Chapter 5
Representing Annotated Texts as RDF

Abstract Text annotation consists in defining markables (elements to be annotated), their features (attributes and values of annotations) and relations between markables (e.g. syntactic dependencies or semantic links). In this chapter we describe the principles for annotating text data using RDF-compliant formalisms. These principles provide the basis for making annotated corporate and text collections accessible from the LLOD ecosystem.

5.1 Introduction

Linguistic analysis of natural language is basically about defining markables (elements of annotations), their features (attributes and values of annotations) and relations between markables (e.g. syntactic dependencies or semantic links).

Before discussing RDF-based data models for representing annotations, we discuss state-of-the-art formalisms in NLP and Digital Humanities (DH) to formalize markables and their annotations, and present possibilities to integrate LLOD-compliant references to textual and other natural language objects on the web.

5.1.1 Tab-Separated Values: CoNLL TSV

Since 1999, the Conference on Natural Language Learning (CoNLL)[1] established a highly successful series of shared tasks in NLP. Subsequently, the data formats employed in these tasks evolved into a widely used community standard for most forms of linguistic annotations, as illustrated in the following example (1), slightly simplified from a clause from the OntoNotes corpus [1], file `wsj-0655`:

(1) James Baker ... told reporters Friday: "I have no reason to deny reports that some Contras ambushed some Sandinista soldiers."

[1] http://www.conll.org/, last accessed 09-07-2019.

P. Cimiano et al., *Linguistic Linked Data*,
https://doi.org/10.1007/978-3-030-30225-2_5

```
James      NNP B-PERSON
Baker      NNP E-PERSON
told       VBG O
reporters  NNS O
Friday     NNP S-DATE
:          :   O
...
```

Fig. 5.1 CoNLL sample with `WORD`, `POS` and `NER` columns

```
(TOP (S (NP-SBJ (NNP James)
                (NNP Baker))
        (VP (VBD told)
            (NP (NNS reporters))
            (NP-TMP (NNP Friday))
            (: :)
```

(a)

```
James      NNP (TOP (S (NP-SBJ *
Baker      NNP *)
told       VBD (VP *
reporters  NNS (NP *)
Friday     NNP (NP-TMP *)
:          :   *
```

(b)

Fig. 5.2 Sample of a parsed sentence in original and CoNLL format. (**a**) Original format (**b**) CoNLL conversion

Figure 5.1 illustrates the CoNLL format for the example of parts of speech and named entity annotation. Every word is written in one line, with a series of tab-separated columns holding different annotations; one column contains the surface form of the word. Sentences are separated by an empty line; comments are marked by #. Along with word-level annotations, CoNLL formats support the annotation of spans, illustrated here for named entity annotation using the IOBES scheme, i.e. `B-`*X* marking the beginning of the annotation *X*, `E-`*X* its end, `I-`*X* intermediate elements, `S-`*X* a single-word annotation and `O` the absence of an annotation.

While word- and span-level annotations can be performed in an intuitive and extensible way with one column per annotation type, the phrase structure syntax can be handled indirectly in CoNLL, only. The original syntax annotation of the OntoNotes uses a bracketing format as illustrated in Fig. 5.2a: Every word is grouped together with its POS tag in a terminal node, e.g. `(NNP James)`. One or more terminal (or nonterminal) nodes may be grouped to a constituent, again marked by parentheses and a label, e.g. `(NP ...)`. The original format is agnostic about line breaks, but for convenience of reading, line breaks may be inserted to put one word (and its annotations) in one single line.

Figure 5.2b shows how this information is split into three different columns for compliance with the CoNLL format: `WORD`, `POS` and `PARSE`, respectively. The column `WORD` contains the current word form, `POS` the associated POS tag and `PARSE` an abbreviated version of the parse structure, where `*` replaces terminal nodes.

As the phrase structure syntax example in Fig. 5.3 illustrates, some phenomena require special handling in order to be representable in a CoNLL TSV format, but a key advantage is that this representation can be easily extended, and easily merged with additional columns. As an example, the conventional way to represent semantic

```
James      NNP B-PERSON (TOP (S (NP-SBJ * _          ARG0
Baker      NNP E-PERSON *)              _          ARG0
told       VBD O        (VP *           tell.v.01  rel
reporters  NNS O        (NP *)          _          ARG2
Friday     NNP S-DATE   (NP-TMP *)      _          ARGM-TMP
:          :   O        *               _          _
# skipped quote
```

Fig. 5.3 Integrated CoNLL representation of POS, NER, syntax and PropBank annotations

role annotations in CoNLL is to add one column for the predicate as well as another column for every predicate that identifies its arguments. In the fourth column of our example, semantic predicates are identified and marked by a sense identifier. For every predicate instance, its arguments (`ARG`*i* with numerical index *i* for core arguments and `ARGM` arguments for various modifiers) are represented in a separate column, indicating whether a word occurs in (the span of) a frame argument and in which role. For every predicate in a sentence, an additional column is created.[2]

CoNLL TSV formats are characterized by the use of one word per line, one tab-separated column per annotation layer and an empty line to separate sentences. While CoNLL-based formats are not generic, they are relatively simple and easy to parse (or at least, commonly known), as they have been designed to provide common output specifications for tools participating in these challenges, and with support from many tools—both participating in the original shared tasks but also their successors and competitors—the CoNLL formats ultimately evolved into de facto standards for many types of linguistic annotation. Probably the most influential CoNLL dialect at the moment is CoNLL-U, the format adopted by the Universal Dependency collection of annotated corpora [2].[3] Individual CoNLL dialects, however, differ in the definition, naming and order of columns; their annotations and tools developed on this basis are thus not mutually interoperable.

Aside from interoperability problems, the CoNLL format family suffers from inherent limitations; CoNLL is limited to annotate words and larger units of text, as tokens (words) constitute the minimal unit of analysis (i.e. lines). As a side-effect, CoNLL formats normally lose information about the original layout. In fact, column-based formats do not annotate primary data but rather a segmental annotation of the primary data, in particular, tokens extracted from the primary data, listed one per line. This approach has the disadvantage of imposing one layer of linguistic interpretation (e.g. what constitutes a token, sentence, etc.) that may not be desired by other users. In addition, the 'one token per line' assumption adopted in the CoNLL format can seriously handicap algorithm performance. For example, some

[2]While generally accepted, this adds to the complexity of the format: Unlike conventional TSV formats which form fixed-size tables, CoNLL tables have no predictable maximum width and their width may vary from one sentence to the next.

[3]http://universaldependencies.org/, last accessed 09-07-2019.

phenomena (e.g. dots in chemical formulas) need to be split apart for one processing step (e.g. POS tagging), but treated as a unit for others (e.g. syntactic parsing). In many cases, CoNLL-based NLP pipelines thus require transformations between different tokenizations in order to process a sentence. Yet, as CoNLL formats do not systematically preserve whitespace information, such transformations can be lossy. Furthermore, annotations with deviating segmentations cannot be easily aggregated into a single CoNLL file, and neither can reliable references between elements in a CoNLL file be established.[4]

Despite these limitations, TSV formats offer advantages for processing and are thus widely used. Indeed, some of their deficits can be easily compensated if words are complemented with a community-approved way to refer to textual objects on the web or elsewhere. URIs provide such a mechanism, and by adding an additional column that holds a URI that identifies the original string in the original document, it is possible to establish links, to facilitate information integration across different CoNLL dialects (resp., tools that generate or consume these), and with the original document, its metadata and details regarding its layout. Below, URI schemes for this purpose are introduced.

5.1.2 Tree-Based Formats: TEI/XML

In computational philology and parts of the language resource community, XML enjoys a high degree of popularity as a representation formalism, only recently being challenged by JSON. Both formats formalize tree (resp., multi-tree) data structures, so that much of what can be said about the XML-based specifications of the Text Encoding Initiative (TEI) below extends to other approaches to formalize linguistic annotations.

