NIF: An ontology-based and linked-data-aware NLP Interchange Format

Sebastian Hellmann
Universität Leipzig, IFI/AKSW
hellmann@informatik.uni-leipzig.de

Jens Lehmann
Universität Leipzig, IFI/AKSW
lehmann@informatik.uni-leipzig.de

Sören Auer
Universität Leipzig, IFI/AKSW
auer@uni-leipzig.de

ABSTRACT

We are currently observing a plethora of Natural Language Processing tools and services being made available. 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. 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 and the creation of sophisticated NLP applications. In addition, the synergistic combination of tools might ultimately yield a boost in precision and recall for common NLP tasks. In this paper, we present the NLP Interchange Format (NIF). NIF is based on a Linked Data enabled URI scheme for identifying elements in (hyper-)texts and an ontology for describing common NLP terms and concepts. NIF aware applications will produce output (and possibly also consume input) adhering to the NIF ontology. Other than more centralized solutions such as UIMA and GATE, NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. We evaluate the NIF approach by (1) benchmarking the stability of the NIF URI scheme and (2) providing results of a field study, where we integrated 6 diverse NLP tools using NIF wrappers.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Languages, Standardization

Keywords

Ontologies, NLP, Linked Data, RDF

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 [10], Spotlight [12] and Fox [1], 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 annota-

[1] `http://aksw.org/Projects/FOX`

tions 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 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 (structural layer) and a comprehensive ontology for describing common NLP terms and concepts (conceptual layer). NIF-aware applications will produce output (and possibly also consume input) adhering to the NIF ontology as REST services (access layer). Other than more centralized solutions such as UIMA [5] 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.

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) standardised by OASIS and the Rule Interchange Format (RIF) [11], standardized by W3C or the family of Clinical Data Interchange Standards.

With NIF we make in particular the following contributions:

- 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 comprehensive upper-level ontology for linguistic concepts integrating various existing ontologies for different NLP domains.
- We performed an evaluation of (1) the stability of NLP annotation URIs using the Wikipedia corpus and revision history, (2) the usability of NIF with a user study with six real-world NLP application development projects.

This article is structured as follows: We present the concepts underlying NIF in Section 2. We discuss the structural interoperability via URI schemes in Section 3, the conceptual interoperability using the NIF ontology in Section 4 and the access interoperability in Section 5. We evaluate the URI stability in Section 6. Our implementation including available NIF wrappers for existing tools and services and the NLP2RDF platform is presented in Section 7. We review related work in Section 8 and conclude with an outlook on future work in Section 9.

2. NIF OVERVIEW

The *NLP Interchange Format* (NIF)[2] is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. An overview of the general NIF architecture is shown in Figure 1. The core of NIF consists of a vocabulary, which allows to represent strings as RDF resources. A special URI design is used to pinpoint annotations to a part of a document. These URIs can then be used to attach arbitrary annotations to the respective character sequence. Employing these URIs, annotations can be published on the Web as Linked Data and interchanged between different NLP tools and applications. NIF consists of the following three components:

1. *Structural Interoperability*: URI schemes are used to anchor annotations in documents with the help of fragment identifiers. The URI schemes are complemented by two ontologies (String Ontology and Structured Sentence Ontology), which are used to describe the basic types of these URIs (i.e. String, Document, Word, Sentence) as well as the relations between them (subString, superString, nextWord, previousWord, etc.).
2. *Conceptual Interoperability*: The Structured Sentence Ontology (SSO) was especially developed to connect existing ontologies with the String Ontology and thus attach common annotations to the text fragment URIs. The NIF ontology can easily be extended and integrates several NLP ontologies such as *OLiA* for the morpho-syntactical NLP domains, the *SCMS Vocabulary*[3] and *DBpedia* for entity linking, as well as the NERD Ontology[4].
3. *Access Interoperability*: A REST interface description for NIF components and web services allows NLP tools to interact on a programmatic level.

In the next sections, we describe these NIF components in more detail.

