Integrating NLP using Linked Data

Sebastian Hellmann[1], Jens Lehmann[1], Sören Auer[1], and Martin Brümmer[1]

University of Leipzig, Institute of Computer Science, AKSW Group,
Augustusplatz 10, D-04009 Leipzig, Germany
{lastname}@informatik.uni-leipzig.de,
http://aksw.org

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 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. In contrast to 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 present several use cases of the second version of the NIF specification (NIF 2.0) and the result of a developer study.

Keywords: Data Integration, Natural Language Processing, 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*, and *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 often differ to a great extent. 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*(cf. Section 4.3, *Spotlight* [11] 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, 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 facilitate the use of Linked Data we developed the *NLP Interchange Format* (NIF). NIF is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. The NIF specification has been released in an initial version 1.0 in November 2011[2] and known implementations for **30** different NLP tools and use cases (e.g. *UIMA*, *Gate's ANNIE* and *DBpedia Spotlight*) exist and a public web demo[3] 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 the *NIF Core Ontology* (structural layer) and a selection of ontologies for describing common NLP terms and concepts (conceptual layer). NIF-aware applications will produce output adhering to the NIF Core Ontology as REST services (access layer). Other than more centralized solutions such as *UIMA* [6] and *GATE* [5], 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

[1] `http://aksw.org/Projects/FOX`
[2] `http://nlp2rdf.org/nif-1-0/`
[3] `http://nlp2rdf.lod2.eu/demo.php`

additional adaptations. Ultimately, we envision an ecosystem of NLP tools and services to emerge using NIF for exchanging and integrating rich annotations.

This article is structured as follows: After describing requirements (Section 2), which guided the development of NIF and the ontology, we present the core concepts of NIF in Section 3, including URI schemes, ontological structures, workflows and extensions. We then describe some of the currently implemented use cases in Section 4. We evaluate NIF by applying it to a large-scale problem, performing a developer study and comparing it to other frameworks in Section 5. Finally, we present lessons learned, conclusions and future work in Section 6.

2 Requirements for NLP Integration

In this section, we will give a list of requirements, we elicited within the LOD2 EU project[4], which influenced the design of NIF. The LOD2 project develops the *LOD2 stack*[5], which integrates a wide range of RDF tools, including a *Virtuoso* triple store as well as Linked Data interlinking and OWL enrichment tools.

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.

Coverage. The wide range of potential NLP tools requires that the produced format and ontology is sufficiently general to cover all or most annotations.

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.

Conceptual Interoperability. In addition to structural interoperability, tools should use the same vocabularies for the same kind of annotations. This refers to **what** annotations are used.

Granularity. The ontology is supposed to handle different granularity not limited to the document level, which can be considered to be 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.

Provenance and Confidence. For all annotations we would like to track, where they come from and how confident the annotating tool was about correctness of the annotation.

Simplicity. We intend to encourage third parties to contribute their NLP tools to the LOD2 Stack and the NLP2RDF platform. Therefore, the format should be as simple as possible to ease integration and adoption.

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 overall memory and index

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

[5] `http://stack.linkeddata.org`

footprint (URI to id look-up tables). Secondly, the complexity of OWL axioms should be low or modularised to allow fast reasoning.

3 NLP Interchange Format (NIF)[6]

3.1 URI Schemes

The idea behind NIF is to allow NLP tools to exchange annotations about text in RDF. Hence, the main prerequisite is that text becomes referenceable by URIs, so that they can be used as resources in RDF statements. In NIF, we distinguish between the *document* d, the *text* t contained in the document and possible *substrings* s_t of this text. Such a substring s_t can also consist of several non-adjacent characters within t, but for the sake of simplicity, we will assume that they are adjacent for this introduction. We call an algorithm to systematically create identifiers for t and s_t a *URI Scheme*. To create URIs, the URI scheme requires a document URI du, a separator sep and the character indices (begin and end index) of s_t in t to uniquely identify the position of the substring. The canonical URI scheme of NIF is based on RFC 5147 [7], which standardizes fragment ids for the text/plain media type. According to RFC 5147, the following URI can address the first occurrence of the substring "Semantic Web" in the text (26610 characters) of the document `http://www.w3.org/DesignIssues/LinkedData.html` with the separator #: `http://www.w3.org/DesignIssues/LinkedData.html#char=717,729` The whole text contained in the document is addressed by "`#char=0,26610`" or just "`#char=0,`". NIF offers several such URI schemes which can be selected according to the requirements of the use case. Their advantages and disadvantages have been investigated in [7] and we will limit ourselves to RFC 5147 in this paper. For practical reasons, the document URI and the separator are henceforth called the `prefix` part of the URI scheme and the remainder (i.e. "`char=717,729`") will be called the `identifier` part. NIF recommends the prefix to end on slash (`/`), hash (#) or on a query component (e.g. `?nif-id=`). Depending on the scenario, we can choose the prefix in the following manner:

