples of using WebLicht and constructing tool chains, see Section 4, “WebLicht usage scenarios”.

Processes can be very long and complex and take time to run. WebLicht provides computing resources to complete the task, without reducing the user's ability to perform other tasks on their computer.

For various reasons, some tools cannot be combined. Many tools have specific input requirements that other tools do not meet, for example a parser may require a particular part-of-speech tagset, and no alternative tagger program will do. Also, many combinations of tools may not make any sense, like combining a named entity recognizer with a word sense disambiguation tool. WebLicht ensures that users can only build chains where the input requirements for each tool are satisfied by the output of the previous ones in the chain. As long as the input and output requirements of each service are being met, WebLicht can combine services from different research groups, hosted and running on different servers, into one robust tool chain.

3.2. Interoperability and the Text Corpus Format

In order for tools provided through WebLicht to exchange data, they must share a common data format. To make sure that all tools are interoperable, WebLicht has implemented a common interchange format for digitized texts: the Text Corpus Format (TCF, [Heid et al. 2010]). Although oriented originally to written language data, TCF has been extended to other media and is intended to provide interoperability between linguistic tools of all types.

TCF is similar in purpose and broadly compatible with LAF and GrAF (see Section 2, “Exchange and combination of annotations”), but more narrowly designed for use in a web-based service oriented architecture. Because TCF only provides layers for supported tools, and the LAF and GrAF formats are open-ended formats that support arbitrary kinds of annotations, there is not necessarily complete interoperability between these data formats. Conversion may not be lossless, and tools for converting specific annotations are only available where a converter for those specific kinds of annotations has been provided.

Before WebLicht can process linguistic data, it must be encoded in TCF. WebLicht is developing converters for many common linguistic data formats, both for input into WebLicht and to allow users to export their data to exchange formats common in the corpus linguistics community. The CLARIN-D consortium, and the resource centres in particular, will offer help and can provide standard converters for many commonly used data formats. As more and more tools and resources are integrated into WebLicht, more conversion tools will be available to tool providers.

TCF is used exclusively as internal processing format designed to support efficient data sharing and web service execution. Using TCF ensures interoperability between WebLicht tools and resources. It is strongly recommended that all tools in WebLicht accept input and produce output in TCF format.

TCF is an XML-based stand-off format (see Section 1.1, “Inline vs. stand-off annotations”). Annotations are added to linguistic data by appending new sections to already present TCF-encoded data. Whenever a new layer of annotation is introduced into WebLicht, it is encoded as a new layer independent of the existing layers. Using WebLicht to annotate TCF-encoded materials never changes or erases the original data or information added by previous tools in a chain, it only adds new information which further processing can use or ignore.

TCF's multi-layered stand-off annotation ensures that TCF and WebLicht are independent from specific linguistic theories. There are no assumptions about the linguistic theories underlying

the data or the annotations added to it. It is equally compatible with all approaches to language, so long as the tools and resources are formally compatible with WebLicht.

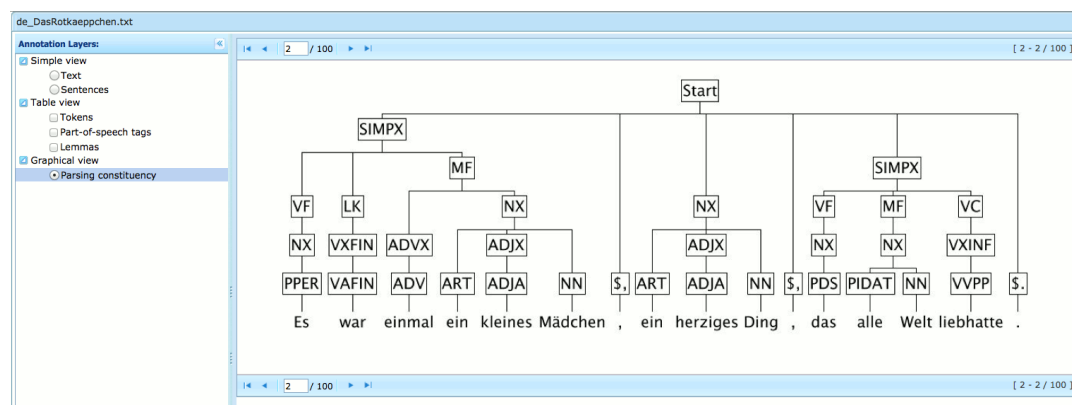
Furthermore, TCF's layering model makes it very easy for tool providers to specify which kinds of annotations are a pre-requisite for a specific tools. For example, if a parser requires texts to be PoS tagged using the Penn Treebank format – a common requirement of several English syntactic parsers – WebLicht can check if a section providing PoS annotations in that format is present in the given TCF file.

3.3. Visualization

For many annotation layers, WebLicht offers visualization tools. These tools are available to tool designers and users automatically if their tools output annotations in a compatible TCF layer. They do not have to develop their own visualization schemes.

Figure 8.2, “WebLicht constituency parse visualization” is a screenshot of WebLicht displaying a constituency parse tree from a syntactical annotation layer in a TCF file.

Figure 8.2. WebLicht constituency parse visualization



3.4. Metadata

Through WebLicht, the CLARIN-D resource centres provide users with information about available services, including details about input requirements and output specifications of each tool. This information is stored as metadata, and every tool integrated into WebLicht must be accompanied by appropriately structured metadata. CMDI metadata descriptions for WebLicht tools must be made available and stored at one of the CLARIN-D data repositories (see Section 6, “The Component Metadata Initiative (CMDI)”). Tool providers must produce such metadata. The CLARIN-D resource centres offer support and assistance in providing this information and correctly formatting it.

Every web service must also be assigned an individual *persistent identifier* (PID). PIDs can come from any existing PID assignment system, but they must be *unique*. Tool providers can also ask for support from a CLARIN-D resource centre in obtaining a PID.

3.5. Security

WebLicht is accessible via the Shibboleth SSO system (see Section 1, “Single Sign-on access to the CLARIN-D infrastructure”). This guarantees that every member of an academic institution which is part of the CLARIN identity federation, can access WebLicht using her institutional

account. Furthermore, the SSO infrastructure assures that only members of the academic community can access WebLicht and its web services.

4. WebLicht usage scenarios

The WebLicht infrastructure can be used in many different contexts and the use cases described here by no means exhaust the possibilities for use. The purpose of WebLicht is to provide easy access to linguistic tools and a relatively simple means for developers to distribute their linguistic tools. In any circumstance where a ready-made linguistic tool could be deployed, WebLicht is at least in principle available.

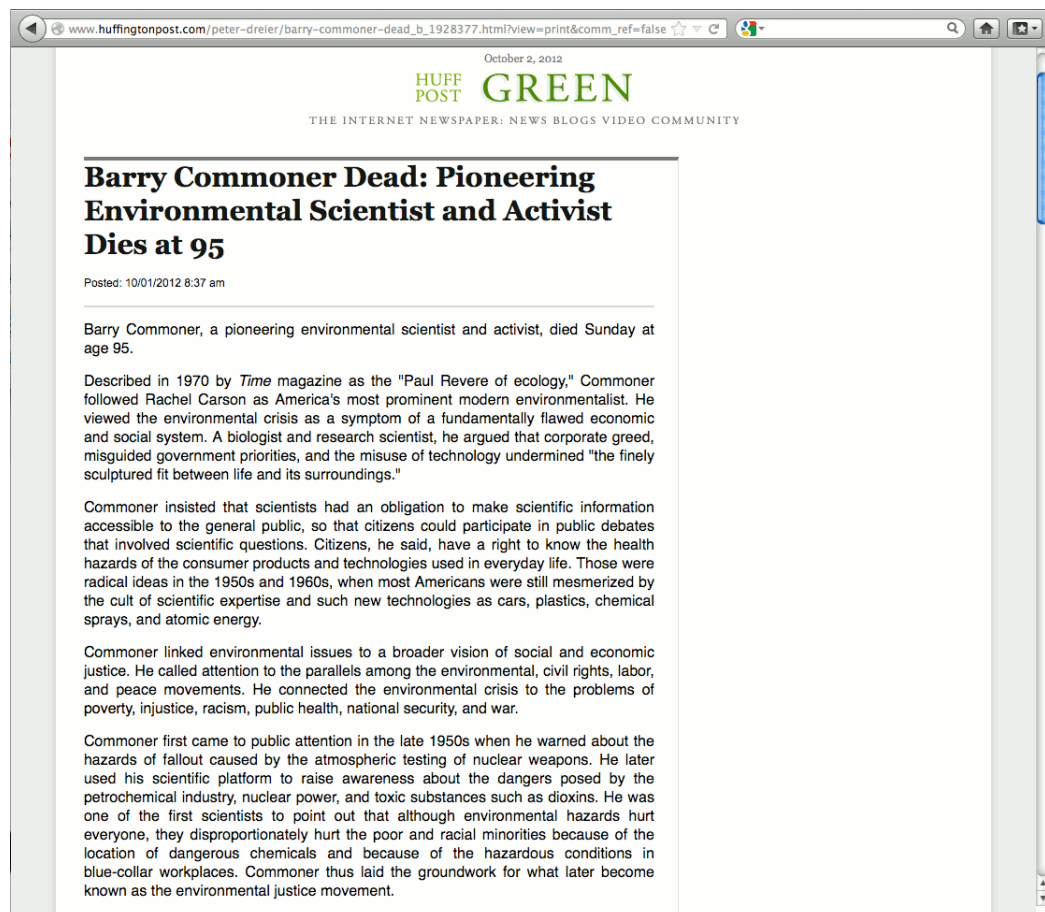
4.1. Quick annotation

WebLicht provides a platform for quickly and reliably annotating arbitrary texts using standard tools. This is a boon for classroom use and for any kind of linguistic research, since it makes full annotation available quickly with little or no preprocessing and no tool preparation.