For background, motivations and applications of the Text Encoding Initiative see Chap. 13. Here, we focus solely on the format and its application to the linguistic annotation of texts. The TEI P5 guidelines provide generic datatypes for many forms of linguistic annotation, including elements for orthographic sentences (`<s>`), grammatical words (`<w>`) and grammatical phrases (`<phr>`), as well as attributes for their respective type (`@type`), interpretation (`@ana`) and

[4]Strategies employed by different CoNLL Shared Tasks involve ad hoc solutions such as a reference to a word by its number (id) in the sentence (in dependency syntax), explicit ids and co-indexation (for coreference), or off-set based solutions (for Semantic Role Labelling). Neither of these, however, permit *absolute* reference, but they are defined with respect to the current sentence (SRL, dependency syntax), a particular tokenization (dependency syntax) or ad hoc ids (coreference). State of the art are thus more generic data models grounded in labelled directed multigraphs [3].

```
<s type="sentence">
 <cl ana="#S">
  <phr ana="#NP-SBJ">
   <w ana="#NNP">James </w>
   <w ana="#NNP">Baker </w>
  </phr>
  <phr ana="#VP">
   <w ana="#VBD">told </w>
   <phr ana="#NP">
    <w ana="#NNS">reporters </w>
   </phr>
   <phr ana="#NP-TMP">
    <w ana="#NNP">Friday </w>
    <w ana="#colon">: </w>
     ...
   </phr>
  </phr>
 </cl>
</s>
```

Fig. 5.4 POS and syntactic annotation in TEI (inline annotation)

identification (`@xml:id`).[5] For our example, syntactic annotations are given in Fig. 5.4, respectively.

The tree structure that both XML and JSON build upon is very convenient to represent syntactic annotations, as illustrated in Fig. 5.4. The direct annotation of syntax trees with `@ana` requires the use of URIs (teidata.pointer) as a reference to a feature structure `<fs>` or interpretation `<interp>` element. References to an external terminology repository such as OLiA [4] would be syntactically valid as well.

In comparison with CoNLL, inline XML annotations permit to preserve the original whitespaces together with the original context of a word, and they provide a directly processable representation of nested structures. In addition, TEI supports standoff mechanisms to refer to markables, and thus allows to create directed graph structures between markables. In the upper part of Fig. 5.5, the `@xml:id` attribute introduces a unique identifier (tei.pointer), i.e. (the local name of) a URI within the current document.

TEI pointers can be used as source and target of interpreted (i.e. typed) links, e.g. for SRL annotation as in the lower part of Fig. 5.5. In this example, all elements are URIs, which basically allows to emulate RDF triples.[6] While TEI URI resolution is

[5]The original definition and various examples can be found under http://www.tei-c.org/release/doc/tei-p5-doc/de/html/AI.html#AILA, last accessed 09-07-2019.

[6]Note that, despite interest in Linked Open Data within the TEI, TEI/XML is not a suitable serialization of RDF in general: On the one hand, it is not sufficiently constrained, *several* different serializations of RDF triples in TEI have been suggested and no consensus about preferences among these has been achieved so far (for different approaches, see Sect. 13.3). On the other

```
<s>             <!-- declaration of TEI URIs with @xml:id -->
 <w xml:id="word-1">James </w>
 <w xml:id="word-2">Baker </w>
 <w xml:id="word-3">told </w>
 <w xml:id="word-4">reporters </w>
 <w xml:id="word-5">Friday</w>
 <w xml:id="word-6">: </w>
 ...
</s>

<linkGrp type="SRL-annotation"> <!-- standoff annotation -->
 <link source="#word-3" @ana="#ARG0" target="#word-1"/>
 <link source="#word-3" @ana="#ARG0" target="#word-2"/>
 <link source="#word-3" @ana="#rel" target="#word-3"/>
 <link source="#word-3" @ana="#ARG2" target="#word-4"/>
 <link source="#word-3" @ana="#ARGM-TMP" target="#word-5"/>
<!--... -->
</linkGrp>
```

Fig. 5.5 Semantic role annotation in TEI (standoff XML)

normally restricted to tei.pointers (i.e. XML elements that are defined with @xml.id within a TEI document), this also provides a suitable device to refer to LLOD URIs in general. In the following sections, we describe two mechanisms to address texts and other natural language entities by means of LLOD formalisms.

5.2 Annotating Web Resources

Documents in the web come in various forms, and, often, it is not possible to embed metadata and annotations directly into them, e.g. because the annotator is not the owner of the document, and distributing a local copy may be restricted. Standoff formalisms support the physical separation of annotated material and annotations. The Open Annotation community and their Web Annotation Data Model provide a RDF-based approach for standoff annotation of web documents, with JSON-LD as its designated serialization. The Web Annotation Data Model provides a flexible means to represent standoff annotations relative to any kind of document on the web. It is being applied to linguistic annotations, primarily in the biomedical domain, although prototypical adaptions in other domains have been described as well, e.g. for NLP [5] or Digital Humanities [6].

hand, the TEI constructions used to emulate RDF triples can also have different interpretations—as evident from uses of TEI pointer structures such as <relation>, <link> or <ptr> that pre-date their (ab)use to represent or refer to linked data.

5.2.1 Web Annotation (Open Annotation)

The Web Annotation Data Model [7] provides specifications for the RDF-based annotation of digital resources and the lossless exchange and (re-)usability of such annotations [7]. The Web Annotation Data Model has been developed by the Open Annotation W3C Community Group[7] with precursors in the Annotation Ontology[8] and the Open Annotation Model.[9] The Annotation Ontology [8] was an effort to create an open OWL-DL ontology for the annotation of scientific documents in the web, in particular from the biological domain and BioNLP. In order to bridge the gap between the available array of biomedical ontologies and the linguistic expression of the corresponding concepts in scientific publications, the Annotation Ontology was developed as an open, sharable data structure for integrating documents with terminology resources [8, 9]. It was subsequently aligned with the specifications of the Open Annotation Community project, finally leading to the formation of the W3C Open Annotation Community Group [10].

The Web Annotation data model and vocabulary have been published as W3C recommendations in 2017 [7, 11]. The aim of Web Annotation is to be applicable across different media formats, the most common use case being *"attaching a piece of text to a single web resource"* [7]. However, in a Semantic Web context, annotations can also include structured elements which may provide, for example, machine-readable representations for a particular textual label, e.g. by providing a link with an external ontology. Accordingly, the data model and the vocabulary have been extended to cover a broad band-width of use cases beyond a plain labelling mechanism. Instead, annotations are understood as structured objects. The Web Annotation Model provides fully reified representation of annotated elements and annotations assigned to it. The Web Annotation Data Model follows the following core principles [7]:

- Annotations form a *directed* graph: An annotation consists of a `Body` (the value of the annotation) that typically expresses information about a `Target` (the element which is annotated).
- Targets are *external web resources*: Whereas a `Body` may be embedded in the annotation, a `Target` may be independently dereferenced.
- Annotations form a *hyper*graph: An annotation can have 0 or more `Body` elements, and 1 or more `Target` elements.
- Annotations are *reified*: `Body`, `Target` and `Annotation` are distinct resources, so that they can be further specified with properties and relationships, e.g. a link with a `Motivation` resource that expresses the intent behind the creation of an annotation.

[7] https://www.w3.org/community/openannotation/, last accessed 09-07-2019.

[8] http://code.google.com/p/annotation-ontology/, last accessed 09-07-2019.

[9] http://www.openannotation.org, last accessed 09-07-2019.

Web Annotation is defined by three W3C recommendations:

- The Web Annotation Data Model (https://www.w3.org/TR/annotation-model/, last accessed 09-07-2019) defines the concept and the core vocabulary.
- The Web Annotation Vocabulary (https://www.w3.org/TR/annotation-vocab/, last accessed 09-07-2019) provides the set of RDF classes, predicates and named entities used by the Web Annotation Data Model.
- The Web Annotation Protocol (https://www.w3.org/TR/annotation-protocol/, last accessed 09-07-2019) defines the mechanisms for accessing, creating and managing annotations by means of RESTful web services, also including the recommendation for JSON-LD as serialization.

