Towards an Ontology for Representing Strings

Sebastian Hellmann, Jens Lehmann, Sören Auer

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany,
{hellmann|lehmann|auer}@informatik.uni-leipzig.de
http://aksw.org

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. The String Ontology, which is the basis for NIF, fixes the referent (i.e. a string in a given text) of annotations unambiguously for machines and thus enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. We evaluate the String Ontology based on the adequacy of the collected requirements and furthermore by benchmarking the stability of the NIF URI scheme in a Web annotation scenario.

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 extend. Also, once a particular set of tools is integrated this integration is *not reusable* by others.

We argue that simplifying the interoperability of different NLP tools performing similar but also complementary tasks will facilitate the comparability of results, the building of sophisticated NLP applications as well as the synergistic combination of tools and might ultimately yield a boost in precision and recall for common NLP tasks. Some first evidence in that direction is provided

by tools such as RDFaCE[1], Spotlight [9] and Fox[2], which already combine the output from several backend services and achieve superior results.

Another important factor for improving the quality of NLP tools is the availability of large quantities of qualitative background knowledge on the currently emerging Web of Linked Data [1]. Many NLP tasks can greatly benefit from making use of this wealth of knowledge being available on the Web in structured form as *Linked Open Data* (LOD). The precision and recall of Named Entity Recognition, for example, can be boosted when using background knowledge from DBpedia, Geonames or other LOD sources as crowdsourced and community-reviewed and timely-updated gazetteers. Of course the use of gazetteers is a common practice in NLP. However, before the arrival of large amounts of Linked Open Data their creation and maintenance in particular for multi-domain NLP applications was often impractical.

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 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 facilitating 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[3] and implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist and a public web demo[4] 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 String Ontology as REST services (access layer). Other than more centralized solutions such as UIMA [5] and GATE [6], 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

[1] http://rdface.aksw.org/lite/test/tinymce/examples/rdface_lite.html
[2] http://aksw.org/Projects/FOX
[3] http://nlp2rdf.org/nif-1-0/
[4] http://nlp2rdf.lod2.eu/demo.php
[5] http://uima.apache.org/
[6] http://gate.ac.uk/

NLP tools and services to emerge using NIF for exchanging and integrating rich annotations.

The creation of NIF can be viewed in the context of similar activities taking place in other communities. Currently a trend to define and use semantic interchange formats can be observed. This includes in particular the *XML Localisation Interchange File Format* (XLIFF) – standardized by OASIS – and the Rule Interchange Format (RIF) [8] – standardized by W3C – or the family of Clinical Data Interchange Standards.

In this paper, we are focusing on the structural layer of NIF – the URI Schemes and the String Ontology:

- We designed an extensible framework for URI schemes and derive properties to evaluate such schemes. Two URI schemes are used in NIF and exploited for (hyper−) text annotations. These offset and context-hash based schemes combine a number of advantages from existing schemes and address their weaknesses.
- We developed a String Ontology that models strings and their context, thus denoting words over Unicode characters and providing a basis for unambiguous exchange of annotations between applications.
- We performed an evaluation of the stability of NLP annotation URIs using the Wikipedia corpus and revision history.
- Additional operations on NIF models are defined, which are adequate to fulfill the given requirements.
- This work is a substantial extension and revision of [11]. No further scientific (peer-reviewed) publications exist.

After the collected requirements, which guided the development of NIF and the ontology, we present the core concepts of String Ontology in Section 2, including an experimental evaluation of the NIF URI schemes (Section 2.1). We give a formal description on operations on NIF models in Section 3. We review related work in Section 4 and conclude with an outlook on future work in Section 5.

1.1 Requirements for NIF

An important aspect for the adequate development and evaluation of an OWL ontology are requirements, it seeks to fulfill. In this section, we will give a list of requirements, we elicitated within the LOD2 EU project[7], which influenced the design of NIF and the String ontology. The main software developed in LOD2 is the *LOD2 stack*, which integrates a wide range of RDF tools, including a Virtuoso triple store as well as Linked Data interlinking and OWL enrichment tools.