3. STRUCTURAL INTEROPERABILITY VIA URI SCHEMES

The fundamental rationale of 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 they can be used as subjects in RDF statements. We call an algorithm to create such identifiers URI SCHEME. The annotation task can be described as follows: 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. $|s| \leq |t|$). We generally assume that this substring consists of adjacent characters only and is therefore a unique character sequence within the document, if we account for parameters such as context and position. 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 1.0 defines

[2]Specification 1.0: `http://nlp2rdf.org/nif-1.0`
[3]`http://scms.eu`
[4]`http://nerd.eurecom.fr/ontology/`

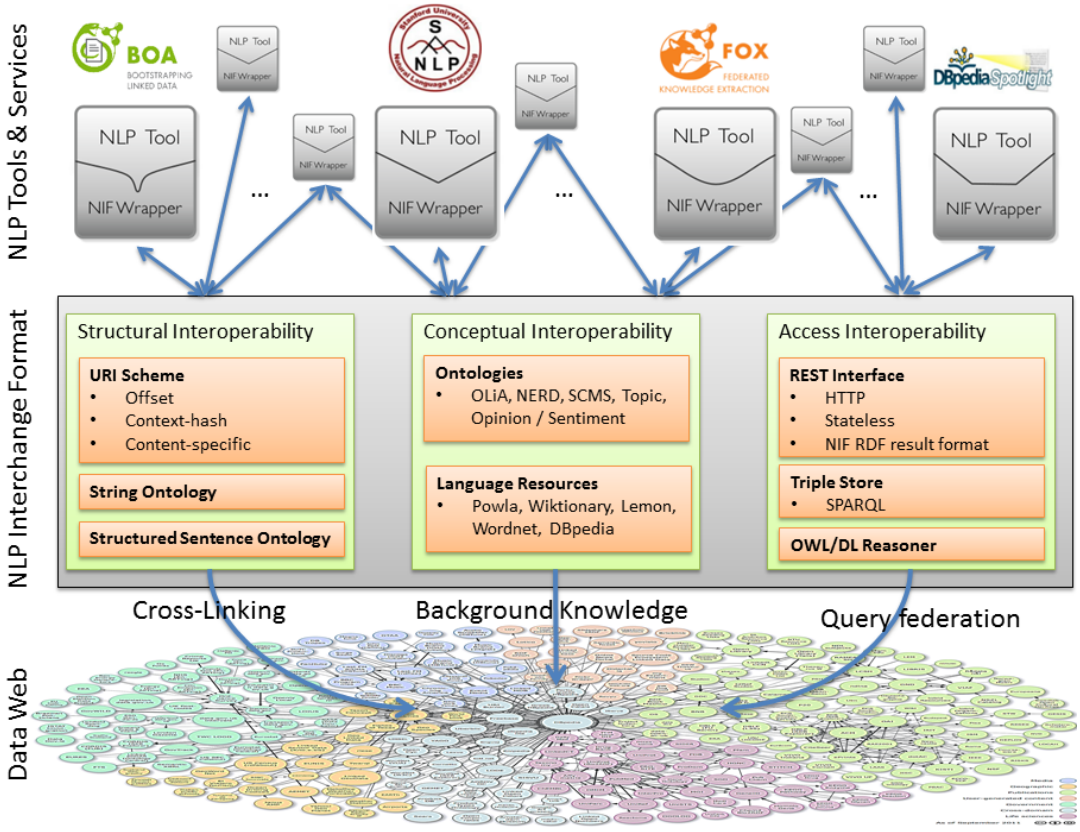


Figure 1: NIF architecture aiming at establishing a distributed ecosystem of heterogeneous NLP tools and services by means of structural, conceptual and access interoperability employing background knowledge from the Web of Data.

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.

3.1 Namespace Prefixes

A NIF-1.0 URI is constructed from a namespace prefix and the actual identifier. To annotate web resources, it is straightforward to use the existing document URI as the basis for the prefix. The following guidelines should be met for NIF-1.0 URIs:

1. As NIF is designed for a client-server architecture, the client (requesting a certain NLP task to be performed by a server) must be able to communicate the prefix to the server. All components must therefore support a parameter named `prefix` which determines how the produced NIF URIs will start.
2. The NIF URIs should be produced by a concatenation of the prefix and the identifier. For practical reasons, it is recommended that the client uses a prefix that ends with slash ('/') or hash ('#').
3. The NIF component must use the prefix of the client transparently without any corrections.

EXAMPLE 1. *Recommended prefixes for* `http://www.w3.org/DesignIssues/LinkedData.html` *are:*
1: `http://www.w3.org/DesignIssues/LinkedData.html/`
2: `http://www.w3.org/DesignIssues/LinkedData.html#`

3.2 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 (esp. Section 2.1.1) [5] and builds upon it in terms of encoding and counting character positions. We deliberately choose a different syntax, however, as RFC 5147 is only valid for the media type *plain text* (and thus not for HTML or XML). As we serve a different use case (i.e. the annotation of documents with NLP tools) our URI schemes are applicable for *text* in general. For our purposes, we therefore define *text* as *a character sequence* or simply *everything that can be sensibly opened in a text editor* (e.g. not binary). Offset-based URIs are constructed of four parts separated by an underscore '_':

1. a *scheme identifier*, in this case the string 'offset',
2. the *start index*,
3. the *end index* (the indexes are counting the gaps between the characters starting from 0 as specified in RFC 5147,
4. a *human readable part*, the first 20 (or less, if the string

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

is shorter) characters of the addressed string, urlencoded.