In addition to these, the Web Annotation Ontology is also provided in a machine-readable view under http://www.w3.org/ns/oa#, and this URL defines the `oa:` namespace prefix. The core data structure of the Web Annotation Data Model is `oa:Annotation` as illustrated in Fig. 5.6: Annotations are required to be declared as instances (`rdf:type`) of `oa:Annotation`. Furthermore, the presence of a `oa:hasTarget` property defining the relationship between annotation and the annotated element is necessary. The target can be an IRI or a selector, i.e. "[a] resource which describes the segment of interest in a representation of a Source resource, indicated with `oa:hasSelector` from the Specific Resource. This class is not used directly in the Annotation model, only its subclasses" [11].

A number of selectors for various source formats and addressing mechanisms are supported, including, for example, the `TextPositionSelector` that identifies text segments based on character offsets, the `TextQuoteSelector` that identifies text segments on grounds of their textual context, and the `XPathSelector` that uses XPaths to identify elements of an XML document. Selectors for other modalities also exist, e.g. the `DataPositionSelector` and the `SvgSelector`; it is thus possible to create annotations across different media

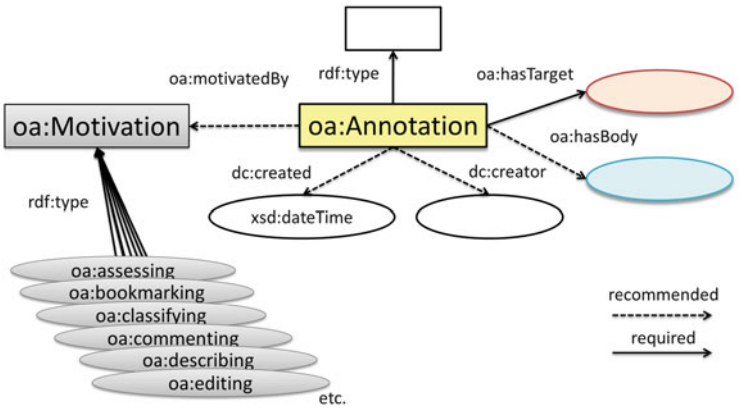

Fig. 5.6 Required and optional features of annotations in the Web Annotation Data Model [7]

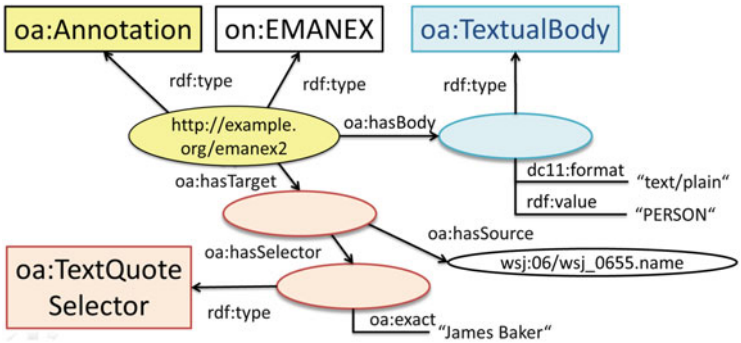

Fig. 5.7 Web Annotation example with named entity annotation, cf. Fig. 5.1

types, and using different reference strategies. Selectors are a highly generic and flexible way to refer to text passages in a text, however, also a comparably verbose one, so that for plain string references in plain text documents, users may want to consider using string URIs (see Sect. 5.3) instead of selectors. Where selectors for other data types exist, these should be preferred.

Additional predicates recommended for annotations are the metadata properties `dcterms:creator`, `dcterms:created` and `oa:motivatedBy`, but most importantly, the property `oa:hasBody`. The property `oa:hasBody` is an object property that specifies its object to be the body of the annotation. Different kinds of bodies are supported, e.g. a textual body, as illustrated in Fig. 5.7. As this representation is rather verbose, Web Annotation also provides the datatype property `oa:bodyValue`, which serves as a short-hand for the property path `oa:hasBody/rdf:value`.

5.2.2 *Annotating Named Entities on the Web*

The primary goal of language technology as well as linguistic research has been to analyse, to formalize and eventually to reproduce the function of language as a relation between form (grammar) and function (meaning). Web Annotation allows to formalize references to forms, e.g. linguistic expressions, but in addition, different target selectors also allow to perform a similar functionality across different modalities. In the context of linked data, the annotation of reference covers a particularly important aspect of meaning, as entities in texts refer to the same entity in the world, and are thus pivotal for creating links between texts and external knowledge bases as well as across texts. Named entities have thus long stood in the focus of interest in the Semantic Web and NLP communities. Their analysis in texts involves several aspects, most notably Entity Linking, where an entity mention in a text is assigned an identifier that represents this individual (say, a URI

from a knowledge base such as DBpedia), and Named Entity Recognition (NER), where entities are identified and classified for their type (say, general types such as organizations, persons, geopolitical entities, dates, or domain-specific concepts such as genes or drugs in BioNLP), as illustrated in example 2:

(2) Secretary of State James Baker, who accompanied President Bush to Costa Rica, told reporters Friday: "I have no reason to deny reports that some Contras ambushed some Sandinista soldiers. "

A Web Annotation representation of the CoNLL representation for the NER annotation of example 2 in Fig. 5.1 is given in Fig. 5.7. In comparison, Web Annotation is less compact, but it pursues a standoff approach, so that the primary data is left intact. Annotations are physically separated from the annotated document, with annotations preferably serialized in JSON-LD.

For the example, we employ the `oa:TextQuoteSelector` (see Table 5.1), which allows to describe a range of text by means of a literal match with the designated string, but also (optionally) some of the text immediately before (a prefix) and after (a suffix) in order to distinguish multiple copies of the same character sequence. As prefix and suffix are optional, text quote selectors are a very elegant solution to annotate all occurrences of a particular entity in a document with the same entity link. If different entities with the same surface string are to be distinguished (e.g. for pronouns), they can be disambiguated by context information.

Considering the first two named entities in the sentence only, this could be encoded in the Web Annotation fragment shown in Fig. 5.8 (in JSON-LD), resp. Fig. 5.9 (in Turtle).

Table 5.1 Characteristics of text quote selectors according to http://www.w3.org/ns/oa#TextQuoteSelector (accessed 09-07-2019)

Term	Type	Description
Type	Relationship	The class of the selector. Text quote selectors MUST have exactly 1 type and the value MUST be TextQuoteSelector
TextQuoteSelector	Class	The class for a selector that describes a textual segment by means of quoting it, plus passages before or after it. The TextQuoteSelector MUST have this class associated with it
Exact	Property	A copy of the text which is being selected, after normalization. Each TextQuoteSelector MUST have exactly 1 exact property
Prefix	Property	A snippet of text that occurs immediately before the text which is being selected. Each TextQuoteSelector SHOULD have exactly 1 prefix property, and MUST NOT have more than 1
Suffix	Property	The snippet of text that occurs immediately after the text which is being selected. Each TextQuoteSelector SHOULD have exactly 1 suffix property, and MUST NOT have more than 1

```
{
  "@graph": [
        {
          "@context": "http://www.w3.org/ns/anno.jsonld",
          "id": "http://example.org/enamex2",
          "type": [
             "Annotation",
                 "https://catalog.ldc.upenn.edu/docs/LDC2007T21/
                     ontonotes-1.0-documentation.pdf#ENAMEX"
          ],
          "body": {
                "type" : "TextualBody",
                "value" : "PERSON",
                "format" : "text/plain"
          },
          "target": {
                "source": "https://catalog.ldc.upenn.edu/
                    ldc2013t19/data/files/data/english/
                    annotations/nw/wsj/06/wsj_0655.name",
                "selector": {
                  "type": "TextQuoteSelector",
                  "exact": "James Baker"
                }  }
        }
] }
```

