Linked-Data Aware URI Schemes for Referencing Text Fragments

Sebastian Hellmann[1], Jens Lehmann[1], and Sören Auer[2]

[1] Universität Leipzig, IFI/BIS/AKSW, D-04109 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org
[2] Technische Universität Chemnitz, Informatik/ISST, D-09107 Chemnitz, Germany
soeren.auer@informatik.tu-chemnitz.de

Abstract. The NLP Interchange Format (NIF) is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations. The motivation behind NIF is to allow NLP tools to exchange annotations about text documents in RDF. Hence, the main prerequisite is that parts of the documents (i.e. strings) are referenceable by URIs, so that they can be used as subjects in RDF statements. In this paper, we present two NIF URI schemes for different use cases and evaluate them experimentally by benchmarking the stability of both NIF URI schemes in a Web annotation scenario. Additionally, the schemes are compared with other available schemes used to address text with URIs. The String Ontology, which is the basis for NIF, fixes the referent (i.e. a string in a given text) of the URIs unambiguously for machines and thus enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform.

1 Introduction

We are currently observing a plethora of *Natural Language Processing* (NLP) tools and services being available and new ones appearing almost on a weekly basis. Some examples of web services providing just *Named Entity Recognition* (NER) services are *Zemanta*, *OpenCalais*, *Ontos*, *Evri*, *Extractiv*, *Alchemy*. Similarly, there are tools and services for language detection, Part-Of-Speech (POS) tagging, text classification, morphological analysis, relationship extraction, sentiment analysis and many other NLP tasks. Each of the tools and services has its particular strengths and weaknesses, but exploiting the strengths and synergistically combining different tools is currently an extremely cumbersome and time consuming task. The programming interfaces and result formats of the tools have to be analyzed and differ often to a great extent. Also, once a particular set of tools is integrated this integration is *not reusable* by others.

Additionally, the use of LOD background knowledge in NLP applications poses some particular challenges. These include: *identification* – uniquely identifying and reusing identifiers for (parts of) text, entities, relationships, NLP

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 175–184, 2012.
© Springer-Verlag Berlin Heidelberg 2012

concepts and annotations etc.; *provenance* – tracking the lineage of text and annotations across tools, domains and applications; *semantic alignment* – tackle the semantic heterogeneity of background knowledge as well as concepts used by different NLP tools and tasks.

In order to simplify the combination of tools, improve their interoperability and facilitate the use of Linked Data we developed the NLP Interchange Format (NIF). NIF is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. The NIF specification has been released in an initial version 1.0 in November 2011[1] and implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist and a public web demo[2] is available. NIF addresses the interoperability problem on three layers: the *structural*, *conceptual* and *access* layer. NIF is based on a Linked Data enabled URI scheme for identifying elements in (hyper-)texts that are described by a String Ontology (structural layer) and a selection of ontologies for describing common NLP terms and concepts (conceptual layer). NIF-aware applications will produce output (and possibly also consume input) adhering to the NIF URI Scheme and the String Ontology as REST services (access layer). Other than more centralized solutions such as UIMA[3] and GATE[4], NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. Another benefit is that a NIF wrapper has to be only created once for a particular tool, but enables the tool to interoperate with a potentially large number of other tools without additional adaptations. Ultimately, we envision an ecosystem of NLP tools and services to emerge using NIF for exchanging and integrating rich annotations.

We present the NIF URI Schemes including an experimental evaluation in Section 2. The usage of identifiers in the String Ontology is discussed in Section 3. We review related work in Section 4 and conclude with an outlook on future work in Section 5.

2 NIF URI Schemes

The motivation behind NIF is to allow NLP tools to exchange annotations about documents in RDF. Hence, the main prerequisite is that parts of the documents (i.e. strings) are referenceable by URIs, so that they can be used as subjects in RDF statements. We call an algorithm to create such identifiers *URI Scheme*: For a given text t (a sequence of characters) of length $|t|$ (number of characters), we are looking for a *URI Scheme* to create a URI, that can serve as a *unique* identifier for a substring s of t (i.e. $|s| \leq |t|$). Such a substring can (1) consist of adjacent characters only and it is therefore a unique character sequence within the text, if we account for parameters such as context and position or (2) derived by a function which points to several substrings as defined in (1).