1. Compatibility with RDF.One of the main requirements, driving the development of NIF, was the need to convert any NLP tool output to RDF as virtually all software developed within the LOD2 Project is based on RDF and the underlying triple store.

[7] `http://lod2.eu`

2. Coverage. The wide range of potential NLP tools requires that the produced format and ontology is sufficiently general to cover all annotations and be able to convert them to RDF.
3. Structural Interoperability. NLP tools with a NIF wrapper should produce unanimous output, which allows to merge annotations from different tools consistently. Here structural interoperability refers to the way **how** annotations are represented.
4. Conceptual Interoperability. In addition to structural interoperability, tools are furthermore supposed to use the same vocabularies for the same kind of annotations. The focus lies on **what** annotations are used.
5. Granularity. The ontology is supposed to handle different kind of granularities not limited to the document level, which can be considered very coarse-grained. As basic units we identified a document collection, the document, the paragraph and the sentence. A keyword search for example might rank a document higher, where the keywords appear in the same paragraph.
6. (Robust) Web annotation. The format should be able to use textual Web content (i.e. blogs) as input and provide tobust standoff annotations for dynamic web data.
7. Provenance. In addition to the required robustness, it was also required to track provenance of the primary data.
8. Simplicity. As in later stages, third parties are supposed to contribute their NLP tools to the LOD2 Stack the format should be as simple as possible to ease integration.
9. Scalability. An especially important requirement is imposed on the format with regard to scalability in two dimensions: Firstly, the triple count is required to be as low as possible to reduce the memory and index footprint in all applications and triple stores. Secondly, the complexity of OWL axioms should ideally not exceed EL++[2] to allow fast reasoning.

Out of scope Alongside the requirement elicitation, we identified several items which are definitely out of scope and will (or can) not be included in NIF. Some tools provide additional information such as confidence about the annotations they produce. As we are required to stay within RDF (see Req. 1) such information can normally only be added using OWL2 axiom annotations or named graphs, which come with their own disadvantages (i.e. tremendous increase of triple count). We accept the limits of RDF as presupposed. Regarding the tracking of provenance, we laid our focus on simplicity and granularity. Due to the wide variety of provenance vocabularies, we are waiting for input from the W3C Provenance working group[8]. This part is currently left underspecified by NIF.

2 Core Concepts of the String Ontology

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

[8] `http://www.w3.org/2011/01/prov-wg-charter.html`

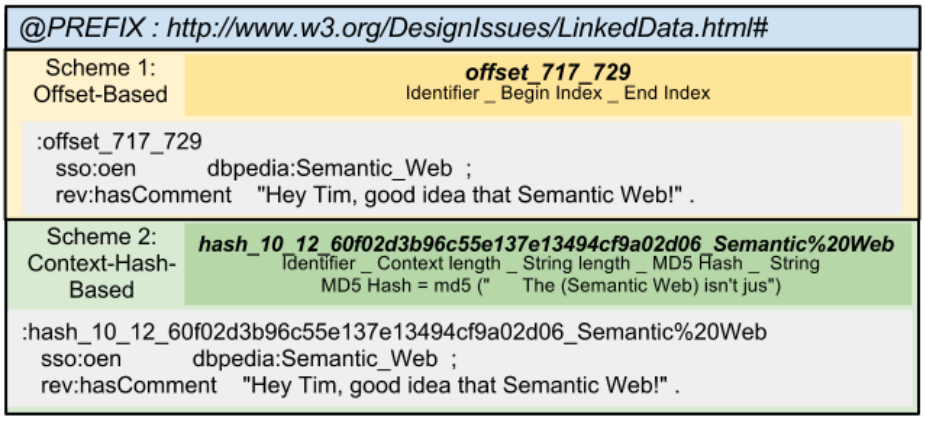

Fig. 1. NIF URI schemes: Offset (top) and context-hashes (bottom) are used to create identifiers for strings, see Section 2.2 for `sso:oen`

(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).