Fig. 5.8 Partial named entity annotation with Web Annotation and JSON-LD

```
<http://example.org/enamex2>
  a oa:Annotation, on:ENAMEX ;
  oa:hasBody [
    a oa:TextualBody ;
    dc11:format "text/plain"^^xsd:string ;
    rdf:value "PERSON"^^xsd:string
  ] ;
  oa:hasTarget [
    oa:hasSelector [
      a oa:TextQuoteSelector ;
      oa:exact "James Baker"^^xsd:string
    ] ;
    oa:hasSource wsj:06/wsj_0655.name
  ] .
```

Fig. 5.9 Partial named entity annotation with Web Annotation and Turtle

Similarly, alternative body values are possible, including references to external resources. Web Annotation thus provides an elegant mechanism to represent the output of Entity Linking systems. Using a service such as DBpedia Spotlight [12] (sample output in Fig. 5.10), the textual mention of *James Baker* can now be enriched with the URI `dbpedia:James_Baker` as annotation body. Note that an annotation can have multiple bodies (and/or targets), with the interpretation

```
<?xml version="1.0" encoding="utf-8"?>
<Annotation text="Secretary of State James Baker, ..." confidence
    ="0.35" support="0" types="" sparql="" policy="whitelist">
  <Resources>
    <Resource URI="http://dbpedia.org/resource/James_Baker"
        support="299" types="DBpedia:Agent,Schema:Person,Http://
        xmlns.com/foaf/0.1/Person,DBpedia:Person,DBpedia:
        OfficeHolder" surfaceForm="James Baker" offset="19"
        similarityScore="0.9999999912981821"
        percentageOfSecondRank="7.541871793467152E-9"/>
    ...
  </Resources>
</Annotation>
```

Fig. 5.10 Sample output of DBpedia Spotlight

```
{
  "@context": "http://www.w3.org/ns/anno.jsonld",
  "id": "http://example.org/enamex2",
  "type": [
    "Annotation",
        "Schema:Person",
        "http://xmlns.com/foaf/0.1/Person",
        "http://dbpedia.org/ontology/Agent",
        "http://dbpedia.org/ontology/Person",
        "http://dbpedia.org/ontology/OfficeHolder",
        "https://catalog.ldc.upenn.edu/docs/LDC2007T21/ontonotes
            -1.0-documentation.pdf#ENAMEX"
   ] ,
  "body": [
    "http://dbpedia.org/resource/James_Baker",
        {
      "type" : "TextualBody",
      "value" : "PERSON",
      "format" : "text/plain"
    } ] ,
  "target": {
    "source": "https://catalog.ldc.upenn.edu/ldc2013t19/data/
        files/data/english/annotations/nw/wsj/06/wsj_0655.name",
    "selector": {
      "type": "TextQuoteSelector",
      "exact": "James Baker"
    }  }
}
```

Fig. 5.11 Joint named entity and entity linking annotation with Web Annotation and JSON-LD

that every `oa:Body` is individually and equally related to the respective target(s). This mechanism can be applied to joint named entity and entity linking annotations (Figs. 5.11, resp. 5.12), but it should be noted that this integration of different body elements also means that their respective types are being conflated.

```
<http://example.org/enamex2>
  a oa:Annotation, on:ENAMEX,
  schema:Person, foaf:Person,
  dbo:Agent, dbo:Person, dbo:OfficeHolder;
  oa:hasBody <http://dbpedia.org/resource/James_Baker>, [
    a oa:TextualBody ;
    dc11:format "text/plain"^^xsd:string ;
    rdf:value "PERSON"^^xsd:string
  ] ;
  oa:hasTarget [
    oa:hasSelector [
      a oa:TextQuoteSelector ;
      oa:exact "James Baker"^^xsd:string
    ] ;
    oa:hasSource wsj:06/wsj_0655.name
  ] .
```

Fig. 5.12 Joint named entity and entity linking annotation with Web Annotation and Turtle

Likewise, it is now possible to extend this annotation to other modalities. For example, if James Baker is shown on a digitized and web-accessible photograph, an `oa:FragmentSelector` (providing the source image and the relevant coordinates) can be added to the annotation as yet another target.

5.3 Annotating Textual Objects

Whereas Web Annotation covers the full bandwidth of web resources, more specialized and less verbose formalisms for referencing strings and formalizing them as objects of linguistic annotation have been developed. Of particular importance in this context is the NLP Interchange Format [13, NIF]. Building on RFC 5147 specifications for URI Fragment Identifiers for the text/plain media type, NIF provides a URI scheme that allows to directly address strings in web-accessible documents, as well as ontologies formalizing strings and selected aspects of 'typical' annotations in NLP.

A key advantage in comparison to Web Annotation is a more compact representation of string references, whereas Web Annotation provides the verbose selector concepts, selector, reference and source document are identified in a compact fashion in a single URI. While Web Annotation focuses on formalizing annotations, NIF focuses on strings to which annotations may be assigned. RFC 5147 [14] defines an extension of earlier specifications for the text/plain MIME type, i.e. simple, unformatted text 'seen simply as a linear sequence of characters, possibly interrupted by line breaks or page breaks' [15, RFC 2046]. In general, URI fragment identifiers extend document URIs with a local name separated from the document URI using a hash sign (#).

RFC 5147 provides a simple offset mechanism to address strings, i.e. sequences of characters, in a web document as follows:

- **Position**: A character offset starting from the beginning of the document, defining an empty string at a particular position in the document. For the document https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt and using the offsets from Fig. 5.10 for *James Baker* from example 2, we arrive at the following position URI:
  https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt#char=19
  Note that the initial BOM character does not count, and that line endings (regardless of whether defined as LF, CR or LF+CR) count as one character.
- **Range:** A consecutive sequence of characters with a particular start position and a particular end position, both defined as character offsets:
  https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt#char=19,30
  If the first value of a range is not defined, it defaults to 0, if the second is not defined, it defaults to the end of the document.
- **Character Offsets:** Number of characters before the designated string, i.e. 0 for the first character. This is illustrated in the examples above.
- **Line Offsets:** Analogously to character offsets, a line offset refers to the number of lines (resp., line separators) before the designated position. The following example refers to the first line in the document:
  https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt#line=0
  With a range definition and underspecified end, the following URI refers to the *textual content* of the entire document (which can thus be distinguished from the document itself):
  https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt#line=0,

The text scheme is optionally followed by an integrity check, i.e. a length specification or an MD5 value:

```
...#char=19,30;length=12
...#char=19,30;md5=67f60186fe687bb898ab7faed17dd96a
```

Furthermore, a character encoding can be defined:

```
...#char=19,30;length=12,UTF-8
...#char=19,30;,UTF-8
```

Originally, RFC 5147 has been developed for highlighting strings in web documents. Aside from this application, its uses seem to be largely limited to language technology, where its URIs can be directly used as targets of web annotations and thus provide a compact alternative to Web Annotation selectors.

It should be noted, however, that RFC 5147 URIs are defined with reference to the text/plain MIME type, and that their application to other kinds of documents on

the web (in particular, documents with markup) is somewhat abusive.[10] Nevertheless, RFC 5147 represents the basis for *all* URI schemes for strings, including NIF [13], NAF [17] and LIF [18]—which are designed to address strings in character streams regardless of MIME type declarations.

Another potential issue of RFC 5147 URIs is that they may involve implicit information. Most importantly, string URIs are sensitive to the respective encoding. Defining the character encoding is optional, but without explicit declarations, RFC 5147 defaults to ASCII. However, with today's predominant use of UTF-8, this default can easily lead to unexpected results. Integrity checks help to detect possible errors, but their specification is optional. To explicate such information in RDF has been one motivation for developing the NLP Interchange Format.