The example below describes the quick construction of a lemmatized and PoS tagged resource from an arbitrary short text. This task requires no software installation beyond a standard Internet browser, and takes minutes to demonstrate and perform. Tool processing time varies, depending on the tool itself.

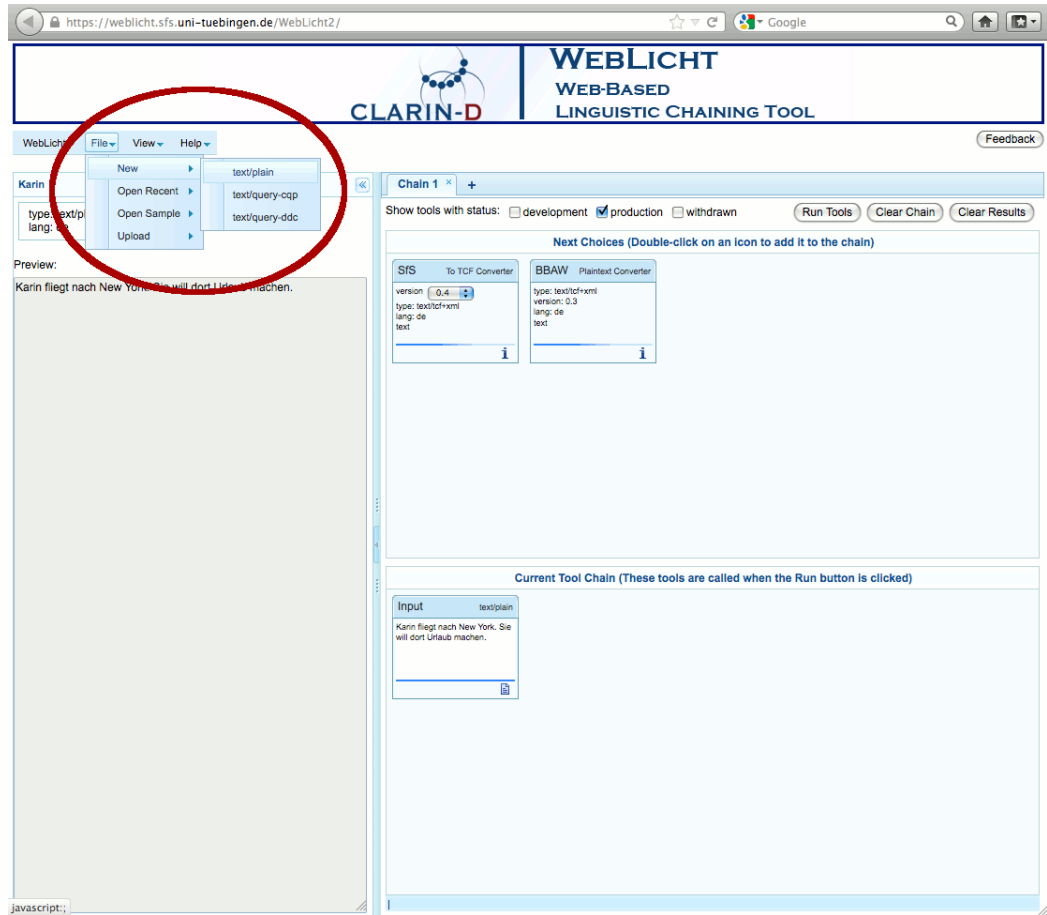
1. Find a short text on the Internet.

Figure 8.3. A short text from a news website



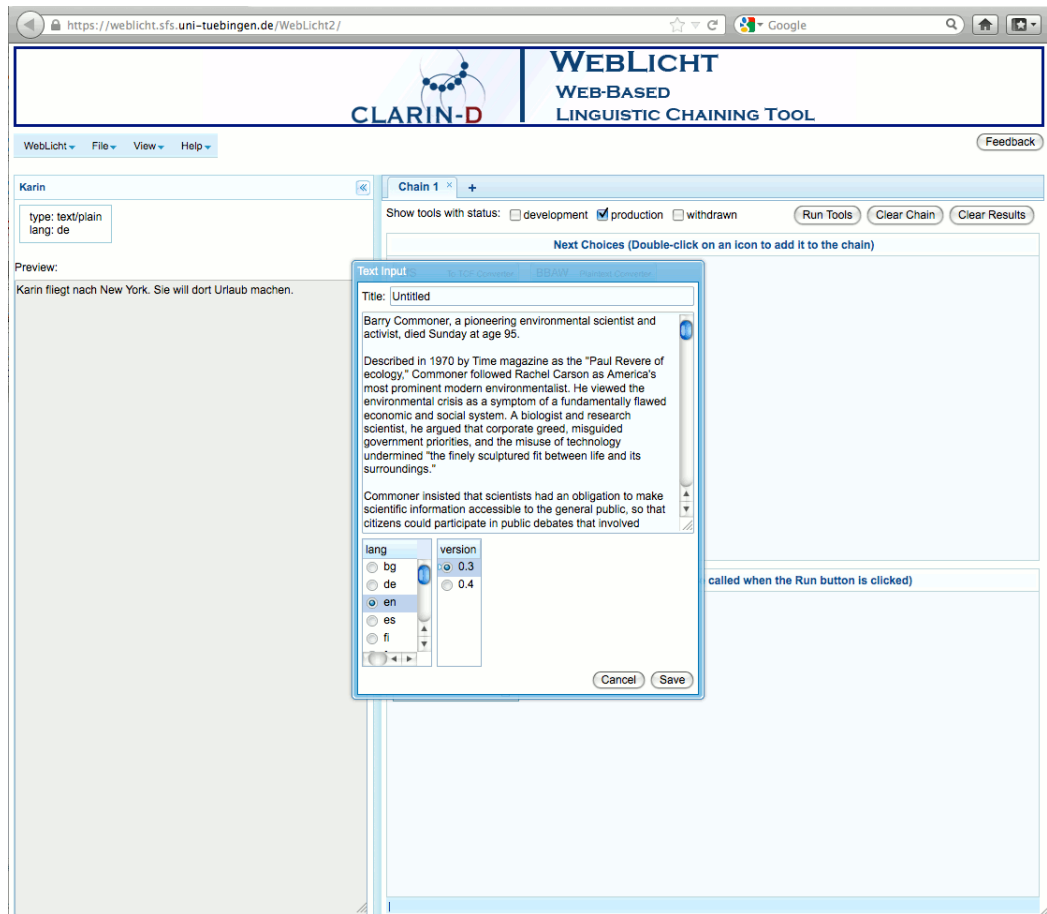
2. Open WebLicht in a browser, and select from the menu: File → New → text/plain.