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. Section 2.1 explains both schemes in more detail. 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 character index 717 until the index 729 (counting all characters). To avoid ambiguity, NIF requires that the whole string of the document has to be included 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`[9]. The term *document* would be inappropriate to capture the real intention of this concept as we would like to refer to an arbitrary grouping of characters forming a unit, which could also be applied to a *paragraph* or a *sentence* and is highly dependent upon the *wider context* in which the string is actually used such as a Web document reachable via HTTP.

To appropriately capture the intention of such a class, 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

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

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. By definition `str:Context` is 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.

```
: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 .
```

NIF URIs are grounded on Unicode Characters using Unicode Normalization Form C [10]. 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, such an interpretation is considered a "semantic extension"[11] of RDF, as "extra semantic conditions" are "imposed on the meanings of terms"[12]. Perspectively, this will allow for an unambiguous interpretation of NIF by machines. Especially, the property `str:isString` fixes the referent of the context. The meaning of a `str:Context` NIF URI is then exactly the string contained in the object. Note that Notation 3 even permits literals as subjects of statements, a feature, which might even be adopted to RDF[13]. For further discussion see Section 3.

2.1 NIF URI Schemes

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).

[10] `http://www.unicode.org/reports/tr15/#Norm_Forms` counted in Code Units `http://unicode.org/faq/char_combmark.html#7`

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

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

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

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` 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 ('#')[14].

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[13] (esp. Section 2.1.1) and builds upon it in terms of encoding and counting character positions (See Section 4.2 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 and counting is fixed on Unicode Code Points.

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).

[14] 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.

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!)")`.

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 (end index - context length, should not be smaller 0, right context analogous). 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 ($\approx$ 202,000 edits per day[15]), the senses of each page tend to remain relatively stable after a certain number of revisions [5].

We downloaded a Wikipedia dump with the full edit revision history[16]. 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[17]. 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 4.2 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 not 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: $\frac{a+2cl-1}{n+1}$.

2.2 Conceptual Interoperability via Ontologies

The *Structured Sentence Ontology* (SSO)[18] 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

[15] `http://www.wikistatistics.net/wiki/en/edits/365`
[16] `http://dumps.wikimedia.org/enwiki/20111007/`
[17] code from `http://www.mediawiki.org/wiki/Extension:ActiveAbstract`
[18] `http://nlp2rdf.lod2.eu/schema/sso/`

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

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.

cases and facilitates interoperability. Details are described elsewhere: *Part-Of-Speech tags and Syntax* uses the Ontologies of Linguistic Annotation (OLiA, [3]); *Entity Linking* is realized using NERD [11], note that the property `sso:oen` – meaning 'one entity per name' – is explained and formalized there.

3 Operations on NIF models

Removing Ambiguity vs. Triple Count The String Ontology is designed with a **one-triple-per-annotation** paradigm to achieve scalability and reduce the triple count. Due to the clearly defined Semantics, it is possible to omit most of the triples by establishing a convention in the form of expansion rules(e.g. in RIF [8]) based on the class `str:StringInContext` ($StringInContext \sqsubseteq String$; $StringInContext \sqcap String \equiv \bot$)

```
@PREFIX : http://www.w3.org/DesignIssues/LinkedData.html#
:offset_717_729  a  str:StringInContext .
```