[1] `http://nlp2rdf.org/nif-1-0/`
[2] `http://nlp2rdf.lod2.eu/demo.php`
[3] `http://uima.apache.org/`
[4] `http://gate.ac.uk/`

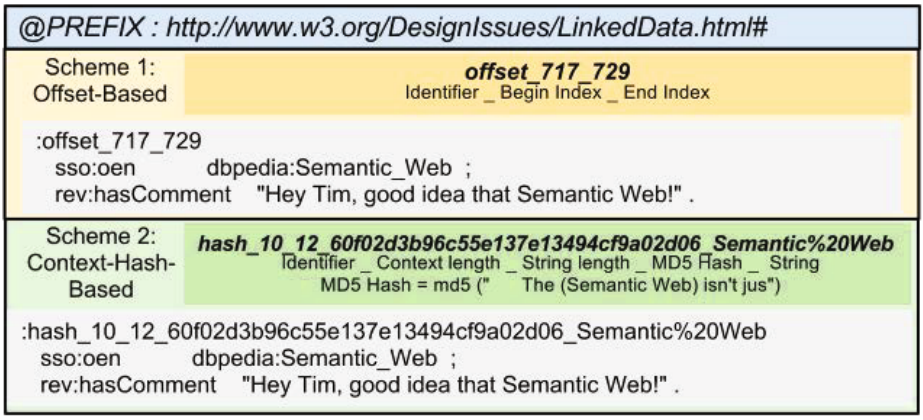

Fig. 1. NIF URI schemes: Offset (top) and context-hashes (bottom) are used to create identifiers for strings, see Section 3 for `sso:oen`

NIF provides two URI schemes, which can be used to represent strings as RDF resources. In this section, we focus on the first scheme using offsets. In the top part of Figure 1, two triples are given that use the following URI as subject: `http://www.w3.org/DesignIssues/LinkedData.html#offset_717_729`

According to the above definition, the URI points to a substring of a given text t, which starts at index 717 until the index 729.

For the URI creation scheme, there are three basic requirements – *uniqueness*, *ease of implementation* and *URI stability* during document changes. Since these three conflicting requirements can not be easily addressed by a single URI creation scheme, NIF defines two URI schemes, which can be chosen depending on which requirement is more important in a certain usage scenario. Naturally further schemes for more specific use cases can be developed easily. After discussing some guidelines on the selection of URI namespaces, we explain in this section how stable URIs can be minted for parts of documents by using *offset-based* and *context-hash* based schemes (see Figure 1 for examples).

Namespace Prefixes. A NIF URI is constructed from a namespace prefix and the actual *identifier* (e.g. "offset_717_729"). Depending on the selected context, different prefixes can be chosen. For practical reasons, it is recommended that the following guidelines should be met for NIF URIs: If we want to annotate a (web) resources, the whole content of the document is considered as `str:Context`, as explained in the next section, and it is straightforward to use the existing document URL as the basis for the prefix. The prefix should then either end with slash ('/') or hash ('#')[5].

[5] Note that with '/' the identifier is sent to the server during a request (e.g. Linked Data), while everything after '#' can only be processed by the client.

Recommended prefixes for `http://www.w3.org/DesignIssues/LinkedData.html` are:
– `http://www.w3.org/DesignIssues/LinkedData.html/`
– `http://www.w3.org/DesignIssues/LinkedData.html#`

Offset-Based URIs. The offset-based URI scheme focuses on ease of implementation and is compatible with the position and range definition of RFC 5147[6] (esp. Section 2.1.1) and builds upon it in terms of encoding and counting character positions (See Section 4 for a discussion). Offset-based URIs are constructed of three parts separated by an underscore '_': (1) a *scheme identifier*, in this case the string 'offset', (2) *start index*, (3) the *end index*. The indexes are counting the gaps between the characters starting from 0 as specified in RFC 5147 with the exception that the encoding is defined to be Unicode Normal Form C (NFC)[6] and counting is fixed on Unicode Code Units[7]. This scheme is easy and efficient to implement and the addressed string can be referenced unambiguously. Due to its dependency on start and end indexes, however, a substantial disadvantage of offset-based URIs is the *instability* with regard to changes in the document. In case of a document change (i.e. insertion or deletion of characters), all offset-based URIs after the position the change occurred become invalid.