Figure 8.4. WebLicht start-up



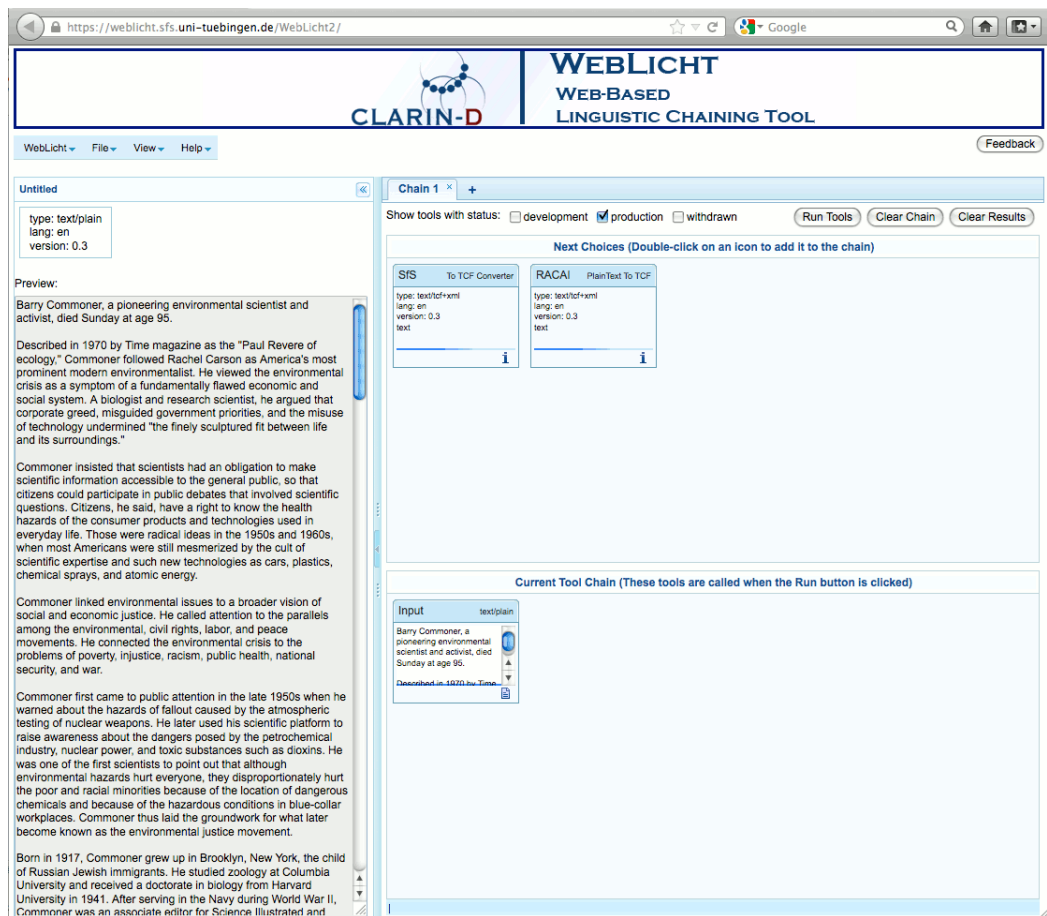
3. Copy the text from the article and paste it into the window, select the language of the text and other options as appropriate, then click the Save button.

Figure 8.5. Inputting a short text



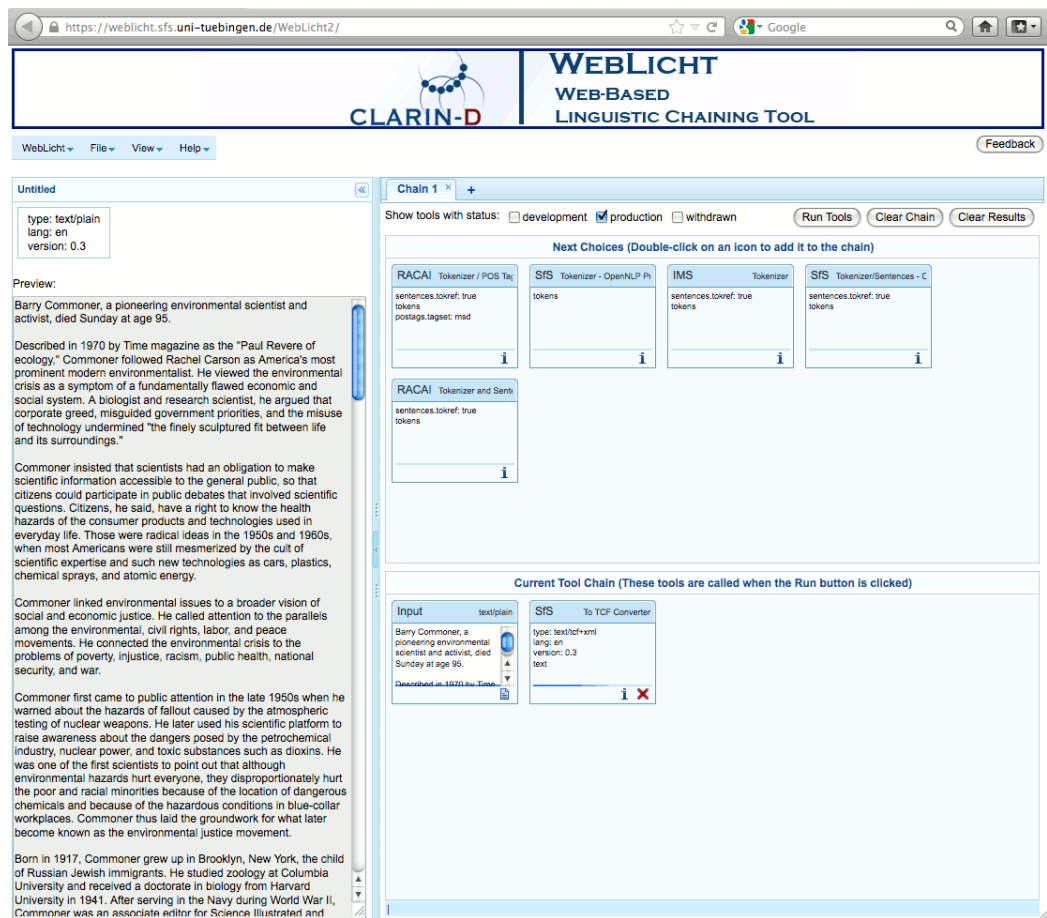
4. The next step is to construct a tool chain. Available tools are always in the upper window, labeled Next Choices. Tools can be added to a chain by double-clicking them, or by dragging them from the upper window to the lower (labeled Current Tool Chain).

Figure 8.6. Constructing a tool chain



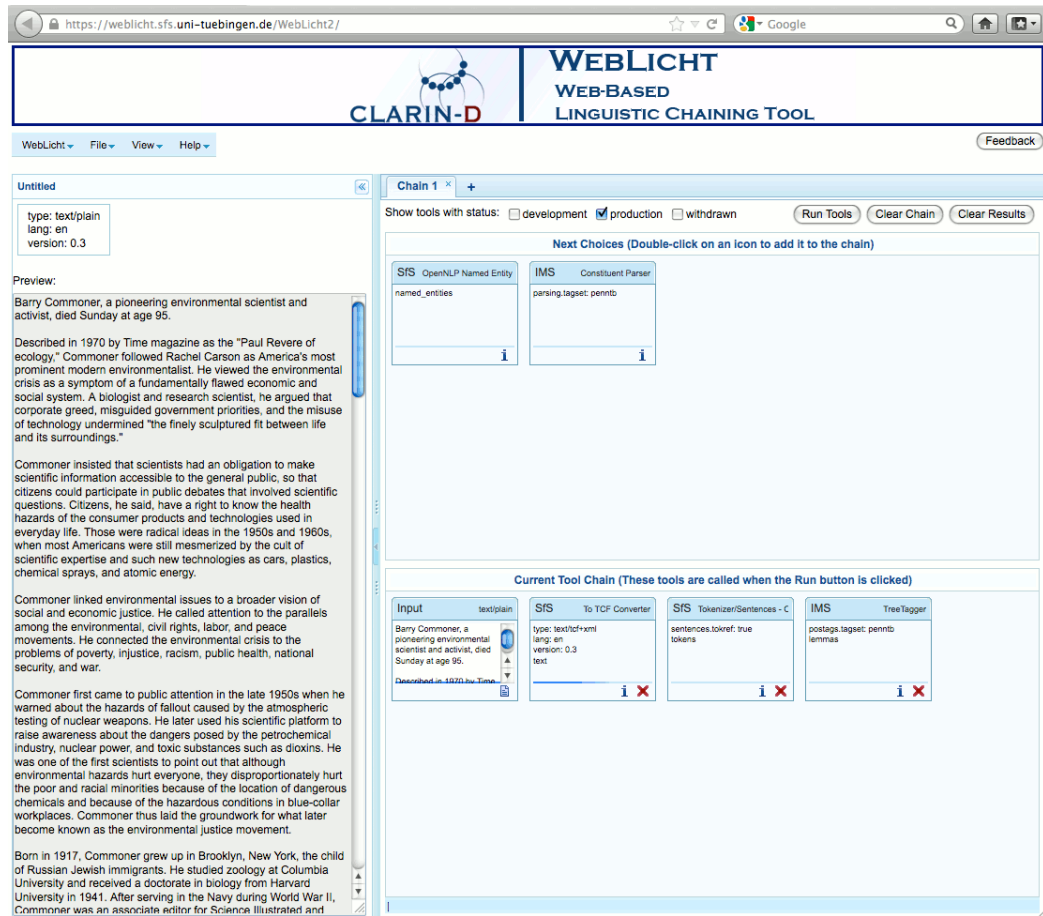
5. Most text processing tools require conversion into TCF format, so first select the text to TCF converter.