The main advantages of this scheme is that it is easy and efficient to implement and the addressed string is 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.

3.3 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'
5. a *human readable part*, 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 hash more distinguishable. If there is a sentence 'The dog barks.' and the context size is too large, e.g. 10, then 'The', 'dog' and 'barks' would have the same hash. By adding brackets, however, the hash is easily distinguishable: `md5("The dog barks.") != md5("The (dog) barks.") != md5("The dog (barks).")`.

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 POS tag `DT`: md5(' (the) '). The resulting URI is `http://www.w3.org/DesignIssues/LinkedData.html#hash_md5_1_5_8dc0d6c8afa469c52ac4981011b3f582_the`.

Trivially, every string is uniquely addressable, if the context-length is large enough. The following algorithm computes the minimal context-length (MinCl) on a fixed document

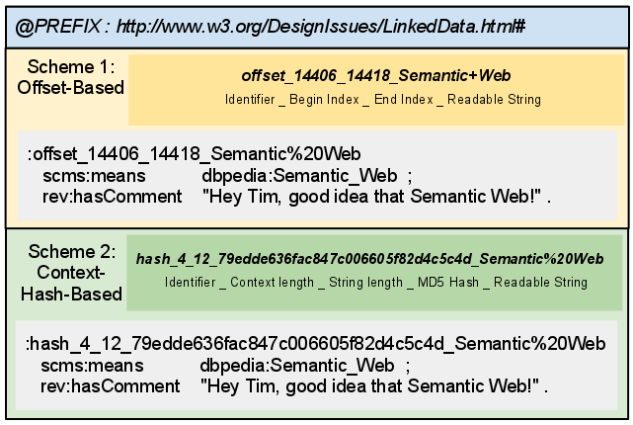


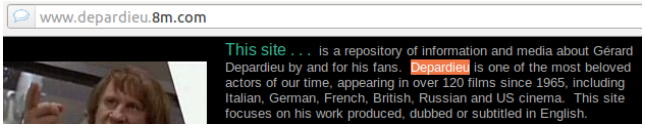
Figure 2: NIF URI schemes: Offset (top) and context-hashs (bottom) are used to create identifiers for strings.

with a given set of annotations.

```
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
```

See Figure 2 for examples of offset and context based URIs. [6]

Example 2. *In the following Web page (`http://www.depardieu.8m.com`) the second occurrence of `Depardieu` is highlighted and is linked in the example to the French DBpedia resource about Gérard Depardieu.*



A NIF 1.0 model, which links the second occurrence of "Depardieu" to the French DBpedia could contain the following RDF triples:

```
@prefix :      <http://www.depardieu.8m.com/#>
@prefix fr:    <http://fr.dbpedia.org/resource>
@prefix str:   <http://nlp2rdf.lod2.eu/schema/string/> .
@prefix scms: <http://ns.aksw.org/scms/> .

:offset_22295_22304_Depardieu
    scms:means      fr:Gerard_Depardieu ;
    rdf:type        str:OffsetBasedString .

:offset_0_54093_%3C!doctype%20html%20publi
```

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

```
    rdf:type          str:OffsetBasedString ;
    rdf:type          str:Document ;
    str:subString     :offset_22295_22304_Depardieu;
    str:sourceUrl     <http://www.depardieu.8m.com/> .
```

This specifies that the document located at `http://www.depardieu.8m.com/` *contains a string starting at position 22295 representing the actor Gérard Depardieu.*

3.4 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:

- Uniqueness. The URIs, produced by the scheme, must uniquely identify the substring.
- Arbitrariness. The URI scheme must be able to produce URIs for arbitrary substrings, while still producing valid URIs. 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[7].
- 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 produced URIs can be utilized to algorithmically find the annotated substring within the text, i.e. calculate the start and end position. This criteria also rates how efficient the begin and end position can be found.
- 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.
- Readability. Although a non-functional criterion, it aids developers as well as users (annotators) in quickly identifying the referenced substring. Readability is enhanced, if the URI contains at least some characters of the referenced string and improves, if it additionally contains the context.
- Functionality. The produced URIs potentially reference strings that are calculated by a certain function (e.g. a case-insensitive regular expression matches all occurrences of a string). The referenced strings are not required to be adjacent in this case. Functionality measures the complexity and expressiveness of the used URI scheme.