Context-Hash-Based URIs. As an alternative to the offset-based scheme, context-hash-based URIs are designed to remain more robust regarding document changes. Context-hash-based URIs are constructed from five parts separated by an underscore '_':

1. a *scheme identifier*, in this case the string 'hash',
2. the *context length* (number of characters to the left and right used in the message for the hash-digest),
3. the *overall length* of the addressed string,
4. the *message digest*, a 32-character hexadecimal MD5 hash created from the string and the context. The message M consists of a certain number C of characters (see 2. context length above) to the left of the string, a bracket '(', the string itself, another bracket ')' and C characters to the right of the string: 'leftContext(String)rightContext'. If there are not enough characters to left or right, C is adjusted and decreased on the corresponding side (see the 'Hurra!' example below).
5. the *string itself*, the first 20 (or less, if the string is shorter) characters of the addressed string, urlencoded.

The additional brackets '(' and ')' around the string were introduced to make the identifier more uniquely distinguishable. If there is a sentence 'Hurra! Hurra!' and the context size is too large, e.g. 10, then the first and the second 'Hurra!' would have the same hash. By adding brackets, however, the hash is easily distinguishable: `md5("(Hurra! Hurra!)")` != `md5("(Hurra!) Hurra!")` != `md5("Hurra! (Hurra!)")`.

[6] `http://www.unicode.org/reports/tr15/#Norm_Forms`
[7] `http://unicode.org/faq/char_combmark.html#7`

Note that context-hash-based URIs are unique identifiers of a specific string only if the context size is chosen sufficiently large. If, for example, a complete sentence is repeated in the document, parts of the preceding and/or subsequent sentences are to be included to make the reference to the string unique. However, in many NLP applications, a unique reference to a specific string is not necessary, but rather, all word forms within the same minimal context (e.g., one preceding and one following word) are required to be analysed in the same way. Then, a context-hash-based URI refers uniquely to words in the same context, not one specific string. Using a small context, one can refer to a whole class of words rather than just an individual one. For example, by using the string ' the ' (with one preceding and following white space as context) we obtain the digest: md5(' (the) '). The resulting URI is `http://www.w3.org/DesignIssues/LinkedData.html#hash_1_5_8dc0d6c8afa469c52ac4981011b3f582_%20the%20` and would denote all occurrences of 'the' in the given reference context, surrounded by a single white space on both sides.

Trivially, every string is uniquely addressable if the context-length is large enough. The algorithm for finding the addressed strings in a given text is simple: 1. URL decode the fifth part (the string itself) and search for all occurrences and get the start indices. 2. From all found start indices generate the hash by calculating the end index (start index + overall length), adding brackets and including the context (if start index − context length < 0, then left context starts at index 0, right context starts at end index and does not go beyond end of text). The following algorithm computes the minimal context-length (MinCl) on a fixed document with a given set of annotations, so that each URI only denotes one substring.

```
 1: procedure MinCl(annotations, cl)
 2:     uris ← {}
 3:     for all annotations a do
 4:         uri ← makeUri(a)
 5:         if uris contains uri then
 6:             return MinCl (annotations, cl +1 )
 7:         else
 8:             uris ← uris ∪ uri
 9:         end if
10:         return cl
11:     end for
12: end procedure
```

URI Stability Evaluation. As the context-hash-based URI scheme differs significantly in terms of uniqueness and stability from the offset-based scheme, we evaluate both schemes with real revision histories from Wikipedia articles. Although Wikipedia pages are edited quite frequently (≈ 202,000 edits per day[8]), the senses of each page tend to remain relatively stable after a certain number of revisions [2].

[8] `http://www.wikistatistics.net/wiki/en/edits/365`

Table 1. Evaluation of URI stability with different context length versus the offset scheme. The second last column measures how many annotations remain valid over 100 edits on Wikipedia.

tok ≈ 7410.7	Unique	URatio	Stability	1...100	Stab 1...100
context 1	2830.2	0.3988	0.3946	2647.3	0.3680
context 5	7060.0	0.9548	0.9454	6417.7	0.8551
context 10	7311.4	0.9871	0.9771	6548.8	0.8712
context 20	7380.6	0.9963	0.9854	6429.1	0.8553
context 40	7402.2	0.9990	0.9866	6146.8	0.8183
context 80	7408.8	0.9998	0.9847	5678.6	0.7568
offset	7410.7	1.00	0.5425	104.4	0.0164