5.3.1 The NLP Interchange Format (NIF 2.0)

The NLP Interchange Format (NIF) [19] is an RDF/OWL-based format designed to combine NLP tools in a flexible, light-weight fashion, originally developed in the FP7 LOD2 EU project (2010–2014). NIF aims to complement aggregator infrastructures for NLP such as UiMA [20] or GATE [21] with a representation that excels beyond either exchange format in terms of interoperability. UiMA, for example, builds on proprietary annotation type systems which are often maintained in-house, so that annotations between modules developed for different UiMA pipelines, e.g. by different providers, cannot be easily interchanged—or require writing new adapters, so that NLP modules from one pipeline can be integrated in the other. NIF provides a way to map the annotations of two or more NLP pipelines into a common representation and to integrate them seamlessly.

NIF includes the following core components:

- URI schemes to refer to strings in documents and to add annotations to such URIs
- An OWL-based vocabulary to express relations between String URIs
- Vocabulary extensions to represent frequent types of annotations in common NLP pipelines
- Best practices on how to integrate NLP tools, adapt them to NIF, and expose them as web services
- A reference implementation and a web demo for this functionality

NIF is a community standard developed at the Agile Knowledge Engineering and Semantic Web group at the University of Leipzig, Germany, with various external contributors. Albeit not being W3C-endorsed yet, it enjoys relatively wide adaptation for NLP services in the LLOD ecosystem, and has also been applied

[10] Indeed, other recommendations for this purpose have been developed, most notably XPointer [16]. In practical applications, however, XPointer seems to be largely unused.

to represent annotated corpora—although this is not its original focus and imposes limitations on the types of annotations that can be represented. NIF is well-suited for word-based annotations, e.g. for entity linking, but it is not capable of differentiating annotations of the same string on multiple annotation layers.

The core of NIF consists of a vocabulary for addressing arbitrary character sequences by RDF URIs to which linguistic annotations can be attached in a flexible fashion. By reference to a common pool of URIs, resp., by means of a mapping of annotated text data to a NIF representation, annotations from different NLP tools can be aggregated easily (if the same URI scheme is chosen).

For referencing strings, NIF provides two URI schemes [22], roughly corresponding in their function to text position selectors and text quote selectors in Web Annotation:

- **offset-based URIs** define strings by the character positions in the underlying document. They consist of four parts:

  1. the namespace (normally, the URI of the annotated document), followed by # or /,
  2. the scheme identifier `offset`, followed by _,
  3. start index, followed by _, and
  4. end index

  Indexes start with 0, and the underlying encoding is assumed to be Unicode Normal Form C [23, NFC]. Using the offsets from Fig. 5.10 for *James Baker* from example 2, we arrive at the following URI[11]:

  ```
  https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt
                                  #offset_19_30
  ```

  In Web Annotation, expressing the same information would require a TextPositionSelector and 5 triples.
- **context-hash URIs** identify strings on grounds of their forms and context. They have been introduced as a means to improve robustness against document changes, but they can also be applied to annotate multiple strings at the same time if these occur in the same contexts. They consist of six parts:

  1. the namespace, followed by # or /
  2. the scheme identifier `hash`, followed by _
  3. the context length, i.e. number of preceding and following characters considered, followed by _
  4. the length of the string, followed by _
  5. the message digest, a 32-character hexadecimal MD5 hash index generated from preceding context, the string (enclosed in parenthesis) and the following context, followed by _

[11] Note that we replaced the URI of the annotated file with the URI of the original text file in the Penn Treebank. As LDC corpora are available for download only, but not for online access, however, neither of these URIs resolve.

6. the first 20 characters of the string itself, in URL encoding

   For James Baker and context size 0, this yields the URI

```
https://catalog.ldc.upenn.edu/docs/LDC95T7/raw/06/wsj_0655.txt
  #hash_0_11_67f60186fe687bb898ab7faed17dd96a_James%20Baker
```

   The information provided by this URI corresponds to five Web Annotation triples in Fig. 5.9.

- **Other String URIs** supported by NIF include RFC 5147 strings, as well as consecutive string instantiations (CStringInst), another schema for offset- and context-based selection introduced in compliance with Apache Stanbol.[12]

In practice, NIF context hash URIs seem to be rarely used, but they are a powerful (albeit potentially dangerous) instrument to annotate all instances of a particular string simultaneously. A disadvantage is that for every context size, the string has a different URI and the relation between these can no longer be treated (nor easily recognized) as `owl:sameAs` because a URI generated from a long context may be unambiguous, but another URI with a short (or no) context that designates the same string may also identify another string in the same document. In combination with Web Annotation, the NIF URI scheme also provides an elegant alternative to the verbose system of selectors. Because web documents may change, either kind of URIs may resolve to an incorrect string, and to preserve interpretability, it is recommended to include the full text of the annotated document in the NIF RDF data. As shown below, string URIs allow us to elegantly mashup different annotations—a feature which is desirable for NLP pipelines, but which may be problematic for linguistic annotations in general.

In addition to string URIs, NIF allows to define relations between and contexts of strings. As shown in Fig. 5.13, the different URI schemes can be made explicit and their transformation can be tracked, and explicit offset information can be added. The (optional) property `nif:anchorOf` allows to provide the string value of the annotated element in RDF, a functionality required for querying NIF data. Furthermore, strings can be positioned relatively to each other (`before`, `after`), and string embedding can be expressed (`subString`, etc.)—a feature which can be subsequently used to model parse trees. As mentioned above, it is recommended to complement strings with an explicit representation of their context, i.e. the content of the full document (itself modelled as a `nif:String`).

Beyond text anchoring, NIF provides a core vocabulary for types of annotations specific for NLP pipelines as shown in Fig. 5.14, covering many practically relevant use cases. The NIF 2.0 Core Ontology [13] provides datastructures for the annotation of words and sentences, (hierarchical—i.e. `nif:subString`—and) sequential relations between these (`nif:nextWord`, `nif:nextSentence`), as well as concepts for groups of words, i.e. (syntactic) phrases, sentences, paragraphs

[12] https://stanbol.apache.org/, last access 09-07-2019.

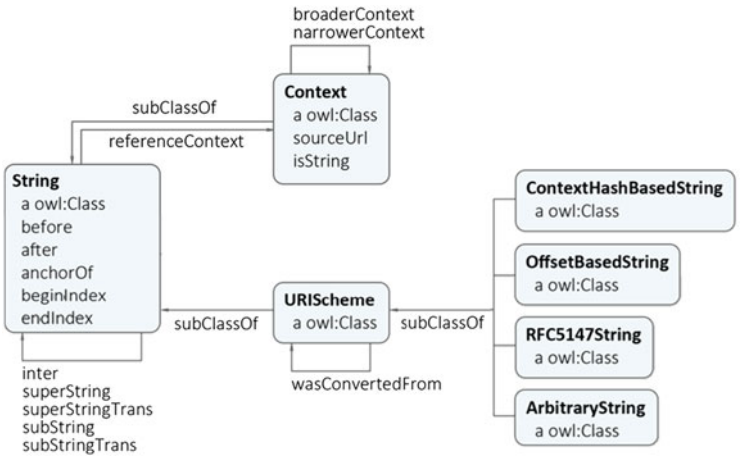

Fig. 5.13 NIF 2.0 Core Ontology, string classes and properties according to [13]

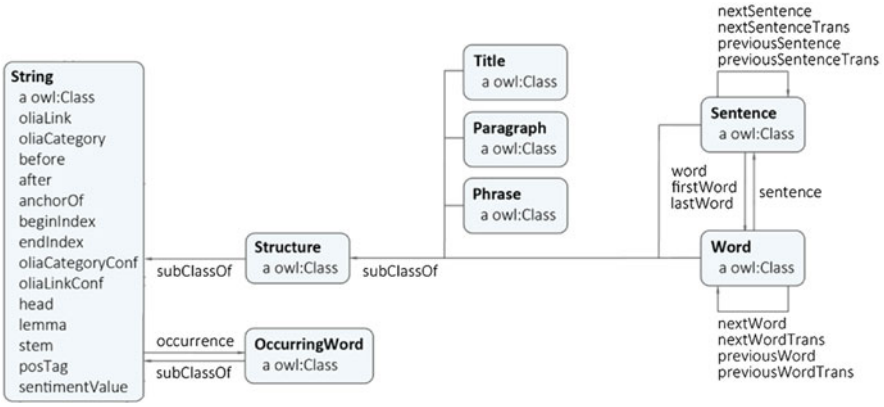

Fig. 5.14 NIF 2.0 Core Ontology, annotation classes and properties according to [13]

and titles. Despite obvious shortcomings, this rudimentary inventory accounts for many applications in NLP.