Figure 8.7. Adding a TCF converter to a tool chain



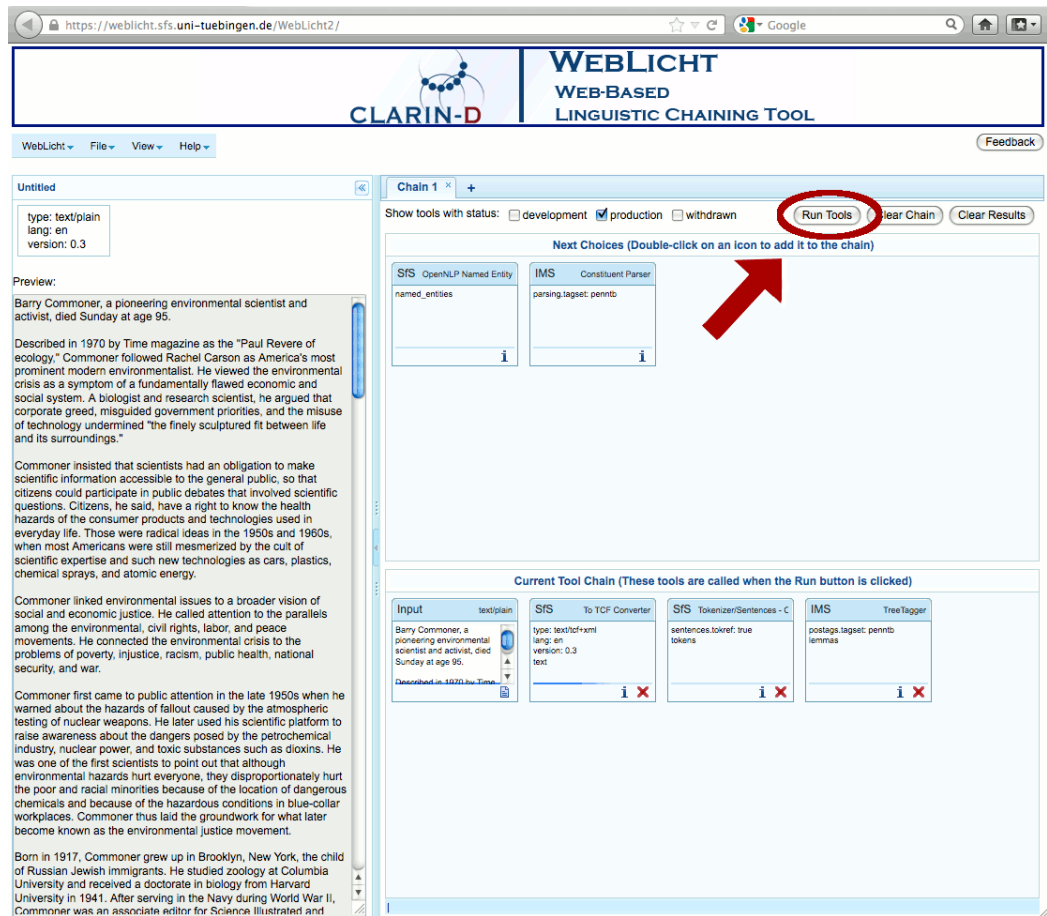
6. The Next Choices window now contains tools that can be chained after the TCF converter. Select the tools that will need to run to produce the desired annotated resource. For example, select in sequence the SfS Tokenizer/Sentence Splitter, then the IMS TreeTagger. This produces a valid, complete annotation chain.

Figure 8.8. Adding a tokenizer, sentence splitter, lemmatizer and tagger to a tool chain



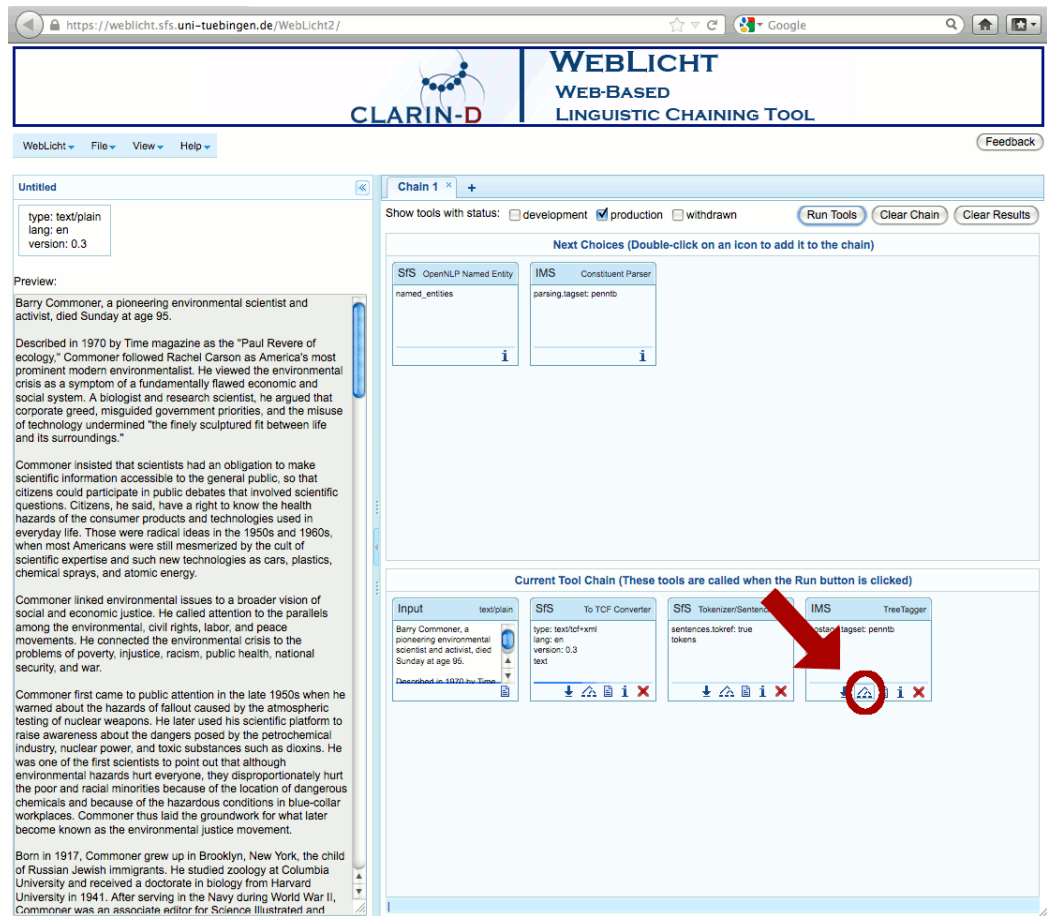
7. Click Run Tools. Processing time depends on the underlying tools and the facilities available to run them. WebLicht displays the current state of the text in processing chain and signals when it has finished running.

Figure 8.9. Running the tool chain on the text



8. WebLicht has facilities for visualizing the output of the processing chain directly. Click the visualization icon on the last element of the chain to inspect the results.

Figure 8.10. Visualization



9. Click the download icon on the last member of the chain after processing is completed to download the TCF file produced at the end of the processing chain. This file is in XML format with distinct and well-documented tags encompassing all the information produced by the tools in the chain. Further processing and analysis can be performed from this file.

Figure 8.11. Downloading the results



- 10.(optional) WebLicht also includes parsers and other common linguistic tools, some of which take a much longer time to run. To fully parse the text, all that is necessary is to add a parser tool to the end of the tool chain and wait for it to finish running.