implies these 15 triples, if they are not already existing:

```
:offset_0_26546  a  str:String, str:Context ;
    str:isString   "[...]" .
    str:occursIn <http://www.w3.org/DesignIssues/LinkedData.html> ;
    owl:differentFrom <http://www.w3.org/DesignIssues/LinkedData.html> ;
:offset_717_729  a  str:String, str:StringInContext ;
    str:referenceContext     :offset_0_26546 ;
    str:superString          :offset_0_26546 ;
    str:superStringTrans     :offset_0_26546 ;
    str:beginIndex           "717"^^xsd:long   ;
    str:endIndex             "729"^^xsd:long   ;
    str:anchorOf             "Semantic␣Web"  ;
# next and previous relations to other Strings:
    str:nextString           :offset_730_732 ;
    str:nextStringTrans      :offset_730_732 ;
# other context-based NIF URIs are possible
    owl:sameAs :hash_10_12_60f02d3b96c55e137e13494cf9a02d06_Semantic%20Web .
```

These additional triples have three main functions: 1. The properties `str:isString`, `str:referenceContext` and `str:occursIn` remove ambiguity. Especially, the `str:isString` includes the string, that fixes the meaning of all NIF URIs (see Section 2). 2. The properties `str:beginIndex`, `str:endIndex` and `str:anchorOf` provide explicit information for SPARQL queries (e.g. to query, where in a text

occurs "Linked" within 40 characters of "Semantic"). 3. next, previous, sub and superString can be used to iterate the data. Note that within the framework of RDF and the current usage of NIF for the interchange of output between NLP tools, the definition of the semantics is sufficient to produce a working system. However, problems arise if additional interoperability with Web architecture, Linked Data or fragment identifiers and ad-hoc retrieval of content from the Web is demanded. The actual retrieval method (such as content negotiation) to retrieve and validate the content for `#offset_0_26546` is left underspecified by the property `str:occursIn`. This is often not a real problem and might be specified in the future.

Merge and Context Change A merge of two NIF models with the same context happens naturally, if the two models use the same prefix (normally the URL of the document or file). The merge is then merely the combination of the triples of both graphs. A change in context is more difficult and requires to rewrite all indices and verify all annotations. Part-Of-Speech tags (e.g. `olia:ProperNoun`) generally stay valid, if the whole sentence is included in the new context. Named entities need to be verified. Note the change below from Berlin, Germany to a Berlin in the US (last line).

```
:offset_0_23  a     str:Context , sso:Sentence;
    str:isString   "Berlin␣has␣a␣new␣mayor!" .
:offset_0_6   a     str:StringInContext, sso:Word ;
    str:referenceContext      :offset_0_23 ;
    sso:oliaLink   <http://purl.org/olia/penn.owl#ProperNoun> ;
    sso:oen                   dbpedia:Berlin .
#####->Context is changed->#######
:offset_0_68  a     str:Context ;
  str:isString  "Berlin␣has␣a␣new␣mayor!␣She's␣the␣first␣female␣mayor␣in␣Conneticut." .
:offset_0_6   a     str:StringInContext, sso:Word ;
    str:referenceContext      :offset_0_68 ;
    sso:oliaLink    <http://purl.org/olia/penn.owl#ProperNoun> ;
    sso:oen                   dbpedia:Berlin,_Connecticut .
```

Check consistency The NIF model can be checked for consistency with the help of an OWL reasoner (after e.g. a merge of two NIF models). The required OWL axioms are not contained within the standard String Ontology (which is mainly used for subClassOf axioms and documentation), because most of the applications using NIF do not require such consistency checks and the necessary functional object properties are not in EL++[19]. The axioms are kept in separate files and can be loaded on demand. An excerpt of relevant axioms in OWL functional-style syntax are shown here:

```
#Step 1: expand isString, referenceContext, beginIndex,
#         endIndex, anchorOf as described above
#Step 2: infer owl:sameAs statements keys:
HasKey(Context ( ) (isString))
HasKey StringInContext (referenceContext ) (beginIndex, endIndex, anchorOf))
#Step 3: impose the following restrictions:
FunctionalDataProperty ( isString, beginIndex, endIndex, anchorOf)
FunctionalObjectProperty ( referenceContext )
DisjointAxioms with virtually most of the other classes (e.g. foaf:Document)
```

[19] `http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/#OWL_2_EL`

The consistency check does not only reveal errors in the structure of the RDF, but it is also useful to detect encoding errors and even document changes, e.g. if two individuals are owl:sameAs, but have two different values in `str:anchorOf`.