Web annotation. If we want to annotate a (web) resource, it is straightforward to use the existing document URL as the basis for the prefix and add a hash (#). The recommended prefix for the 26610 characters of `http://www.w3.org/DesignIssues/LinkedData.html` is: `http://www.w3.org/DesignIssues/LinkedData.html#`

This works best for plain text files either on the web or on the local file system (`file://`). For demonstration purposes, we minted a URI that contains a plain text extraction (19764 characters) created with 'lynx –dump', which we will use as the prefix for most of our examples: `http://persistence.uni-leipzig.org/nlp2rdf/examples/doc/LinkedData.txt#` and `http://persistence.uni-leipzig.org/nlp2rdf/examples/doc/LinkedData.txt#char=333,345` NIF can be used as a true stand-off format linking to external text.

[6] We refer the reader to `http://prefix.cc` for all prefixes used.

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

Web service. If the text is, however, sent around between web services or stored in a triple store, the prefix can be an arbitrarily generated URN[8]. Communication between the NLP tools in NIF is done via RDF and therefore mandates the inclusion of the text in the RDF during the POST or GET request. The main purpose here is to exchange annotations between client and server and the used URIs do not require to resolve to an information resource. NIF requires each web service to have a parameter "prefix" that empowers any client to modify the prefix of the created NIF output. The prefix parameter can be tested at `http://nlp2rdf.lod2.eu/demo.php`.

Annotations as Linked Data. For static hosting of annotations as linked data (e.g. for a corpus), the `/` and query component separator is advantageous. Often the basic unit of a corpus are the individual sentences and it makes sense to create individual prefixes on a per sentence basis.

In the following, we will explain how the relation of document, text and substring can be formalized in RDF and OWL.

3.2 NIF Core Ontology

The NIF Core Ontology[9] provides classes and properties to describe the relations between substrings, text, documents and their URI schemes. The main class in the ontology is `nif:String`, which is the class of all **words over the alphabet of Unicode characters** (sometimes called Σ^*). We built NIF upon the Unicode Normalization Form C, as this follows the recommendation of the RDF standard[10] for `rdf:Literal`. Indices are to be counted in code units. Each URI scheme is a subclass of `nif:String` and puts further restrictions over the syntax of the URIs. For example, instances of type `nif:RFC5147String` have to adhere to the NIF URI scheme based on RFC 5147. Users of NIF can create their own URI schemes by subclassing `nif:String` and providing documentation on the Web in the `rdfs:comment` field.

Another important subclass of `nif:String` is the `nif:Context` OWL class. This class is assigned to the whole string of the text (i.e. all characters). The purpose of an individual of this class is special, because the string of this individual is used to calculate the indices for all substrings. Therefore, all substrings have to have a relation `nif:referenceContext` pointing to an instance of `nif:Context`. Furthermore, the datatype property `nif:isString` can be used to include the reference text as a literal within the RDF as is required for the web service scenario. An example of NIF Core can be seen on the top left of Figure 1.

3.3 Workflows, Modularity and Extensibility of NIF

Workflows. NIF web services are loosely coupled and can receive either text or RDF. To allow seamless NLP integration, clients should create work flows

[8] `http://tools.ietf.org/html/rfc1737`
[9] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#`
[10] `http://www.w3.org/TR/rdf-concepts/#section-Literals`

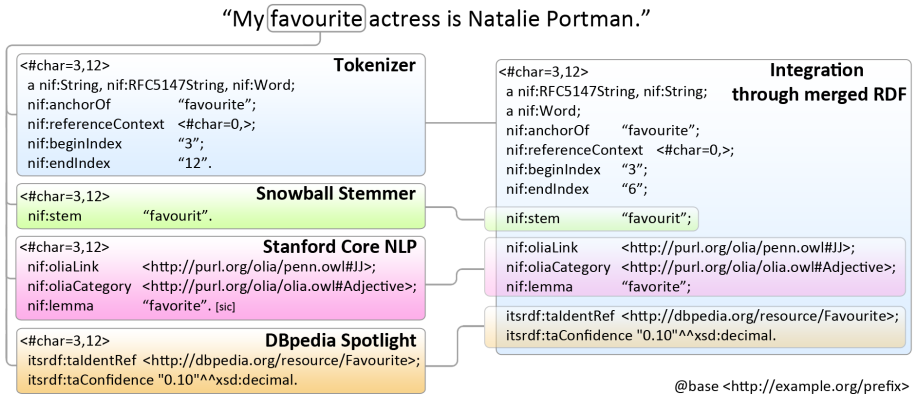

Fig. 1. An example of NIF integration. Tool output from four tools is merged via URLs. Reproducible at the NIF demo site: `http://nlp2rdf.lod2.eu/demo.php`