Figure 8.12. Adding a parser to the tool chain

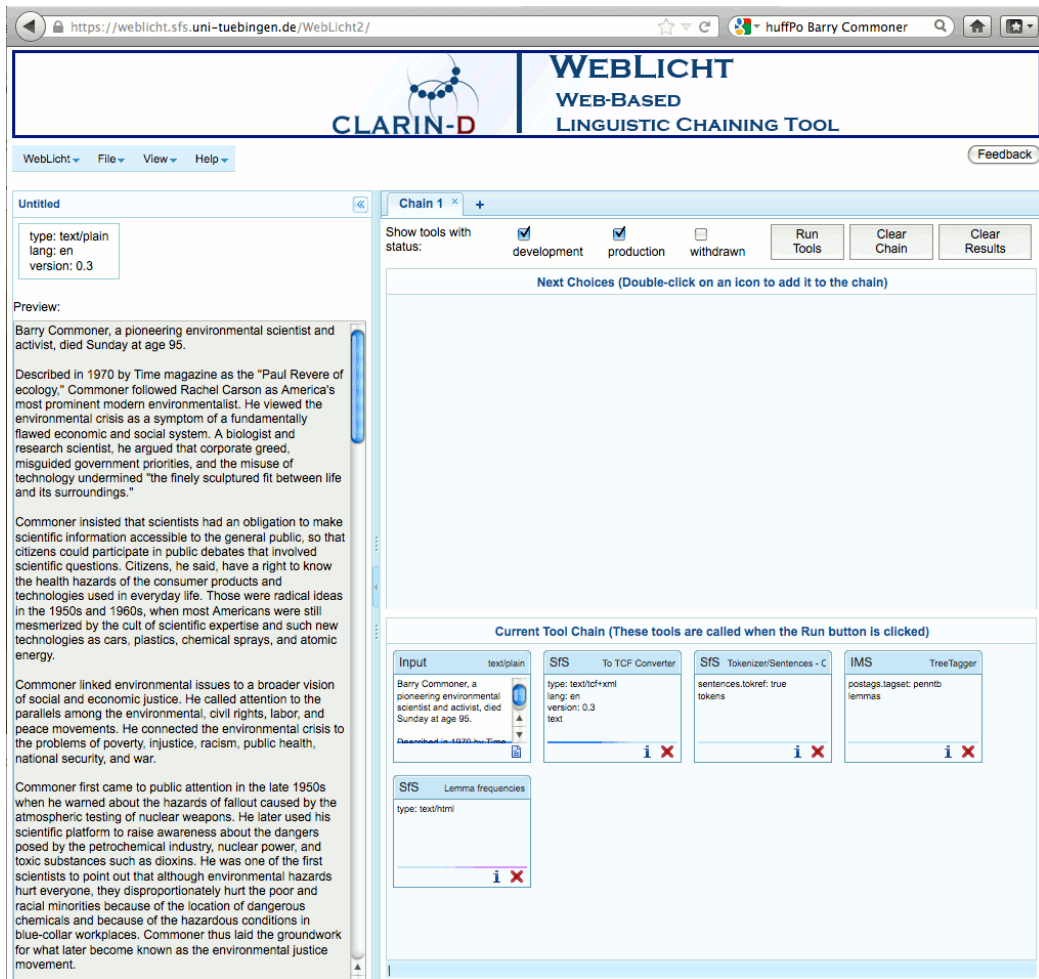


4.2. Statistics

WebLicht is also able to incorporate facilities for doing statistical analysis as part of the tool chain. All that is required is to add an analysis tool to the chain. For example, the tool chain from the previous section can include a lemma count tool.

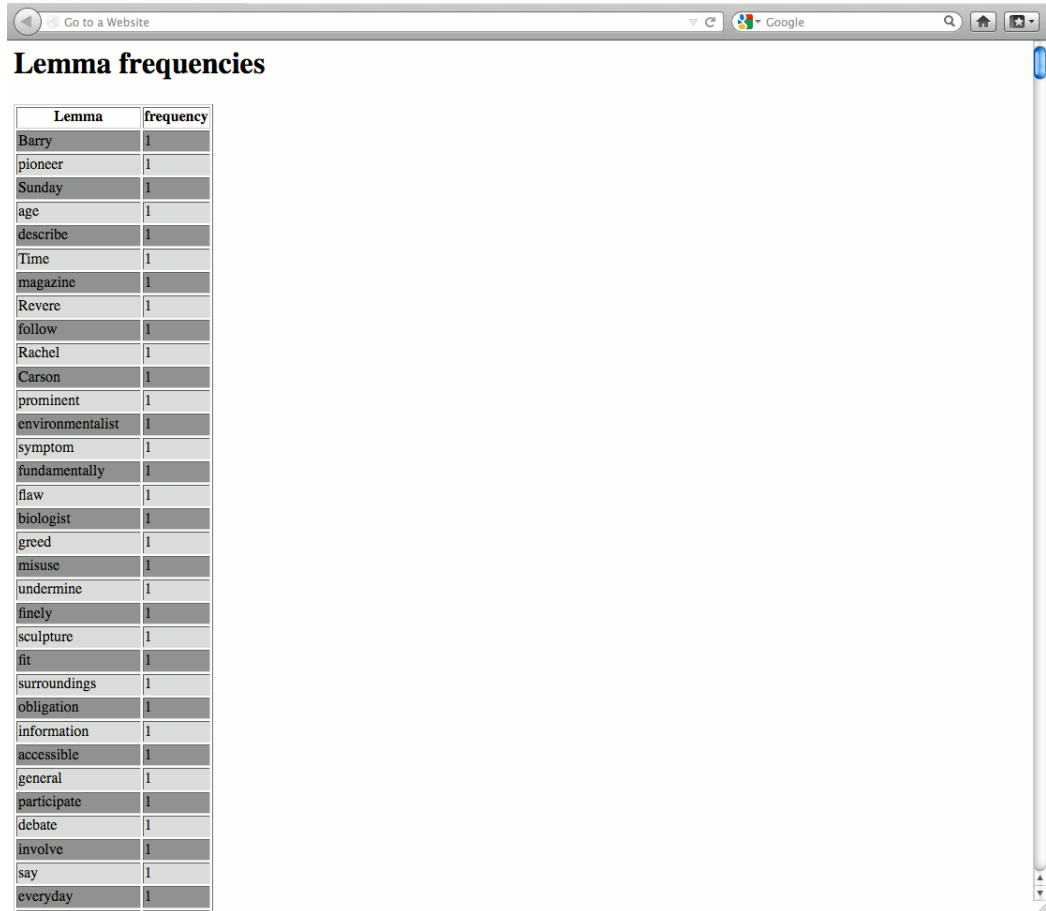
1. Add the Lemma Frequency Tool to a processing chain that includes a lemmatizer.

Figure 8.13. Adding a lemma frequency tool to the chain



2. Once processing is completed, the ordered list of lemmas and frequencies is available for viewing and downloading.

Figure 8.14. Viewing word frequency data in a table



The screenshot shows a web browser window with the address bar displaying 'Go to a Website' and the Google logo. The page title is 'Lemma frequencies'. Below the title is a table with two columns: 'Lemma' and 'frequency'. The table contains 30 rows, each with a lemma and a frequency of 1. The lemmas are: Barry, pioneer, Sunday, age, describe, Time, magazine, Revere, follow, Rachel, Carson, prominent, environmentalist, symptom, fundamentally, flaw, biologist, greed, misuse, undermine, finely, sculpture, fit, surroundings, obligation, information, accessible, general, participate, debate, involve, say, and everyday.