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

Table 1 shows a comparison of various URI construction schemes (cf. Section 8). Both schemes used in NIF address shortcomings of other methods with respect to the discussed criteria. Better URI schemes can be envisioned, e.g. a combination of the offset and context-hash schemes. Since both URI schemes can be applied to arbitrary text documents, they are also applicable to HTML, XML, software source code, CSS etc. However, with its addressing scheme identifier NIF is extensible and further annotation schemes (e.g. more content-specific URI schemes such as XPath/XPointer for XML) can be easily included in future.

3.5 NIF Structure Ontologies

In addition to the URI scheme, structural interoperability in NIF is achieved using two ontologies: the *String Ontology* and the *Structured Sentence Ontology* (SSO).

The *String Ontology*[8] comprises a class `String` and a property `anchorOf` to associate URIs in a given text and describe the relations between these string URIs. The available properties are descriptive. Only some properties are actually required for NIF, all others are optional and it is even discouraged to use them per default as they will cause quite a large amount of logical assertions (e.g. transitive properties). The following four are primarily important: `str:Document`, `str:anchorOf`, `str:OffsetBasedString` and `str:ContextHashBasedString`.

The *Structured Sentence Ontology* (SSO)[9] is built upon the String Ontology and provides additional classes for three basic units: *sentences*, *phrases* and *words*. Properties such as `sso:nextWord` and `sso:previousWord` can be used to express relations between these units. Furthermore properties are provided for the most common annotations such as the data type properties for stem, lemma, statistics, etc.

We give two examples of their usage in Figure 3. In the first figure, we can see how the offset based URIs are used for each word in the sentence '*My favorite actress is Natalie Portman!*' and the sentence itself, which coincides with the whole document here. *Word* and *Sentence* are OWL classes in the Structured Sentence Ontology and they are modelled as a subclass of *String*. Each *String* has one mandatory property `anchorOf`, which contains the referenced substring. In Figure 3, a simple annotation is added to each word, i.e. the stem of the referenced string using the Porter algorithm [15]. In most cases, the stem and the original string overlap, except, for example, for the stripped '*e*' of '*favorite*'. We provide an implementation that produces the output shown in Figure 3[10].

The ontologies alongside with the documentation are published with enabled content negotiation, which makes it possible to explore them in an HTML browser and accessing them programmatically at the same time. The documentation was created with the OWLDoc-CLI fork of the Ontology Browser[11] and is shown in Figure 4.

[8] `http://nlp2rdf.lod2.eu/schema/string/`
[9] `http://nlp2rdf.lod2.eu/schema/sso/`
[10] `http://nlp2rdf.org/implementations/snowballstemmer`
[11] `https://bitbucket.org/kurzum/ontology-browser-fork/overview`

	Uniq	Arb	XML	Trans	Addr	Self	Impl	Func	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 [20]	++	++	+	- - -	++	++	+	+	#char=14406-12
LiveURL (Content)	- -	+	-	+	+	-	++	o	#8Semantic12+0x206A73ED
LiveURL (Position)	+	+	-	- -	+	-	-	o	not available
Wilde et al. (Regex)	o	++	+	+	+	+	- -	++	#matching=`Semantic\sWeb`

Table 1: Comparison of URI schemes (cf. Section 8.3 for a detailed description of the different approaches)

4. CONCEPTUAL INTEROPERABILITY VIA ONTOLOGIES

Conceptual interoperability is ensured in NIF by providing ontologies and vocabularies for representing the actual annotations in RDF. We divided the potentially different output of NIF tools into different NLP domains. For each domain a vocabulary was chosen that serves the most common use cases and facilitates interoperability. In simple cases, a property has been designated in the Structured Sentence Ontology. In the more complex cases fully developed linguistic ontologies which already existed were reused. NIF currently supports the following domains:

- *Begin and end index and context.* Adding extra triples for indexes and context produces redundant information as the begin and end index as well as the context can be calculated from the URI. Properties for describing context and begin and end index can be found in the String Ontology.
- *Lemmata & Stems.* Lemma and stem annotations are realized as simple data type properties in the Structured Sentence Ontology.
- *Part of speech tags.* Annotations for part of speech tags must make use of Ontologies of Linguistic Annotations (OLiA), which is described in more detail below.
- *Syntax Analysis* `sso:child`, which is a subproperty of `str:subString` is used to represent *coverage inheritance*. Here as well OLiA provides ontologies modeling e.g. the Stanford Dependency Parse Tree[12].
- *Named Entity Recognition and Entity Linking.* NIF uses the SCMS ontology for describing the meaning of a string. When linking a string to another entity in NIF, the property `scms:means` must be used. Furthermore the type system of the Named Entity Recognition and Disambiguation (NERD) Ontology[13] is exploited.