We downloaded a Wikipedia dump with the full edit revision history[9]. From this dump, we randomly selected 100 articles which had more than 500 edits total. We retrieved the last 100 revisions of these 100 articles and removed the wiki markup[10]. Then we split the resulting plain text into tokens at word level. We used a deterministic algorithm (mostly based on regular expressions) for the markup removal and the tokenisation to avoid any side effects. The text for each revision contained 57062.4 characters on average, which we split into 7410.7 tokens on average (around 7.7 chars/token). About 47.64 characters were added between each revision. For each token and each revision, we generated one URI for the offset scheme and six URIs for the context-based scheme with context length 1, 5, 10, 20, 40 and 80. Cf. Section 4 for details why other approaches were not included in this evaluation. For every same URI that was generated within one revision i for two different tokens (a violation of the uniqueness property), the uniqueness ratio (*URatio*) decreases: $\frac{|UniqueURIs_i|}{|Tokens_i|}$. The stability was calculated by the intersection of UniqueURIs of two revisions (i and *i+1*) over the number of tokens of the second revision: $\frac{|UniqueURIs_i \cap UniqueURIs_{i+1}|}{|Tokens_i+1|}$. Thus non-unique URIs were penalized for the calculation of stability (without this penalty the percentage was always about 99%). We did the same measurement between the first and the last revision (columns *1...100* and *Stab 1...100*) of each article. The results are summarized in Table 1.

While a high context length (cl=80) provides more stability between revisions (99.98%), cl = 10 yields 87.12% of the URIs valid over 100 edits. The offset-based URIs have a probability of 54% to become invalid between revisions. This corresponds roughly to the theoretically probability for a random insertion to break a URI: $\frac{a-1}{n+1} + \frac{n-a+2}{2n+2} = \frac{a+n}{2n+2}$ (n = text length, a = annotation length). For context-hash URIs: $\frac{a+2cl-1}{n+1}$.

3 Usage of Identifiers in the String Ontology

We are able to fix the referent of NIF URIs in the following manner: To avoid ambiguity, NIF requires that the whole string of the document has to be included

[9] `http://dumps.wikimedia.org/enwiki/20111007/`

[10] Code from `http://www.mediawiki.org/wiki/Extension:ActiveAbstract`

in the RDF output as an `rdf:Literal` to serve as the reference point, which we will call *inside context* formalized using an OWL class called `str:Context`[11]. By typing NIF URIs as `str:Context` we are refering to the content only, i.e. an arbitrary grouping of characters forming a unit. The term *document* would be inappropriate to capture the real intention of this concept as `str:Context` could also be applied to a *paragraph* or a *sentence* and is **absolutely independent** upon the *wider context* in which the string is actually used such as a Web document reachable via HTTP.

We will distinguish between the notion of outside and inside context of a piece of text. The *inside context* is easy to explain and formalise, as it is the text itself and therefore it provides a *reference context* for each substring contained in the text (i.e. the characters before or after the substring). The *outside context* is more vague and is given by an outside observer, who might arbitrarily interpret the text as a "book chapter" or a "book section".

The class `str:Context` now provides a clear reference point for all other relative URIs used in this context and blocks the addition of information from a larger (outside) context. `str:Context` is therefore disjoint with `foaf:Document`, because labeling a context resource as a document is an information, which is not contained within the context (i.e. the text) itself. It is legal, however, to say that the string of the context occurs in (`str:occursIn`) a `foaf:Document`. Additionally, `str:Context` is a subclass of `str:String` and therefore its instances denote textual strings as well.

```
@prefix : <http://www.w3.org/DesignIssues/LinkedData.html#> .
@prefix str: <http://nlp2rdf.lod2.eu/schema/string/> .
:offset_0_26546  a  str:Context ;
#the exact retrieval method is left underspecified
  str:occursIn <http://www.w3.org/DesignIssues/LinkedData.html> ;
# [...] are all 26547 characters as rdf:Literal
    str:isString    "[...]" .
:offset_717_729  a  str:String ;
    str:referenceContext :offset_0_26546 .
```

As mentioned in Section 2, NIF URIs are grounded on Unicode Characters using Unicode Normalization Form C counted in Code Units. For all resources of type `str:String`, the universe of discourse will then be the **words over the alphabet of Unicode characters** (sometimes called Σ^*). According to the "*RDF Semantics W3C Recommendation*", such an interpretation is considered a "semantic extension"[12] of RDF, because "extra semantic conditions" are "imposed on the meanings of terms"[13]. This "semantic extension" allows – per definitionem – for an unambiguous interpretation of NIF by machines. In particular, the `str:isString` term points to the string that fixes the referent of the context. The meaning of a `str:Context` NIF URI is then exactly the string contained in the object of `str:isString`. Note that Notation 3 even permits literals as subjects of statements, a feature, which might even be adopted to RDF[14].