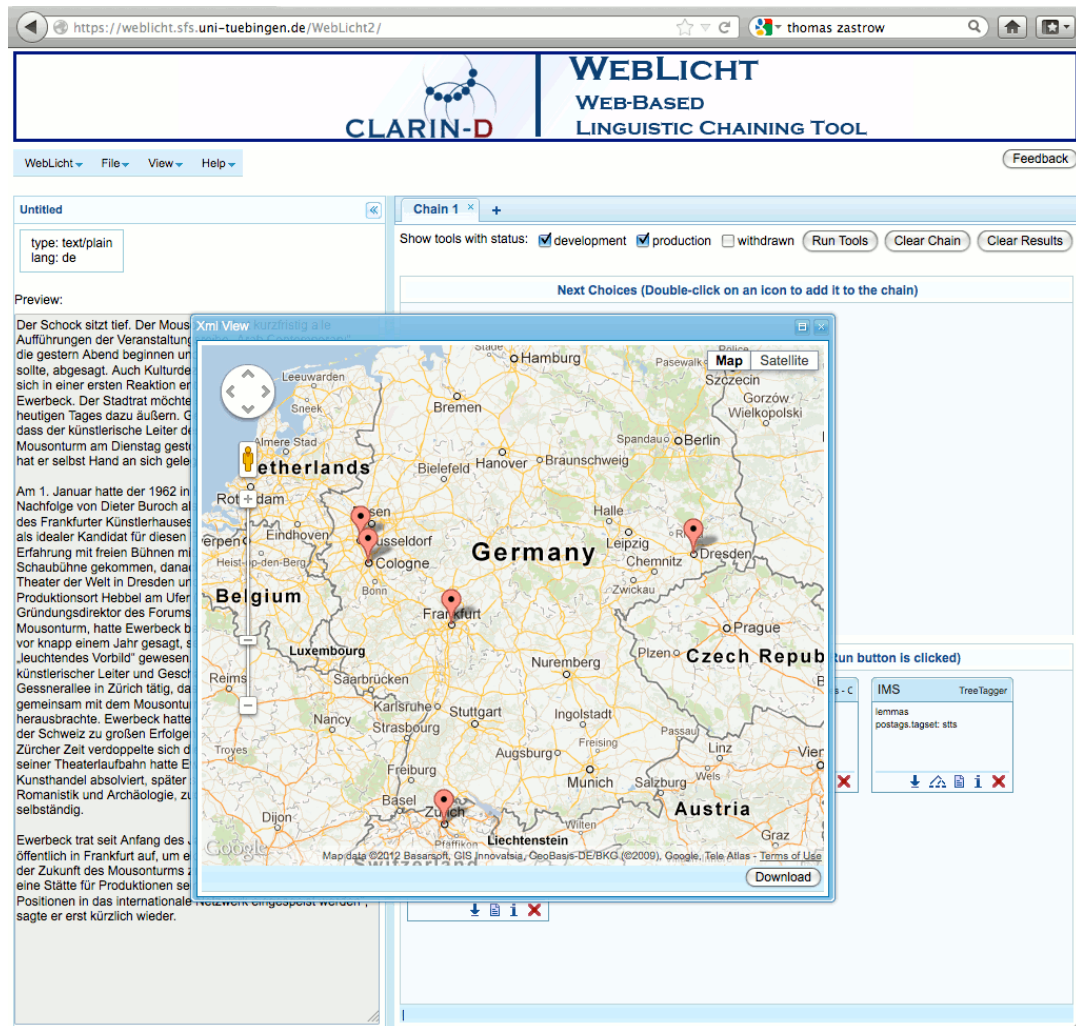
Lemma	frequency
Barry	1
pioneer	1
Sunday	1
age	1
describe	1
Time	1
magazine	1
Revere	1
follow	1
Rachel	1
Carson	1
prominent	1
environmentalist	1
symptom	1
fundamentally	1
flaw	1
biologist	1
greed	1
misuse	1
undermine	1
finely	1
sculpture	1
fit	1
surroundings	1
obligation	1
information	1
accessible	1
general	1
participate	1
debate	1
involve	1
say	1
everyday	1

WebLicht also has tools for extracting PoS distributions, displaying histograms and other statistical visualizations, and a variety of statistical analysis tools in progress.

4.3. Geovisualization

Another application for WebLicht is data transformation for more complex visualization. Geovisualization extracts placenames from annotated texts and displays them on a map. Figure 8.15, “Viewing placenames from a text on a map” is an example of a geovisualization from a German newspaper article, processed in a few seconds using WebLicht.

Figure 8.15. Viewing placenames from a text on a map



5. Integrating existing linguistic tools into WebLicht

This section is addressed to tool providers who want to integrate their tools into WebLicht. CLARIN-D can provide advice and assistance, time and resources permitting, through the CLARIN-D technical help desk [<http://de.clarin.eu/en/training-helpdesk/technical-helpdesk.html>] and individual centres. More detailed information about the technical specifications for WebLicht services and tutorials for creating and WebLicht tools are available through the WebLicht website [<https://weblicht.sfs.uni-tuebingen.de/>].



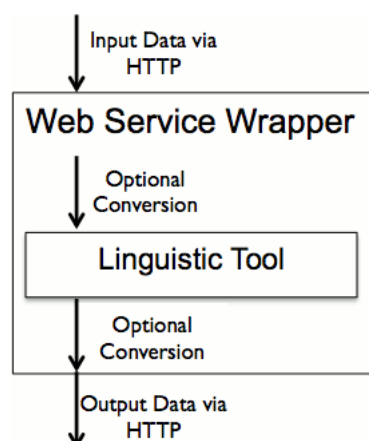
RESTstyle architecture

Before integration into WebLicht, linguistic tools and resources must have a standard web service interface. Web service standards are defined by the W3C, which considers a web service to be “a software system designed to support interoperable machine-to-machine interaction over a network.” [<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>]

Web services can be implemented in several ways, and there are several established standards and best practice systems. WebLicht uses the RESTstyle architecture [Fielding 2000], which is well suited to highly scalable systems of independent software components, like large collections of independently authored linguistic tools. Every WebLicht-integrated tool *must* be implemented as RESTstyle web service.

Rewriting linguistic tools as web services may be time-consuming, complicated, and, in the case of tools with intellectual property restrictions, impossible. Therefore, it is common practice to construct a *wrapper* around an existing tool. A web service wrapper is a program that is implemented as a web service and invokes the existing tool in response to users' input. The wrapper often must convert user input from the formats provided by the web service system to the formats expected by the tool, and then converts the output into the format expected by the web service or the user (see Figure 8.16, “Web service wrapper”).

Figure 8.16. Web service wrapper



There are no fixed limitations on the programming languages used for WebLicht tools. The only strict technical requirements for WebLicht integration are

1. interfaces that follow the requirements for a RESTful architecture, and
2. metadata descriptions in the CMDI format (see Section 6, “The Component Metadata Initiative (CMDI)”).

Although not as strictly required, it is very strongly preferred that all web services available through WebLicht employ the TCF format described in section Section 3.2, “Interoperability and the Text Corpus Format”. Devising a wrapper for a new tool should generally mean devising a robust data converter between TCF and that tool's usual processing format.

Generally, tool integration into WebLicht can be accomplished using any programming language and software development framework. The Java EE programming environment [<http://www.oracle.com/technetwork/java/javae/overview/index.html>] (particularly version 6 and above) and the Apache Tomcat web application server [<http://tomcat.apache.org/>] are good best practices for building web services for WebLicht.



Checklist for WebLicht integration

The following steps have to be performed in order to successfully integrate a tool or resource into WebLicht:

- Implement the tool as RESTstyle web service or embed it in a wrapper which acts as RESTstyle web service and make it accessible via standard internet protocols.
- Tools that process textual information should use the TCF format for input and output whenever possible.
- Describe the web service in the CMDI metadata format and assign a PID to it. Examples of CMDI files can be found at any CLARIN-D repository which hosts web services.
- The CMDI file has to be stored at one of the CLARIN-D centre repositories.
- Decide whether you want to host your tool or web service at your site or whether you want to draw on the hosting facilities of CLARIN-D. Creator-hosted resources can still be part of WebLicht, as long as server availability is high and the hosting system is able to handle a sufficiently large number of simultaneous calls.

Bibliography

- [Abney1991] Steven Abney. 1991. *Parsing by chunks. Principle-based parsing*. Kluwer Academic Publishers.
- [Artstein/Poesio 2008] Ron Artstein and Massimo Poesio. 2008. “Inter-coder agreement for computational linguistics”. *Computational Linguistics*. 34. 4.
- [Beagrie 2001] Neil Beagrie. 2001. *Preserving UK digital library collections* [<http://dx.doi.org/10.1108/EUM0000000006955>]. *Program: electronic library and information systems*. 35. 3. 215-226.
- [Beißwenger et al. 2012] Michael Beißwenger, Maria Ermakova, Alexander Geyken, Lothar Lemnitzer, and Angelika Storrer. to appear. “A TEI Schema for the Representation of Computer-mediated Communication”. *Journal of the TEI*. 3.
- [Bell Labs 1979] . 1979. *UNIX™ time-sharing system. UNIX programmer’s manual* [<http://plan9.bell-labs.com/7thEdMan/>]. 7th edition.
- [Bies et al. 1995] Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. 1995.
- [Boyd et al. 2008] Adriane Boyd, Markus Dickinson, and Detmar Meurers. 2008. “On detecting errors in dependency treebanks”. *Research on Language and Computation*. 6. 2. 113-137.
- [Brants et al. 2002] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. *The TIGER Treebank* [<http://www.coli.uni-saarland.de/publikationen/softcopies/Brants:2002:TT.pdf>].
- [Burchardt et al. 2006] A. Burchardt, K. Erk, A. Frank, A. Kowalski, and S. Pado. 2006. *SALTO – a versatile multi-level annotation tool. Proceedings of LREC'2006, Genoa (IT)*.
- [Chiarcos 2008] Christian Chiarcos. 2008. *An ontology of linguistic annotations. LDV Forum (=Journal for Computational Linguistics and Language Technology)*. 23. . 1-16.
- [Chiarcos 2010] Christian Chiarcos. 2010. *Grounding an ontology of linguistic annotations in the Data Category Registry. Proceedings of the LREC'2010 workshop on language resource and language technology standards*. State of the art, emerging needs, and future developments. Valetta (MT). 37-40.
- [Chomsky1965] Noam Chomsky. 1965. *Aspects of the theory of syntax*. The MIT Press.
- [Collins 1997] Michael Collins. 1997. *Three generative, lexicalised models for statistical parsing. Proceedings of the 35th annual meeting of the Association for Computational Linguistics (jointly with the 8th conference of the EACL)*. 16-23.
- [Compston1919] Herbert Fuller Bright Compston. *The inscription on the stele of Měša, commonly called the Moabite Stone*. 1919. Society for Promoting Christian Knowledge.
- [Coward/Grimes 2000] David F. Coward and Charles E. Grimes. 2000. *Making dictionaries. A guide to lexicography and the Multi-Dictionary Formatter* [http://www.sil.org/computing/shoebox/MDF_2000.pdf].
- [Dipper 2005] Stefanie Dipper. 2005. *XML-based stand-off representation and exploitation of multi-level linguistic annotation. Proceedings of Berliner XML Tage 2005 (BXML 2005)*. Berlin (DE). 39-50.