In addition to this, NIF has been extended for a number of use cases, and `nif:Strings` can thus be annotated with the corresponding properties for part-of-speech tagging (`nif:posTag`), morphological base forms (`nif:stem`, `nif:lemma`), sentiment (`nif:sentimentValue`) and (not shown in the diagram) syntactic dependencies (`nif:dependency`, `nif:dependency RelationType`). Except for `nif:dependency`, which points to the URI of the syntactic head, these are data type properties and provide literal values only. However, part-of-speech annotations, syntactic categories and dependency relations can be represented in a machine-readable and formal way by resolving tags to explicit links to the Ontologies of Linguistic Annotation (`nif:oliaCategory`).

This linking can be performed automatically if tags are compared against the string values or patterns defined in the corresponding OLiA Annotation Models. Further types of linguistic annotation are provided by external, community-maintained vocabularies, e.g. the NERD ontology for entity linking.[13]

The following RDF code represents a sample annotation for tokenization and sentence segmentation with NIF (included in other annotations):

```
PREFIX nif: <http://persistence.uni-leipzig.org/nlp2rdf/
    ontologies/nif-core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX doc: <https://catalog.ldc.upenn.edu/docs/LDC95T7/raw
    /06/wsj_0655.txt#>

doc:offset_0_188 a nif:Sentence, nif:Context , nif:
    OffsetBasedString ;
        nif:isString "Secretary of State James Baker, who
            accompanied President Bush ..." .

doc:offset_0_9 a nif:Word, nif:OffsetBasedString ;
        nif:anchorOf "Secretary" ;
        nif:beginIndex "0" ; nif:endIndex "9" ;
        nif:nextWord doc:offset_10_12 ;
        nif:sentence doc:offset_0_188 ;
        nif:referenceContext doc:offset_0_188 .

doc:offset_10_12 a nif:Word, nif:OffsetBasedString ;
        nif:anchorOf "of" ;
        nif:beginIndex "10" ; nif:endIndex "12" ;
        nif:nextWord doc:offset_13_18 ;
        nif:previousWord doc:offset_0_9 ;
        nif:sentence doc:offset_0_188 ;
        nif:referenceContext doc:offset_0_188 .
```

A data sample for part-of-speech annotations is provided below:

```
PREFIX nif: <http://persistence.uni-leipzig.org/nlp2rdf/
    ontologies/nif-core#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX doc: <https://catalog.ldc.upenn.edu/docs/LDC95T7/raw
    /06/wsj_0655.txt#>

doc:offset_0_9 nif:anchorOf "Secretary" ;       # included for
     readability, only
        nif:posTag "NNP" .

doc:offset_10_12 nif:anchorOf "of" ;
        nif:posTag "IN" .

doc:offset_13_18 nif:anchorOf "State";
        nif:posTag "NNP" .
```

[13] http://nerd.eurecom.fr/ontology, last accessed 09-07-2019.

```
13
14 doc:offset_19_24 nif:anchorOf "James";
15         nif:posTag "NNP" .
16
17 doc:offset_25_30 nif:anchorOf "Baker";
18         nif:posTag "NNP" .
19
20 doc:offset_30_31 nif:anchorOf ",";
21         nif:posTag "," .
22
23 doc:offset_32_35 nif:anchorOf "who";
24         nif:posTag "WP" .
25
26 doc:offset_36_47 nif:anchorOf "accompanied";
27         nif:posTag "VBD" .
28
29 doc:offset_48_57 nif:anchorOf "President";
30         nif:posTag "NNP" .
31
32 doc:offset_58_62 nif:anchorOf "Bush";
33         nif:posTag "NNP" .
```

Named entity categories can be represented analogously in NIF. NIF does not provide a designated property for the purpose, instead one can apply a property that points to the original documentation (using the same URI as in Web Annotation, i.e. `on:ENAMEX`).

```
1  PREFIX nif: <http://persistence.uni-leipzig.org/nlp2rdf/
       ontologies/nif-core#>
2  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3  PREFIX doc: <https://catalog.ldc.upenn.edu/docs/LDC95T7/raw
       /06/wsj_0655.txt#>
4  PREFIX on: <https://catalog.ldc.upenn.edu/docs/LDC2007T21/
       ontonotes-1.0-documentation.pdf#>
5
6  doc:offset_13_18                # context information skipped
7          nif:anchorOf "State";   # cf. POS example above
8          nif:beginIndex "13" ; nif:endIndex "18" ;
9          on:ENAMEX "ORG" .
10
11 doc:offset_19_30
12         nif:anchorOf "James Baker";
13         nif:beginIndex "19" ; nif:endIndex "30" ;
14         on:ENAMEX "PERSON" .
15
16 doc:offset_58_62
17         nif:anchorOf "Bush";
18         nif:beginIndex "58" ; nif:endIndex "62" ;
19         nif:posTag "PERSON" .
```

The first annotated `nif:Word` takes a URI that equals that of the POS annotation. Accordingly, both can be trivially merged.

```
doc:offset_13_18
    a nif:Word;                          # tokenization
        nif:beginIndex "13" ; nif:endIndex "18" ;
        nif:anchorOf "State";
        nif:nextWord doc:offset_19_24 ;
        nif:referenceContext doc:offset_0_188 ;
        nif:sentence doc:offset_0_188 ; # sentence splitting
        nif:posTag "NNP" ;               # POS annotation
        on:ENAMEX "ORG" .                # NER annotation
```

In the example, the IOBES annotation of the original CoNLL annotation (Fig. 5.1) has been expanded to full tokens.[14] However, this leads to a problem in that *James Baker* does not directly correspond to a URI in the POS annotation. Automated merging is thus limited to single-word expressions. With explicit `nif:beginIndex`, and `nif:endIndex`, it is possible to query for word spans containing each other.

Similarly, the entity linking annotation from Fig. 5.10 can be rendered in NIF as follows:

```
PREFIX itsrdf: <http://www.w3.org/2005/11/its/rdf#>

doc:offset_19_30 a nif:Word;
        nif:anchorOf "James Baker";
        nif:beginIndex "19" ; nif:endIndex "30" ;
        itsrdf:taIdentRef  <http://dbpedia.org/resource/
            James_Baker> ;
        a dbo:Agent, dbo:Person, dbo:OfficeHolder .

doc:offset_48_62 a nif:Word;
        nif:anchorOf "President Bush";
        nif:beginIndex "48" ; nif:endIndex "62" ;
        itsrdf:taIdentRef  <http://dbpedia.org/resource/
            George_W._Bush> ;
        a dbo:Agent, dbo:Person, dbo:OfficeHolder .
```

While `doc:offset_19_30` matches the URI generated by NER annotation, we have to observe another mismatch with multi-word expressions: Again, the URI `doc:offset_48_62` can only be indirectly related to the URI `doc:offset_58_62`.

In Natural Language Processing, different annotation tools can produce their annotations independently. If the same tokenization is applied, word URIs generated by different annotators become identical, such that the information from different annotators is seamlessly integrated. This is a particularly useful feature for word-level annotations. At those points where the tokenization differs, the anchoring in

[14] As an alternative to multi-word expressions, it is possible, of course, to operate with IOBES-based single-word annotations, thereby facilitating the merging process.

the same context as well as `nif:beginIndex` and `nif:endIndex` allows to infer overlaps and spans. Implicit unification of annotations as illustrated above requires the use of the same URI scheme. If annotators use different NIF URI schemes, an explicit conversion routine is to be applied. If the complete original text is provided as reference context and the URI scheme can be identified, NIF URIs are convertible. URI conversion, however, benefits from explicating offset and context information.