NIF OLiA [2][14] provides a stable conceptual interface for applications. In Figure 3 we show how this interface is used. The annotations are provided by the *Stanford POS Tagger*[18], which uses the *Penn Tag Set*[15]. OLiA provides an *Annotation Model* for the most frequently used tag sets, such as Penn[16]. These annotation models are then linked to a *reference model*, which provides the interface for applications. Consequently, queries such as 'Return all Strings that are annotated (i.e. typed) as `olia:PersonalPronoun`' are possible, regardless of the underlying tag set. This also guarantees language independence. We provide an implementation that produces the output shown in Figure 3[17].

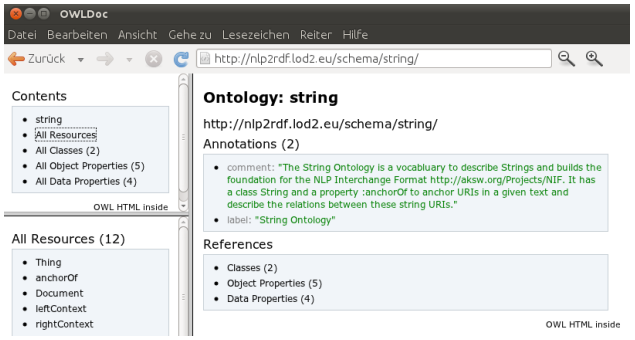


Figure 4: The HTML documentation of the String Ontology.

5. ACCESS INTEROPERABILITY VIA REST SERVICES

Most aspects of access interoperability are already tackled by using the RDF standard and Linked Data principles. This section contains some additions, which further improve interoperability and accessibility of NIF components. While the first two facilitate general interoperability, access provides easier integration and off-the-shelf solutions. Of special importance is the *prefix* parameter as it enables the client to influence directly the RDF output. Note that the RDF in Figure 3 (upper and lower) is produced by different tools, but can be merged directly with the fragment ids under the condition that the URI prefixes are the same.

Workflow. NIF can be used for import and export of data from and to NLP tools. Therefore NIF enables to create ad-hoc workflows following a client-server model or the SOA principle. Following such an approach, clients are responsible for implementing the workflow. Figure 1 shows the communication model. The client sends requests to the different tools either as text or RDF and then receives responses in RDF. This RDF can be aggregated into a local RDF model. Transparently, external data in RDF can also be requested and added without using additional formalisms. For acquiring and merging external data from knowledge bases, all existing RDF techniques and tools can be used.

[12] `http://nachhalt.sfb632.uni-potsdam.de/owl/stanford.owl`
[13] `http://nerd.eurecom.fr/ontology/`
[14] `http://nachhalt.sfb632.uni-potsdam.de/owl/`
[15] `http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html`
[16] `http://nachhalt.sfb632.uni-potsdam.de/owl/penn.owl`
[17] `http://nlp2rdf.org/implementations/stanford-corenlp`