- [Engelberg/Lemnitzer 2009] Stefan Engelberg and Lothar Lemnitzer. 2010. *Lexikographie und Wörterbuchbenutzung*. 4th edition. Stauffenburg, Tübingen.
- [Erk et al. 2003] Katrin Erk, Andrea Kowalski, Sebastian Pado, and Manfred Pinkal. 2003. *Towards a resource for lexical semantics. A large German corpus with extensive semantic annotation. Proceedings of ACL 2003*. Sapporo (JP). 537–544.
- [Erk/Pado 2004] Katrin Erk and Sebastian Pado. 2004. *A powerful and versatile XML format for representing role-semantic annotation. Proceedings of LREC 2004*.
- [Fellbaum 1998] Christiane Fellbaum. 1998. *WordNet. An electronic lexical database*. MIT Press. Cambridge, MA.
- [Fielding 2000] Roy Thomas Fielding . 2000. *Architectural styles and the design of network-based software architectures* [http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf]. University of California (dissertation). Irvine.
- [Fischer 2005] Steven R. Fisher. 2005. Reaktion Books. *A history of writing*.
- [Furrer/Volk 2011] Lenz Furrer and Martin Volk. 2011. “<http://aclweb.org/anthology-new/W/W11/W11-4115.pdf>Reducing OCR errors in Gothic script documents”. 97-103. *Proceedings of Workshop on Language Technologies for Digital Humanities and Cultural Heritage (associated with RANLP 2011)*, Hissar, Bulgaria.
- [Garside et al. 1997] Roger Garside, Geoffrey Leech, and Anthony McEnery. 1997. *Corpus annotation. Linguistic information from computer text corpora*. Addison Wesley Longman.
- [Geyken 2007] Alexander Geyken. “The DWDS corpus. A reference corpus for the German language of the 20th century”. 2007. Christiane Fellbaum. *Collocations and Idioms. Linguistic, lexicographic, and computational aspects*. Continuum. London. 23–41.
- [Geyken et al. 2011] Alexander Geyken, Susanne Haaf, Bryan Jurish, Matthias Schulz, Christian Thomas, and Frank Wiegand. 2012. “TEI und Textkorpora: Fehlerklassifikation und Qualitätskontrolle vor, während und nach der Texterfassung im Deutschen Textarchiv [<http://www.computerphilologie.de/jg09/geykenetal.pdf>]”. *Jahrbuch für Computerphilologie*. 9.
- [Geyken et al. 2012] Alexander Geyken, Susanne Haaf, and Frank Wiegand. 2012. “The DTA ‘base format’: A TEI-Subset for the Compilation of Interoperable Corpora [<http://www.oegai.at/konvens2012/proceedings.pdf#page=383>]”. Jeremy Jancsary. *11th Conference on Natural Language Processing (KONVENS) – Empirical Methods in Natural Language Processing, Proceedings of the Conference. Schriftenreihe der Österreichischen Gesellschaft für Artificial Intelligence*. 5.
- [Grosso et al. 2003] Paul Grosso, Eve Maler, Jonathan Marsh, and Norman Walsh. 2003. *XPointer Framework. W3C recommendation* [<http://www.w3.org/TR/xptr-framework/>].
- [König et al. 2003] Esther König, Wolfgang Lezius, and Holger Voormann. 2003. *TIGERSearch 2.1 user's manual* [<http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/pdf/manual.pdf>] . IMS (Universität Stuttgart).
- [Haaf et al. forthcoming] Susanne Haaf, Frank Wiegand, and Alexander Geyken. “Measuring the correctness of double-keying. Error classification and quality control in a large corpus of TEI-annotated historical text”. forthcoming. *Journal of the Text Encoding Initiative*.
- [Heid et al. 2010] Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs. 2010. *A corpus representation format for linguistic web services: the D-SPIN text corpus format and its relationship*

- with ISO standards*. Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10).
- [Hinrichs 2004] E. Hinrichs, S. Kübler, K. Naumann, H. Telljohann, and J. Trushkina. 2004. *Recent Developments in Linguistic Annotations of the TüBa-D/Z Treebank.. Proceedings of the Third Workshop on Treebanks and Linguistic Theories (TLT)..*
- [Hinrichs/Vogel 2010] Erhard Hinrichs and Iris Vogel. 2010. *Interoperability and standards* [<http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-51>] . CLARIN D5C-3.
- [Holley 2009] Rose Holley. “How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs [doi:10.1045/march2009-holley]”. 2009. *D-Lib Magazine*. 14. 3/4.
- [Ide 1998] Nancy M. Ide. *Corpus encoding standard. SGML guidelines for encoding linguistic corpora*. 1998. 463-470. *Proceedings of the First International Language Resources and Evaluation Conference, Granada, Spain*.
- [Ide et al. 2000] Nancy M. Ide, Patrice Bonhomme, and Laurent Romary. “XCES: An XML-based standard for linguistic corpora”. 2000. 825-830. *Proceedings of the Second language Resources and Evaluation Conference (LREC), Athens, 2000*. European Language Resources Association (ELRA).
- [Ide/Suderman 2007] N. Ide and K. Suderman. 2007. *GrAF: A graph-based format for linguistic annotations. Proceedings of the linguistic annotation workshop, held in conjunction with ACL 2007. Praha (CZ)*. 1-8.
- [IPA 1999] *Handbook of the International Phonetic Association. A guide to the use of the international phonetic alphabet*. 1999. Cambridge University Press.
- [ISO 639-3:2007] . 2007. *Codes for the representation of names of languages – Part 3: Alpha-3 code for comprehensive coverage of languages* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39534].
- [ISO 3166-1:2006] . 2006. *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39719].
- [ISO 3166-2:2007] . 2002. *Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39718].
- [ISO 12620:2009] . 2009. *Terminology and other content and language resources – Specification of data categories and management of a Data Category Registry for language resources* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37243].
- [ISO 24610-1:2006] . 2006. *Language resource management – Feature structures – Part 1: Feature structure representation* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37324].
- [ISO 24612:2012] . 2012. *Language resource management – Linguistic annotation framework (LAF)* [http://www.iso.org/iso/catalogue_detail.htm?csnumber=37326].
- [ISO 24613:2008] . 2008. *Language resource management – Lexical markup framework (LMF)* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37327].