4 Related Work

4.1 Other Annotations Formats

Annotation Ontology (AO, [4]) is closest to our work as they present an 'open ontology in OWL-DL for annotating scientific documents on the web'. AO provides structural mechanisms to annotate arbitrary electronic artifacts (including images). Due to its generic approach it can be considered a heavy-weight annotation framework as it needs 7 triples (cf. [4, Figure 4]) to explicitly express the same information encoded in one NIF URI. As NLP has special requirements regarding scalability, NIF has been designed with the lowest possible amount of overhead and only optional reasoning. AO can easily be transformed to NIF and vice versa. Even more important, their ontological modeling of provenance and document versioning can be integrated seamlessly into NIF. Another aspect that distinguishes both approaches is that Ciccarese et al. are focusing on the annotation scheme only and the AO is domain agnostic. NIF provides best practices as well as a community infrastructure to agree on common domain annotations and reference ontologies for interoperability.

In [6], the *Graph Annotation Framework (GrAF)* was used to bridge the models of UIMA and GATE. GrAF is the XML serialization of the Linguistic Annotation Framework (LAF). The authors describe several workarounds to represent links between entities (due to the limits of XML, additional NameMap and mapping descriptor files are required). One strength of NIF is the ability to integrate RDF data and reuse and link existing ontologies for data integration, which is easily possible as it is based on the RDF and OWL standards.

Extremely Annotational RDF Markup (EARMARK, [10] is a stand-off format to annotate text with markup and represent the markup in RDF. EARMARK focuses on representing (XML) markup in RDF and is capable to express overlapping annotations. The main method to address content is via ranges that are similar to the offset URI scheme. Other inline annotation formats for text are *RDFa* and *Microdata*. A major difference is that those annotations are intended to be directly integrated in the document by the owner, whereas NIF allows third parties to add annotations on existing documents. Moreover, NIF can also be used to annotate overlapping strings in text: In the sentence 'Time flies like an arrow; fruit flies like a banana.', an NLP tool could annotate "fruit flies" as noun phrase (NP) while another one annotates "flies like a banana" as verb phrase (VP). Integrating such overlapping annotations is not possible with RDFa.

4.2 URI Scheme Characterization and Comparison

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:

	Uniq	Val	XML	Trans	Addr	Self	Impl	Exp	Example
Context-Hash(NIF)	+	+	+	+	+	+	o	o	#hash_md5_4_12_abeb...
Offset(NIF)	++	++	+	- - -	++	+	++	o	#offset_14406-14418_Semantic+Web
Offset plain	++	++	-	- - -	++	-	++	o	#14406-14418
Yee (Context)	+	- -	+	+	- -	- -	- -	o	#:words:we-dont-call-it-(Semantic We...
RFC 5147 [13]	++	++	+	- - -	++	++	+	+	#char=14406-12
LiveURL (Content)	- -	+	-	+	+	-	++	o	#8Semantic12+0x206A73ED
LiveURL (Position)	+	+	-	- -	+	-	-	o	not available for text
Wilde et al. (Regex)	o	++	+	+	+	+	- -	++	#matching=`Semantic\sWeb`

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

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[20].

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 `<14406-14418>`.

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 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. Both NIF schemes address shortcomings of other methods. All URI schemes, which are applicable to text documents, are also applicable to HTML, XML, source code, CSS etc. With its addressing scheme identifier, NIF is extensible and further schemes (e.g. more content-specific URI schemes such as XPath/XPointer for XML) can be easily included in the future. The relation of NIF URIs to other fragment approaches might be formalized with `owl:sameAs`.