5.3.2 Provenance and Annotation Metadata in NIF

In a typical NIF workflow, a stream of textual data or a single document is consumed and transformed by a web service which returns NIF so that the result can be further enriched by further NIF annotation services. Instead, the output of different NIF annotators can be put into different RDF graphs which then represent the corresponding levels of annotation.

It is somewhat problematic, though, in ensemble combination architectures where different NLP modules generate annotations of the same kind in order to achieve a more robust annotation [24]. Similar problems may arise in multi-layer corpora, where the same document may be annotated for the same phenomenon in independent annotation efforts. In both cases, an explicit representation of documents and annotation layers within a document would be preferred.

Coming back to the example, the file `wsj_0655` has not only been annotated within OntoNotes, but also as part of the Penn Treebank [25] as well as various corpora that build on the Penn Treebank, cf. Sect. 6.3.2. For parts of speech and syntax annotation of most OntoNotes texts, we have at least three versions (with marginal differences in tokenization and annotation) of Penn Treebank annotations, as well as an independent annotation (according to the same scheme, but with adjustments in tokenization and certain design decisions) as part of OntoNotes.

Annotations may differ slightly, and without a versioning system, different annotators may generate different values for the same property, thereby leading to a clash of annotations. A NIF 2.1 solution to provenance is to define companion properties for properties that express linguistic annotations, e.g. `nif:taIdentConf` (confidence) and `nif:taIdentProv` (provenance) for the property `itsrdf:taIdentRef` (entity linking) [26]. However, different entity linking routines can produce alternative linkings, and there is no explicit association between a particular `itsrdf:taIdentRef` and a particular `nif:taIdentProv` property. In the NIF 2.1 draft, `nif:AnnotationUnits` have been introduced to cluster annotations of the same string generated from different annotators. However, this is described as a future extension, and this is not reflected in the persistent NIF documentation. If provenance, confidence or other metadata is to be provided, the interested reader may resort to Web Annotation instead, as their specifications are more stable and more mature than those of NIF 2.1.

For the moment, advanced challenges in NLP (architectures that implement parallel rather than sequential processing, i.e. blackboard or ensemble architectures) and corpus linguistics (redundantly annotated corpora) are beyond the scope of NIF and require a representation that disentangles strings and units of annotations and that provides an explicit organization of units of annotations in annotation layers or tiers. Both aspects will be addressed in Chap. 6.

5.4 Summary and Further Reading

We described two representative corpus formalisms currently used in language technology, resp., computational philology and corpus linguistics, and described how they can be complemented with URI references to LLOD resources and thereby establish bridges between state-of-the-art technology and resources on the one hand and the emerging field of linguistic linked data on the other hand. In CoNLL TSV formats, an additional column may be added that links every word to a URI; for TEI/XML, the current use of TEI-specific URIs can be easily extended to URIs in general.

Existing approaches to refer to textual (and non-textual) objects on the web in a linked-data-compliant fashion are based either on the use of target-specific selectors or compact string URIs:

- Web Annotation represents a promising and widely used approach to address textual and non-textual objects on the web by means of selectors, and by linking them with an annotation. If these annotations define a resolvable URI on their own, these URIs may be referred to from pre-RDF formalisms.
- RFC 5147 is a URI scheme that directly allows to address strings in a web document and represents a more compact alternative to Web Annotation selectors. It is the basis for the development of the NLP Interchange Format that extends the applicability of offset- and context-based URI schemes from plain text to web documents in general.

Note that these strategies and formalisms to refer to and to annotate textual objects on the web are not mutually exclusive. The NIF String ontology allows to describe information that underlies the URI formation process, thereby acting in analogy with Web Annotation selectors. Likewise, Web Annotation can refer to NIF or RFC 5147 URIs as targets of annotations as an alternative instead of declaring selectors.

Beyond merely referencing strings and other web objects as units of annotation, Web Annotation provides the expressive means to represent annotations on their own, in particular for annotations that can be reduced to labelling and identification. It is less clear how complex annotations for, say, syntactic dependencies, phrase structure syntax or semantic roles are to be represented in this context. NIF does provide explicit data structures for selected types of linguistic annotation frequently occurring in NLP pipelines, but it is not exhaustive in this regard.

At the time of writing, Web Annotation and NIF (resp., RFC 5147) are the most popular RDF-based formalisms to refer to natural language objects as units of annotation in the web. Far from being the only proposals, both are representative for JSON/LD and RDF/OWL-based approaches, respectively. In technical contexts, NIF enjoys considerable popularity, and its application to NLP pipelines is described in Chap. 11. Likewise, Web Annotation is an established community standard in BioNLP and Digital Humanities.

Independently from developments in language technology, URI-based methods to address text segments have been developed in computational philology: The Canonical Text Service (CTS, see Sect. 13.2.4) in the CITE architecture defines a URI (URN) scheme to address *canonical* units of texts. One goal of CTS is to facilitate intertextuality and stemmatology; these canonical units are thus *not* defined for an individual text, but rather for a family of texts or fragments of texts that originate from a common source, and thereby explicate corresponding passages. These efforts aim at defining intertextual reference points rather than units of annotation and are thus not directly comparable. In the context of language technology, alternative, application- or system-specific representation formalisms have been developed: For example, TELIX [27] used RDFa to infuse RDF content into an exchange format for an NLP pipeline, with the goal of linking it with lexical entries defined in SKOS XL [28].

In the NewsReader project,[15] the standoff format NAF was employed in NLP pipelines for entity and event extractions for Dutch, English and German [29]. NAF is an XML format that uses standoff mechanisms as described in Sect. 5.1.2, but an RDF conversion along the lines of NIF has been suggested [17].[16] NAF covers several types of NLP annotations relevant for event extraction and entity tracking, but only provides a vocabulary specifically oriented towards the NLP pipeline(s) it was originally designed for.

The LAPPS Interchange Format (LIF) was designed to integrate various NLP tools into the LAPPS Grid [30], a workflow system for multi-step analyses, evaluation tools and facilities for sharing and publishing results. LIF thus serves a similar purpose as NIF and NAF, but it adopts JSON-LD as RDF serialization. A LIF document consists of three sections: `metadata`, `text` and `views`. The metadata section contains optional metadata, the text section contains the text that is originally input to the service and the views section contains the annotations that have been added by the service to the text—together with an `id` and annotation-specific `metadata`. Similar to the `oa:TextPositionSelector` in Web Annotation, explicit attributes encode start and end positions of markables. We discuss LAPPS in more detail in Chap. 11.

Furthermore, a number of application- or tool-specific formats with their own URI schemes can be mentioned, e.g. the MATE parser [31],[17] a system for

[15] https://github.com/newsreader/NAF, accessed 09-07-2019.

[16] Also see http://wordpress.let.vupr.nl/naf/, accessed 09-07-2019.

[17] http://barbar.cs.lth.se:8081/, accessed 09-07-2019.

dependency parsing and semantic role labelling, the machine reading system FRED [32][18] or the LODeXporter [33],[19] a component for automatic knowledge base construction integrated in the GATE architecture [21]. In the longer perspective, and with continuing growth of the LLOD cloud and LLOD-aware applications, we expect an increasing degree of convergence in this area, probably based on formats and schemes already popular now.

Promising candidates are NIF and Web Annotation. But also these have been developed from an application perspective in a bottom-up fashion and thus require extensions for unforeseen applications, e.g. the annotation of morphology—currently neither addressed by the Web Annotation community nor by maintainers and users of NIF. Such limits of applicability of NIF and Web Annotation are not evident to most of their users, as they focus on frequently requested functionalities such as handling metadata about web objects, and the output of off-the-shelf NLP and Entity Linking pipelines, respectively. With the continuing growth of LLOD technology, we expect that increased exchange between different groups of users of LLOD technology will eventually lead to more expressive and more robust means to address and to annotate textual objects on the web as well as to the emergence of increasingly mature standards. At the moment, we recommend using Web Annotation for the conjoint handling of textual objects and non-textual objects in the web, and NIF/RFC 5147 for representing the output of NLP pipelines. For the future, we expect increased convergency between these and related representations.