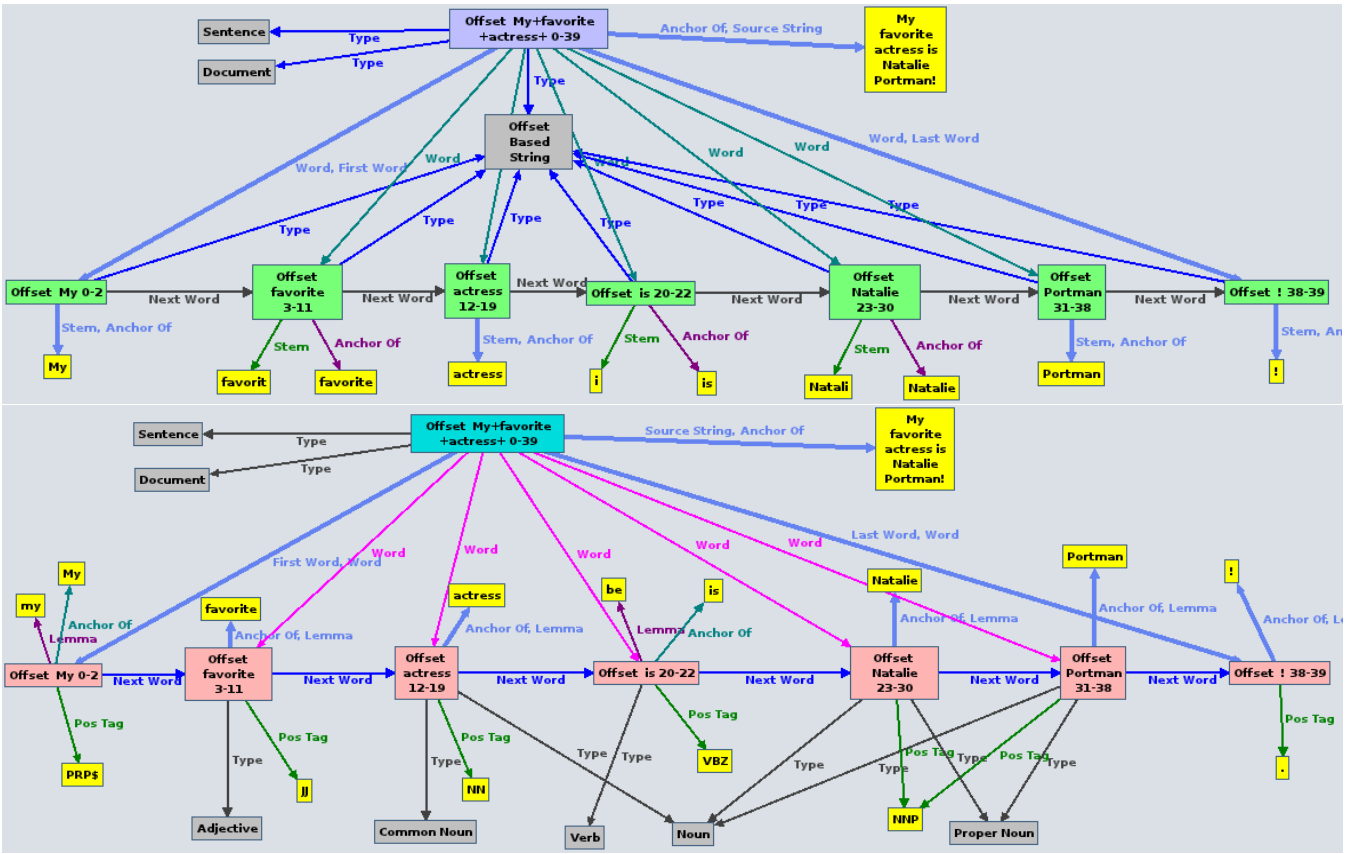


Figure 3: Two visualisations of NIF annotations for the sentence 'My favorite actress is Natalie Portman!' – Top: An annotation for the stem is added to each word. Bottom: The yellow boxes show the actual annotations (POS tags) produced by the Stanford Tagger, while the gray OWL classes are provided by OLiA.

Interfaces. Currently the main interface is a wrapper that provides the NIF Web service. Other interfaces, such as CLI or a Java interface (using Jena) are simple to create. A NIF web service must be *stateless*, *HTTP method agnostic* (i.e. treat POST and GET the same) and accept the following *parameters*:

- *Input type* (required, `input-type = text | nif-owl`). Determines the content required for the next parameter input, either plain text or RDF/XML in NIF.
- *Input* (required, `input = text | rdf/xml`). Either URL encoded text or URL encoded RDF/XML format in NIF.
- *Compatibility* (optional, `nif = true | nif-1.0`). If the application already has a web service, this parameter enables NIF output (allows to deploy NIF in parallel to legacy output). It is recommend for a client to always send this with each request.
- *Format* (optional, `format = rdfxml | ntriples | n3 | turtle | json`). The RDF serialisation format. Standard RDF frameworks support all the different formats.
- *Prefix* (optional, `prefix = uriprefix`). A URL encoded prefix, which must be used for any URIs that the client will create. If missing it should be substituted by a sensible default (e.g. the web service URI).
- *URI Scheme* (optional, `urirecipe = offset | context-hash`). The URI scheme that should be used (default is offset).

6. 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[18]), the senses of each page tend to remain relatively stable after a certain number of revisions [7].

We downloaded a Wikipedia dump with the full edit revision history[19]. 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[20]. 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

[18] `http://www.wikistatistics.net/wiki/en/edits/365`
[19] `http://dumps.wikimedia.org/enwiki/20111007/`
[20] code from `http://www.mediawiki.org/wiki/Extension:ActiveAbstract`

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 2: 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.

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 8.3 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 2.

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: $\frac{a+2cl-1}{n+1}$.

7. NIF IMPLEMENTATION: NLP2RDF PLATFORM AND WRAPPERS

NLP2RDF[21] is the main platform for the adoption of NIF. It serves as a host for NIF 1.0 and will host future specifications. Moreover, it builds a community around NIF by providing demos, development guides, code samples, tutorials, challenges and a mailing list. The platform was used as a base for the developer study described in the sequel.