[11] For the resolution of prefixes, we refer the reader to `http://prefix.cc`

[12] `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#urisandlit`

[13] `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#intro`

[14] `http://lists.w3.org/Archives/Public/www-rdf-comments/2002JanMar/0127.html`

Table 2. Comparison of URI schemes (first two are used in NIF)

	Uniq	Val	XML	Trans	Addr	Self	Impl	Exp	Example
Context-Hash(NIF)	+	+	+	+	+	+	o	o	#hash_10_12_60f0...
Offset(NIF)	++	++	+	- - -	++	+	++	o	#offset_717-729
Offset plain	++	++	-	- - -	++	-	++	o	#717-729
Yee (Context)	+	- -	+	+	- -	- -	- -	o	#:words:The-(Semantic We...
RFC 5147 [6]	++	++	+	- - -	++	++	+	+	#char=717-12
LiveURL (Content)	- -	+	-	+	+	-	++	o	#8Semantic12+0x206A73ED
LiveURL (Position)	+	+	-	- -	+	-	-	o	not available for text
Wilde et al. (Regex)	o	++	+	+	+	+	- -	++	#matching=`Semantic\sWeb`

Conceptual Interoperability via Ontologies. The *Structured Sentence Ontology* (SSO)[15] is built upon the String Ontology and provides additional classes for three basic units: *sentences*, *phrases* and *words*. Conceptual interoperability is ensured in NIF by providing ontologies and vocabularies for representing the actual annotations in RDF. For each NLP domain a pre-existing vocabulary was chosen that serves the most common use cases and facilitates interoperability. Details are described elsewhere: *Part-Of-Speech tags and Syntax* uses the Ontologies of Linguistic Annotation (OLiA, [1]); *Entity Linking* is realized using NERD [4], note that the property `sso:oen` – meaning 'one entity per name' – is explained and formalized there.

4 Related Work

As the suitability of the string identifiers highly depends on the specific task, we present in the following a list of criteria, which allow to evaluate and design suitable identifiers:

Uniqueness. The URIs must uniquely identify the substring. **Validity.** The URI scheme must produce valid URIs for arbitrary substrings. Valid URIs must not contain invalid characters and must be limited in length, since most browsers limit the size of the URIs, they can handle[16]. **XML Compatibility.** The identifier part for the generated URIs should be usable as an XML tag name (for RDF/XML serialisation). For example, XML tag elements can not begin with a number, thus prohibiting tags such as `<717-729>`. **Stability.** The URI should only become invalid if the referenced string is changed significantly, thus rightfully rendering the annotations void. It should not become invalid through unrelated changes. **Addressability.** The URIs can efficiently find the annotated substring within the text, i.e. calculate the start and end index (ideally rule based). **Self-Description.** Some URI schemes require certain parameters to find the appropriate substring in the document. The URIs should contain encoded information that can be used to identify the scheme itself and that can be used to reproduce the *configuration* of the scheme. As correct implementations are necessary to allow the creation of tool chains, it is beneficial, if the

[15] `http://nlp2rdf.lod2.eu/schema/sso/`

[16] MS Internet Explorer has a maximum URL length of 2,083 characters. `http://support.microsoft.com/kb/q208427/`

scheme has a low complexity to avoid **implementation** errors. **Expressivity.** This criteria measure how expressive the function is that references the strings (e.g. regex is more expressive than just start/end index).