- [ISO 24615:2010] . 2010. *Language resource management – Syntactic annotation framework (SynAF)* [http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=37329].
- [Jurafsky/Martin 2009] Dan Jurafsky and James Martin. 2009. *Speech and language processing*. Prentice Hall. 2nd edition.
- [Kahn/Wilensky 2006] Robert Kahn and Robert Wilensky. 2006. *A framework for distributed digital object services*. *International Journal on Digital Libraries*. 6. 2. 115-123.
- [Kunze/Lemnitzer 2007] Claudia Kunze and Lothar Lemnitzer. 2007. *Computerlexikographie. Eine Einführung*. Narr. Tübingen.
- [Kupietz et al. 2010] Marc Kupietz, Cyril Belica, Holger Keibel, and Andreas Witt. 2010. “The German Reference Corpus DEREKO. A primordial sample for linguistic research [http://www.lrec-conf.org/proceedings/lrec2010/pdf/414_Paper.pdf]”. 1848-1854. *Proceedings of the seventh conference on International Language Resources and Evaluation (LREC 2010)*. European Language Resources Association (ELRA).
- [Leech 1993] 1993. Geoffrey Leech. *Corpus annotation schemes*. *Literary and Linguistic Computing*. 8. 4. 275-281.
- [Lemnitzer/Zinsmeister 2010] Lothar Lemnitzer and Heike Zinsmeister. 2010. *Korpuslinguistik. Eine Einführung*. Narr. Tübingen. 2.
- [Lieberman et al. 2005] Henry Lieberman, Alexander Faaborg, Waseem Daher, and José Espinosa. 2005. *How to wreck a nice beach you sing calm incense*. *Proceedings of the International Conference on Intelligent User Interfaces (IUI 2005)*.
- [Lezius 2002] Wolfgang Lezius. 2002. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora*. University of Stuttgart Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS).
- [Lezius 2002] Wolfgang Lezius. 2002. *TIGERSearch – ein Suchwerkzeug für Baumbanken*. *Proceedings of Konvens 2002*. Saarbrücken (DE).
- [Lüdeling/Kytö 2009] Anke Lüdeling and Merja Kytö. 2009. *Corpus Linguistics*. An International Handbook. 2. de Gruyter. Berlin/New York. *Handbooks of Linguistics and Communication Science/Handbücher zur Sprach- und Kommunikationswissenschaft*. 29.2.
- [Lüngen/Sperberg-McQueen 2012] Harald Lüngen and Michael Sperberg-McQueen. *A TEI P5 document grammar for the IDS text model*. 2012. *Journal of the Text Encoding Initiative*. 3.
- [MacWhinney 2000] Brian MacWhinney. 2000. *The CHILDES project. Tools for analyzing talk. Part 1: The CHAT transcription format* [<http://childes.psy.cmu.edu/manuals/CHAT.pdf>]. 3rd edition (newer editions available online). Lawrence Erlbaum Associates. Mahwah, NJ.
- [Magerman 1994] David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Doctoral dissertation.
- [Martens 2011] Scott Martens. 2011. *Quantifying linguistic regularity*. Centrum voor Computerlinguïstiek, KU Leuven.
- [McEnery/Wilson 2001] Tony McEnery and Andrew Wilson. 2001. *Corpus linguistics. An introduction*. Edinburgh university press. Edinburgh. 2nd edition. *Edinburgh textbooks in empirical linguistics*.
- [Nartker et al. 2003] Thomas A. Nartker, Kazem Taghva, Ron Young, Julie Borsack, and Allen Condit. 2003. “OCR correction based on document level knowledge [<http://citeseer.ist.psu.edu/viewdoc/>]

- download?doi=10.1.1.74.4701&rep=rep1&type=pdf]". 103-110. *Proc. IS&T/SPIE 2003 Intl. Symp. on Electronic Imaging Science and Technology*.
- [NISO:2004] *Understanding Metadata* [<http://www.niso.org/publications/press/UnderstandingMetadata.pdf>]. . 2004.
- [Odijk/Toral 2009] Jan Odijk and Antonio Toral. 2009. *Existing evaluation and validation of LR*s [<http://www.flarenet.eu/sites/default/files/D5.1.pdf>] . FlaReNet D5.1.
- [Perkuhn et al. 2012] Rainer Perkuhn, Holger Keibel, and Marc Kupietz. 2012. *Korpuslinguistik*. Fink. Paderborn.
- [Porter 1980] Martin F. Porter. 1980. *An algorithm for suffix stripping*. *Program*. 14. 3. 130-137.
- [Przepiórkowski 2011] Adam Przepiórkowski. 2011. *Integration of language resources into web service infrastructure* [<http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-56>] . CLARIN D5R-3b.
- [Pytlik Zillig 2009] Brian L. Pytlik Zillig. "TEI analytics. Converting documents into a TEI format for cross-collection text analysis [<http://llc.oxfordjournals.org/content/24/2/187.full>]". 2009. 187-192. *Literary & Linguistic Computing*. 24. 2.
- [Riester et al. 2010] Arndt Riester, David Lorenz, and Nina Seemann. 2010. *A recursive annotation scheme for referential information status*. *Proceedings of the 7th International Conference of Language Resources and Evaluation (LREC)*. Valletta (MT). 717-722.
- [Ringersma/Drude/Kemp-Snijders 2010] 2010. J. Ringersma, S. Drude, and M. Kemps-Snijders. *Lexicon standards: From de facto standard Toolbox MDF to ISO standard LMF*. Talk presented at *LRT standards workshop, Seventh conference on International Language Resources and Evaluation (LREC'2010)*.
- [Romary et al. 2011] Laurent Romary, Amir Zeldes, and Florian Zipser. 2011. *<tiger2/> documentation* [http://hal.inria.fr/docs/00/59/59/13/PDF/tiger2_documentation_20100525.pdf]. Draft version as of May 25, 2011.
- [Russel/Norvig 2009] Stuart Russell and Peter Norvig. 2009. *Artificial intelligence. A modern approach*. Prentice Hall.
- [Saenger 1997] Paul Saenger. 1997. Stanford University Press. *Space between words: the origins of silent reading*.
- [Santorini 1990] Beatrice Santorini. 1990. *Part-of-speech tagging guidelines for the Penn Treebank project* [http://repository.upenn.edu/cis_reports/570/]. 3rd revision.
- [Schiller et al. 1999] Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)* [<http://www.sfs.uni-tuebingen.de/resources/stts-1999.pdf>].
- [Schmandt-Besserat 1992] Denise Schmandt-Besserat. 1992. University of Texas Press. *Before writing*.
- [Sinclair 1991] John McHardy Sinclair. *Corpus, concordance, collocation*. 1991. Oxford University Press. *Describing English language*.
- [Sinclair 2005] John Sinclair. 2005. *Corpus and text – basic principles*. Martin Wynne. *Developing linguistic corpora. A guide to good practice*. 1–16. Oxbow Books. Oxford.

- [Soria/Monacchini 2008] Claudia Soria and Monica Monacchini. 2008. *Kyoto-LMF WordNet representation format (version 4)* [http://www2.let.vu.nl/twiki/pub/Kyoto/TechnicalPapers/Kyoto-LMF_v04.pdf]. KYOTO working paper WP2/TR2.
- [Svensen 2009] Bo Svensen. 2009. *A handbook of lexicography. The theory and practice of dictionary-making*. Cambridge University Press. Cambridge (UK).
- [Tanner et al. 2009] Simon Tanner, Trevor Muñoz, and Pich Hemy Ros. “Measuring mass text digitization quality and usefulness. Lessons learned from assessing the OCR accuracy of the British Library’s 19th century online newspaper archive [doi:10.1045/july2009-munoz]”. 2009. *D-Lib Magazine*. 15. 7/8.
- [TEI P5] , Lou Bournard, and Syd Bauman. *TEI P5: Guidelines for electronic text encoding and interchange* [<http://www.tei-c.org/release/doc/tei-p5-doc/en/Guidelines.pdf>] .
- [Thompson/McKelvie 1997] H. S. Thompson and D. McKelvie. 1997. *Hyperlink semantics for standoff markup of read-only documents* [<http://www.ltg.ed.ac.uk/~ht/sgmleu97.html>]. *Proceedings of SGML Europe’97*. Barcelona (ES).
- [Unsworth 2011] John Unsworth. “Computational work with very large text collections. Interoperability, sustainability, and the TEI [<http://jtei.revues.org/215>]”. 2011. *Journal of the Text Encoding Initiative* . 1.
- [Windhouwer 2012] Menzo Windhouwer. 2012. *RELcat: a Relation Registry for ISOcat data categories* [<http://www.lrec-conf.org/proceedings/lrec2012/summaries/954.html>]. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12)*. European Language Resources Association (ELRA).
- [Yamada/Matsumoto 2003] Hiroyasu Yamada and Yuji Matsumoto. 2003. *Statistical dependency analysis with support vector machines*. *Proceedings of IWPT 3*.
- [Zipser/Romary 2010] Florian Zipser and Laurent Romary. 2010. *A model oriented approach to the mapping of annotation formats using standards*. *Proceedings of the Workshop on Language Resource and Language Technology Standards (LREC’2010)*. Malta (MT).
- [Zinsmeister et al. 2008] Heike Zinsmeister, Andreas Witt, Sandra Kübler, and Erhard Hinrichs. 2008. *Linguistically annotated corpora. Quality assurance, reusability and sustainability*. Anke Lüdeling and Merja Kytö. *Corpus linguistics. An international handbook*. 1. 759-776. Mouton de Gruyter. Berlin. *Handbücher zur Sprach- und Kommunikationswissenschaft*.
- [Zinsmeister 2010] Heike Zinsmeister. 2010. *Korpora*. K.-U. Carstensen, Ch. Ebert, C. Ebert, S. Jekat, R. Klabunde, and H. Langer. *Computerlinguistik und Sprachtechnologie. Eine Einführung*. 3rd edition. 482-491. Spektrum Akademischer Verlag. Heidelberg.