We performed a NIF developer study by assigning the task of developing NIF wrappers for 6 popular NLP tools to 5 postgraduate students at our institute. Wrappers were developed for UIMA, GATE-ANNIE, Mallet, MontyLingua, OpenNLP and DBpedia Spotlight by one developer each[22]. Table 3 summarizes the results of our NIF developer study. The first section contains the self-assessment of the developers regarding their experience in Semantic Web, NLP, Web Services and application development frameworks on a scale from 1 (no experience) to 5 (very experienced). The next section summarizes the required development effort in hours including learning the NLP tool, learning NIF and performing the complete wrapper implementation. The development effort in hours (ranging between 3 and 40 hours) as well as the number of code lines (ranging between 110 and 445) suggest, that the implementation of NIF wrappers is easy and fast for an average developer. The next section in Table 3 summarizes the NIF assessment by the developers regarding their experience during the development with respect to the adequacy of the general NIF framework, the coverage of the provided ontologies and the required extensibility. All developers were able to map the internal data structure to the NIF URIs to serialize RDF output (Adequacy). Although NIF did not provide a NLP Domain Ontology for Mallet the developer was able to create a compatible OWL Ontology to represent Topic Models[23]. He also developed a wrapper that can read NIF format. In this manner, Mallet can be supplied with tokenization (`sso:Word`), `sso:stem` and `sso:StopWord` annotations via NIF. Both UIMA and GATE are extensible frameworks and NIF was currently not able to provide NLP domain ontologies for all possible domains, but only for the used plugins in this study. After inspecting the software the developers agreed however that NIF is general enough and adequate to provide a generic RDF output based on NIF using literal objects for annotations. In case of UIMA such an RDF serializer already exists[24]. The developer compared the triple count for a 75 word document and the output of NIF was significantly lower(238 compared to 1137 triples).

Finally, the last section contains an assessment of the NIF approach by the developers regarding the perceived scalability, interoperability, quality of the documentation, the usefulness of the reference implementation, the learning curve / entrance barrier and the performance overhead on a scale from 1 (very low) to 5 (very high). Also, here the results suggest, that NIF lives up to its promise of ease-of-use and increased interoperability and is generally perceived positive by developers. The developed wrappers are available as open source[25] and hosted as a web service[26].

8. RELATED WORK

Related work primarily concerns other annotation formats, the relationship to NLP frameworks as well as related URI schemes.

8.1 Other Annotations Formats

Annotation Ontology (AO, [3]) 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. [3, 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

[21] `http://nlp2rdf.org`

[22] one student implemented 2 wrappers (UIMA and OpenNLP)

[23] topic ontology: `http://nlp2rdf.lod2.eu/schema/topic/`

[24] `http://uima.apache.org/sandbox.html#rdfcas.consumer`

[25] `http://nlp2rdf.org/implementations`

[26] `http://nlp2rdf.lod2.eu/demo.php`

	UIMA	GATE	Mallet	Monty	Spot	ONLP
Prior knowledge						
Semantic Web	3	4	1	4	3	
NLP	2	1	4	1	3	
Web Services	3	4	2	4	5	
Frameworks	4	4	3	2	1	
Effort (h)	35	20	40	25	20	3
Tool	20	3	4	4	4	1
NIF	5	3	8	3	4	0
Implementation	10	14	28	18	12	2
Lines of code	271	445	≈ 400	252	110	267
Language	Java	Java	Java	Python	Node-JS	Java
Adequacy	✓	✓	✓	✓	✓	✓
Coverage	(✓)	(✓)	0	✓	✓	✓
NIF Extension			✓			
NIF assessment						
Scalability	2	4	3	4	4	
Interoperability	4	5	4	4	5	
Documentation	4	4	3	5	4	
Reference Impl.	5	5	5	n.a.	5	
Entrance barr.	3	3	4	3	4	
Perf. overhead	2	2	3	3	3	

Table 3: Results of the NIF developer case study.

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 [8], the *Graph Annotation Framework (GrAF)* was used to bridge the models of UIMA and GATE. GrAF relies on XML as a serialisation format and the authors describe several work-arounds to represent links between entities (next to the XML file, additional NameMap and mapping descriptor files are required). One strength of NIF is the ability to integrate RDF data and reuse existing ontologies for data integration, which is easily possible as it is based on the RDF and OWL standards. *Extremely Annotational RDF Markup* (EARMARK, [13] 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.