LiveURLs [7] [21] is realized as a Firefox plugin and offers two different ways to produce string identifiers: a content-based and a position based. Both can be appended to the URL as a fragment identifier. The user can select a text in the browser and then the plugin creates the URL with the corresponding

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

[21] `http://liveurls.mozdev.org`

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. As all position based methods it is highly susceptible to change. *Wilde and Dürst* [13] filed an RFC in April 2008[22] 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. 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 exclude XML and HTML. Furthermore the scheme contains many optional parameters for integrity checking, which renders it unsuitable for use as subjects in RDF (`#line=10,20` is not automatically the `owl:sameAs` of `#line=10,20;length=9876`. *Yee* [14] 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:we-dont-call-it- (Semantic Web).-The-second-rule,-to-use`. The problem is that the proposed normalization (i.e. remove HTML and tokenise context) can not be standardized easily as it relies on NLP methods, which are difficult to normalize. 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* [12] 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, this recipe is very powerful, it is not straight-forward to implement an algorithm that produces regular expressions that address the correct strings in a text. Considering that it is possible to include the context in such a URI, this recipe is a superset of the previous approach by Yee. Disadvantages are the unpredictability respective uniqueness and high implementation complexity of URI generation.

5 Conclusions and Future Work

In this paper, we presented the URI schemes and the String Ontology, which underly 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

[22] `http://tools.ietf.org/html/rfc5147`

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. Furthermore, we have given concrete requirements collected within the LOD2 project and the NLP2RDF community[23] to measure and justify the adequacy of the presented ontologies as well as the accompanying guidelines and code of practice. If possible an OWL formalisation was provided (RIF rules are future work).

Acknowledgments

We would like to thank our colleagues from AKSW research group and the LOD2 project for their helpful comments and inspiring discussions 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. S. Auer and J. Lehmann. Making the web a data washing machine - creating knowledge out of interlinked data. *Semantic Web Journal*, 2010.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope further. In K. Clark and P. F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.
3. C. Chiarcos. Ontologies of linguistic annotation: Survey and perspectives. In *LREC*. European Language Resources Association, 2012.
4. P. Ciccarese, M. Ocana, L. Garcia Castro, S. Das, and T. Clark. An open annotation ontology for science on web 3.0. *Journal of Biomedical Semantics*, 2(Suppl 2):S4+, 2011.
5. M. Hepp, K. Siorpaes, and D. Bachlechner. Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. *IEEE Internet Computing*, 11(5):54–65, 2007.
6. N. Ide and K. Suderman. Bridging the Gaps: Interoperability for Language Engineering Architectures using GrAF. *Language Resources and Evaluation*, 46(1):75–89, 2012.
7. N. Kannan and T. Hussain. Live urls: breathing life into urls. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 879–880, New York, NY, USA, 2006. ACM.
8. M. Kifer. Rule interchange format: The framework. In *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems*, RR '08, pages 1–11, Berlin, Heidelberg, 2008. Springer-Verlag.
9. P. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *I-Semantics*, 2011.

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

10. S. Peroni and F. Vitali. Annotations with earmark for arbitrary, overlapping and out-of order markup. In U. M. Borghoff and B. Chidlovskii, editors, *ACM Symposium on Document Engineering*, pages 171–180. ACM, 2009.
11. G. Rizzo, R. Troncy, S. Hellmann, and M. Bruemmer. NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In *LDOW*, 2012.
12. E. Wilde and M. Baschnagel. Fragment identifiers for plain text files. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '05, pages 211–213, New York, NY, USA, 2005. ACM.
13. E. Wilde and M. Duerst. URI Fragment Identifiers for the text/plain Media Type. `http://tools.ietf.org/html/rfc5147`, 2008. [Online; accessed 13-April-2011].
14. K. Yee. Text-Search Fragment Identifiers. `http://zesty.ca/crit/draft-yee-url-textsearch-00.txt`, 1998. [Online; accessed 13-April-2011].