Web Annotation and NIF thus aim to facilitate the transition from pre-RDF representation formalisms for linguistic annotation to LLOD-compliant representations. This aspect is further explored in the following chapter.

References

1. E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, R. Weischedel, OntoNotes: the 90% solution, in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2006)* (Association for Computational Linguistics, New York, 2006), pp. 57–60
2. J. Nivre, Ž. Agić, L. Ahrenberg, et. al., Universal dependencies 1.4 (2016). http://hdl.handle.net/11234/1-1827
3. N. Ide, C. Chiarcos, M. Stede, S. Cassidy, Designing annotation schemes: from model to representation, in *Handbook of Linguistic Annotation*, ed. by N. Ide, J. Pustejovsky, Text, Speech, and Language Technology (Springer, Berlin, 2017)
4. C. Chiarcos, Ontologies of linguistic annotation: survey and perspectives, in *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, Istanbul, 2012, pp. 303–310
5. K. Verspoor, K. Livingston, Towards adaptation of linguistic annotations to scholarly annotation formalisms on the Semantic Web, in *Proceedings of the 6th Linguistic Annotation Workshop* (Association for Computational Linguistics, Jeju, 2012), pp. 75–84

[18] http://wit.istc.cnr.it/stlab-tools/fred/, accessed 09-07-2019.

[19] https://github.com/SemanticSoftwareLab/TextMining-LODeXporter, accessed 09-07-2019.

6. L. Isaksen, R. Simon, E.T. Barker, P. de Soto Cañamares, Pelagios and the emerging graph of ancient world data, in *Proceedings of the 2014 ACM Conference on Web Science* (ACM, New York, 2014), pp. 197–201
7. R. Sanderson, P. Ciccarese, B. Young, Web Annotation Data Model. Technical Report, W3C Recommendation (2017). https://www.w3.org/TR/annotation-model/
8. P. Ciccarese, M. Ocana, L.J. Garcia Castro, S. Das, T. Clark, An open annotation ontology for science on web 3.0, J. Biomed. Semant. **2**(Suppl. 2), S4 (2011). https://doi.org/10.1186/2041-1480-2-S2-S4, http://www.jbiomedsem.com/content/2/S2/S4/abstract
9. D.C. Comeau, R. Islamaj Doğan, P. Ciccarese, K.B. Cohen, M. Krallinger, F. Leitner, Z. Lu, Y. Peng, F. Rinaldi, M. Torii, et al., BioC: a minimalist approach to interoperability for biomedical text processing, Database **2013**, bat064 (2013)
10. R. Sanderson, P. Ciccarese, H. Van de Sompel, Designing the W3C Open Annotation data model, in *Proceedings of the 5th Annual ACM Web Science Conference, WebSci '13* (ACM, New York, 2013), pp. 366–375. https://doi.org/10.1145/2464464.2464474
11. R. Sanderson, P. Ciccarese, B. Young, Web Annotation vocabulary. Technical Report, W3C Recommendation (2017). https://www.w3.org/TR/annotation-vocab/
12. P. Mendes, M. Jakob, A. García-Silva, C. Bizer, DBpedia Spotlight: shedding light on the web of documents, in *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics 2011)*, Graz, 2011
13. S. Hellmann, NIF 2.0 Core Ontology. Technical Report, AKSW, University Leipzig (2015). http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core.html, version of 08-04-2015. Accessed 9 July 2019
14. E. Wilde, M. Duerst, RFC 5147 – URI fragment identifiers for the text/plain media type. Technical Report, Internet Engineering Task Force (IETF), Network Working Group (2008)
15. N. Freed, N. Borenstein, RFC 2046 – Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. Technical Report, Internet Engineering Task Force (IETF), Network Working Group (1996)
16. P. Grosso, E. Maler, J. Marsh, N. Walsh, XPointer Framework. W3C Recommendation 25 March 2003. Technical Report, W3C (2003)
17. A. Fokkens, A. Soroa, Z. Beloki, N. Ockeloen, G. Rigau, W.R. van Hage, P. Vossen, NAF and GAF: Linking linguistic annotations, in *Proceedings of the 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation* (2014), pp. 9–16
18. N. Ide, K. Suderman, E. Nyberg, J. Pustejovsky, M. Verhagen, LAPPS/Galaxy: Current state and next steps, in *Proceedings of the 3rd International Workshop on Worldwide Language Service Infrastructure and 2nd Workshop on Open Infrastructures and Analysis Frameworks for Human Language Technologies (WLSI/OIAF4HLT2016)* (2016), pp. 11–18
19. S. Hellmann, J. Lehmann, S. Auer, M. Brümmer, Integrating NLP using Linked Data, in *Proceedings of the 12th International Semantic Web Conference, 21–25 October 2013*, Sydney, 2013. Also see http://persistence.uni-leipzig.org/nlp2rdf/
20. M. Egner, M. Lorch, E. Biddle, UIMA Grid: Distributed large-scale text analysis, in *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'07)*, Rio de Janeiro, 2007, pp. 317–326
21. H. Cunningham, GATE, a general architecture for text engineering. Comput. Hum. **36**(2), 223 (2002)
22. S. Hellmann, J. Lehmann, S. Auer, Linked-data aware URI schemes for referencing text fragments, in *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management* (Springer, Berlin, 2012), pp. 175–184
23. M. Davis, K. Whistler, Unicode Standard Annex #15. Unicode Normalization Forms. Technical Report, Unicode, Inc. (2017). Unicode 10.0.0, version of 2017-05-26, revision 45
24. E. Brill, J. Wu, Classifier combination for improved lexical disambiguation, in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, Montréal, 1998, pp. 191–195

25. M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of English: the Penn treebank. Comput. Linguist. **19**, 313 (1993)
26. S. Hellmann, M. Brümmer, M. Ackermann, Provenance and confidence for NIF annotations. Technical Report, AKSW, University of Leipzig, Germany (2016). Version of Oct 17, 2016
27. E. Rubiera, L. Polo, D. Berrueta, A. El Ghali, TELIX: An RDF-based model for linguistic annotation, in *Proceedings of the 9th Extended Semantic Web Conference (ESWC 2012)*, Heraklion, 2012
28. A. Miles, S. Bechhofer, SKOS Simple Knowledge Organization System eXtension for Labels (SKOS-XL). Technical Report, W3C Recommendation (2009)
29. R. Agerri, I. Aldabe, E. Laparra, G. Rigau Claramunt, A. Fokkens, P. Huijgen, R. Izquierdo Beviá, M. van Erp, P. Vossen, A.L. Minard, et al., Multilingual event detection using the NewsReader pipelines, in *Proceedings of the Workshop on Cross-Platform Text Mining and Natural Language Processing Interoperability, collocated with International Conference on Language Resources and Evaluation (LREC)* (2016)
30. M. Verhagen, K. Suderman, D. Wang, N. Ide, C. Shi, J. Wright, J. Pustejovsky, The LAPPS Interchange Format, in *Proceedings of the International Workshop on Worldwide Language Service Infrastructure* (Springer, Berlin, 2015), pp. 33–47
31. B. Bohnet, J. Kuhn, The best of both worlds: a graph-based completion model for transition-based parsers, in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (Association for Computational Linguistics, Stroudsburg, 2012), pp. 77–87
32. A. Gangemi, V. Presutti, D. Reforgiato Recupero, A.G. Nuzzolese, F. Draicchio, M. Mongiovì, Semantic Web machine reading with FRED Semantic Web **8**(6), 873 (2017)
33. R. Witte, B. Sateli, The LODeXporter: flexible generation of linked open data triples from NLP frameworks for automatic knowledge base construction, in *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)* (2018)