8.2 Relation to other NLP frameworks

With the early Tipster [6] and the more modern UIMA [5], GATE [4], Ellogon [14], Heart-of-Gold [16] and OpenNLP[27] a number of comprehensive NLP frameworks already exist. NIF, however, focuses on interchange, interoperability as well as decentralization and is complementary to existing frameworks. Ultimately, NIF rather aims at establishing an ecosystem of interoperable NLP tools and services (including the ones mentioned above) instead of creating yet another monolithic (Java-)framework. By being directly based on RDF, Linked Data and ontologies, NIF also comprises crucial features such as *annotation type inheritance* and *alternative annotations*, which are cumbersome to implement or not available in other NLP frameworks [17]. With its focus on conceptual and access interoperability NIF also facilitates *language resource* and *access structure* interchangeability, which is hard to realize with existing frameworks. NIF does not aim at replacing NLP frameworks, which are tailored for high-performance throughput of terabytes of text, it rather aims to ease access to the growing availability of heterogeneous NLP web services as already provided by Zemanta and Open Calais.

8.3 Related URI Schemes

LiveURLs [9] [28] 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 fragment. This URL can be shared and the referenced string is highlighted. The **content based** fragment is created using `sSl+c`, where s is the length of the starting word of the selection, S is the starting word of selection, l is the length of the total selection, + is a delimiter, c is the checksum. 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* [20] filed an RFC in April 2008[29] 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 directly comparable 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. Thus the approach is not general enough as it excludes XML and HTML documents per definition. 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* [21] 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`. This scheme

[27] `http://incubator.apache.org/opennlp/`

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

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

produces the URIs with the best readability. 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 URIs. It is for example impossible to annotate a large paragraph due to URL size restrictions. *Wilde and Baschnagel* [19] 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.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we presented 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 see this effort as the first step in a larger research and engineering agenda: Firstly, we expect NIF to be applied to many more tools and NLP applications beyond our current implementations. Secondly, we aim at further enriching NIF with more comprehensive and fine-grained conceptual alignments. Besides the six wrapper implementations, which were created in the developers study, two more (Stanford and Snowball Stemmer) exist as a reference implementation. Several more wrappers will be created during the life time of the LOD2 EU (for Zemanta, NOOJ, Korean POS-tagger,...) to achieve interoperability with the RDF-based LOD2 Stack[30]. The experience gathered during the presented field test of NIF 1.0 will be refined into NIF 2.0 on the community portal http://nlp2rdf.org . For other areas of NLP, UIMA and Gate integration will be extended and the web service wrapping the Stanford Core Framework will receive more options to expose the multitude of languages supported. Ultimately, we will work with the developers and maintainers of these tools and provide the source code and support to integrate NIF-output directly into their projects.

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 and the students that participated in the fields study: Markus Ackermann, Martin Brümmer, Didier Cherix, Marcus Nitzschke, Robert Schulze.

This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

10. REFERENCES

[30] `http://stack.lod2.eu`

[1] S. Auer and J. Lehmann. Making the web a data washing machine - creating knowledge out of interlinked data. *Semantic Web Journal*, 2010.

[2] C. Chiarcos. An ontology of linguistic annotations. *LDV Forum*, 23(1):1–16, 2008.

[3] 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.

[4] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA*, 2002.

[5] D. Ferrucci and A. Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3/4):327–348, 2004.

[6] D. Harman. The darpa tipster project. *SIGIR Forum*, pages 26–28, 1992.

[7] 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.

[8] N. Ide and K. Suderman. Bridging the gaps: Interoperability for graf, gate, and uima. In *Linguistic Annotation Workshop*, pages 27–34. The Association for Computer Linguistics, 2009.

[9] 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.

[10] A. Khalili and S. Auer. Rdface – an authoring environment for the social semantic web. *Submitted to ACM Transactions on Intelligent Systems and Technology Special Issue on Semantic Adaptive Social Web*, 2011.

[11] 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.

[12] P. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *I-Semantics*, 2011.

[13] 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.

[14] G. Petasis, V. Karkaletsis, G. Paliouras, I. Androutsopoulos, and C. D. Spyropoulos. Ellogon: A New Text Engineering Platform. In *LREC 2002*, pages 72–78, Las Palmas, Canary Islands, Spain, May 29–31 2002. European Language Resources Association.

[15] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[16] U. Schäfer. *Heart of Gold – an XML-based middleware for the integration of deep and shallow natural language processing components, User and Developer Documentation.* DFKI Language Technology Lab,

Saarbrücken, Germany, 2005.

[17] M. Schierle. *Language Engineering for Information Extraction*. Phdthesis, Universität Leipzig, Leipzig, 2011. The thesis was submitted and will be defended in December 2011.

[18] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part–of–speech tagger. In *In (EMNLP/VLC-2000)*, pages 63–70, 2000.

[19] 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.

[20] 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].

[21] K. Yee. Text-Search Fragment Identifiers. `http://zesty.ca/crit/draft-yee-url-textsearch-00.txt`, 1998. [Online; accessed 13-April-2011].