Table 2 shows a comparison of various URI schemes. **LiveURLs** [3][17] is realized as a Firefox plugin and offers two different ways to produce string identifiers: a context-based and a position based. The user can select a text in the browser and then the plugin creates the URL pointing to the corresponding fragment. This URL can be shared and the referenced string is highlighted. As the identifier starts with a number, it can create a conflict with XML serialisation. Furthermore, the identifier does not contain enough information to uniquely distinguish duplicates, i.e. it would match several occurrences. The position based method uses a combination of the parent node's id and index in the DOM tree alongside an identifier for the child position. The position based method is content-specific and works only on XHTML. Analogous to all position based methods, the scheme is highly susceptible to change. **Wilde and Dürst** [6] filed an RFC in April 2008[18] proposing a parameter-like syntax using fragments that refer to statistics about the characters in the string (e.g. offsets, line, length), e.g. `ftp://example.com/text.txt#line=10,20;length=9876,UTF-8`. The basic properties of this scheme are a super set to the offset-based NIF scheme and the `owl:sameAs` relation holds: `:offset_717_729 owl:sameAs :char=717,729.` The *line* parameter will be considered for further benchmarks, but lacks the necessary granularity. The spec of the RFC restricts this scheme to the "plain text" media type, which excludes XML and HTML. Furthermore the scheme contains many optional parameters for integrity checking. When used as RDF subjects, it is tedious to resolve such optional parts, as `#line=10,20` is neither syntactically the same URI as `#line=10,20;length=9876`, nor can we automatically infer an `owl:sameAs` relation. **Yee** [7] proposed *Text-Search Fragment Identifiers*, which pinpoint the wanted substring with a fragment that includes the string and its context. Before the creation of the fragment identifier, however, the original HTML source is manipulated and all HTML tags are removed and special characters are normalized. The resulting URL for our example is: `#:words:The-(Semantic Web)-isnt-just-about-putting`. The problem is that the proposed normalization (i.e. remove HTML and tokenise context) can not be standardized easily as it relies on difficult to normalize NLP methods. Therefore, there is no guarantee to reproduce the manipulation bi-directionally (e.g. to find the annotated substring). Longer selected substrings lead to longer, invalid URIs. **Wilde and Baschnagel** [5] propose to use regular expression patterns following the parameter "matching" as fragment identifiers, i.e. `matching=Semantic\sWeb` would match all nine occurrences of "Semantic Web" at once. Although being powerful, it is not straight-forward to implement an algorithm that produces regular expressions addressing the correct strings in a text and thus results in high implementation complexity and unpredictability

[17] `http://liveurls.mozdev.org`
[18] `http://tools.ietf.org/html/rfc5147`

regarding uniqueness. Considering the possibility to include the context in an URI, this scheme is a superset of the previous approach by Yee.

5 Conclusions and Future Work

In this paper, we presented the URI schemes and relevant parts of the String Ontology, which underlie the NLP Interchange Format for integrating NLP applications. NIF addresses weaknesses of centralized integration approaches by defining an ontology-based and linked-data aware text annotation scheme. We argued that the URI schemes used in NIF have advantageous properties when compared with other approaches. This comparison is based on an extensive qualitative comparison as well as an experimental evaluation benchmark, which can be easily reproduced and extended for other scenarios. Especially, the context-hash based URIs look promising to provide a solution for web-scale annotation exchange. Future work comprises the creation of a new version NIF 2.0 with community feedback[19] as well as interoperability with (1) XML-based schemes, e.g. XPointer, (2) Media Fragments and the almost completed Provenance AQ .

Acknowledgments. We would like to thank our colleagues from AKSW research group and the LOD2 project for their helpful comments during the development of NIF. Especially, we would like to thank Christian Chiarcos for his support while using OLiA. This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Chiarcos, C.: Ontologies of linguistic annotation: Survey and perspectives. In: LREC. European Language Resources Association (2012)
2. Hepp, M., Siorpaes, K., Bachlechner, D.: Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. IEEE Internet Computing 11(5), 54–65 (2007)
3. Kannan, N., Hussain, T.: Live urls: breathing life into urls. In: 15th Int. Conf. on World Wide Web, WWW 2006, pp. 879–880. ACM, New York (2006)
4. Rizzo, G., Troncy, R., Hellmann, S., Bruemmer, M.: NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In: LDOW (2012)
5. Wilde, E., Baschnagel, M.: Fragment identifiers for plain text files. In: ACM HYPERTEXT 2005, pp. 211–213. ACM, New York (2005)
6. Wilde, E., Duerst, M.: URI Fragment Identifiers for the text/plain Media Type (2008), `http://tools.ietf.org/html/rfc5147` (Online; accessed April 13, 2011)
7. Yee, K.: Text-Search Fragment Identifiers (1998), `http://zesty.ca/crit/draft-yee-url-textsearch-00.txt` (Online; accessed April 13, 2011)

[19] `http://nlp2rdf.org/get-involved`