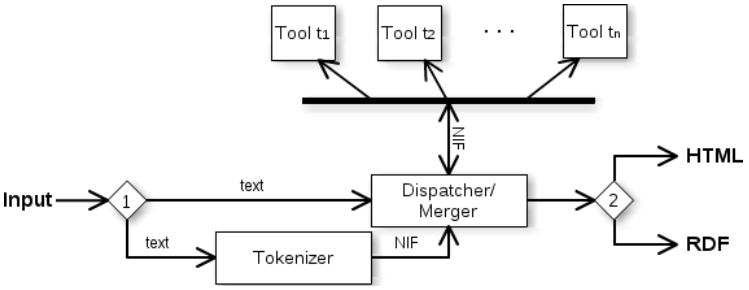

Fig. 2. Workflow implemented by the NIF Combinator [8]

where the text is normalized (Unicode) at the beginning and tokenization is provided. Figure 2 shows one of the possible workflows that uses an NLP tokenizer in a preprocessing step [8]. Based on the normalization and tokenization, the combined RDF of several tools merges naturally based on the subject URIs as shown in Figure 1. Tokenization conflicts are a recognized problem in NLP; other algorithms are applicable (cf. [3]), if no *a priori* resolution is applied.

Logical Modules: The NIF ontology[11] is split in three parts: The *terminological model* is lightweight in terms of expressivity and contains the core classes and properties. Overall, it has 125 axioms, 28 classes, 16 data properties and 28 object properties. The *inference model* contains further axioms, which are typically used to infer additional knowledge, such as transitive property axioms. The *validation model* contains axioms, which are usually relevant for consistency checking or constraint validation[12], for instance class disjointness and functional

[11] Available at `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/version-1.0/`.

[12] See e.g. `http://clarkparsia.com/pellet/icv/`.

properties. Depending on the use case, the inference and validation model can optionally be loaded. Overall, all three NIF models consist of 177 axioms and can be expressed in the description logic $\mathcal{SHIF}(\mathcal{D})$ with exponential reasoning time complexity [17].

Vocabulary modules: NIF incorporates existing domain ontologies via vocabulary modules to provide best practices for NLP annotations for the whole breadth of the NLP domain, e.g. FISE (see below), ITS (Sect. 4.1), OLiA (Sect. 4.2), NERD [13].

Granularity Profiles: We will give a brief technical introduction into the four different granularities, which are shown in Figure 3.

NIF Simple. Basic properties describe the strings and the reference text unambiguously. NIF simple allows to express *the best estimate* of an NLP tool in a flat data model. The profile is sufficient for most use cases including simple NLP tasks, such as POS tagging or NER. The client is responsible to resolve any inconsistencies and merge the data retrieved in a web service context. Most properties such as `itsrdf:taIdentRef` and `nif:oliaLink` are functional and enforce (if validated) at most one annotation of a certain type per string. Confidence can be encoded for each annotation, though no alternatives can be included. Provenance can only be encoded for one tool, which is sufficient in the context of a single web service request.

NIF Simple Underspecified. A variant of the above this profile may only be applied, iff the *prefix* equals the annotated information resource. Other information (especially the reference context) may be omitted and later recreated from the identifier part of the URI scheme. In our running example, the file LinkedData.txt can be retrieved from the Web and the identifier `<char=333,345>` would be enough to explicate the remaining triples on the client side. The profile has the lowest triple count (*one triple per annotation*), but can not be queried effectively with SPARQL and has the risk of running out of sync with the primary data.

NIF Stanbol. Alternative annotations with different confidence as well as provenance information (i.e. which NLP engine produced which annotation) can be attached to the additionally created URN for each annotation. The NIF Stanbol profile is complementary to NIF simple, transformation is lossless, except, of course, for the alternatives and the provenance information. The model is interesting for creating algorithms that try to optimize output from different engines and require the detailed NLP graph.

NIF OA (Open Annotation). Open Annotation provides the most expressive model, but requires more triples and creates up to four new URNs per annotation.

Apache Stanbol[13] is a Java framework, that provides a set of reusable components for semantic content management. One component is the content enhancer that serves as an abstraction for entity linking engines. For Stanbol's use case, the NLP graph is required, including provenance, confidence of annotations as well as full information about alternative annotations (often ranked by confidence)

[13] `http://stanbol.apache.org`

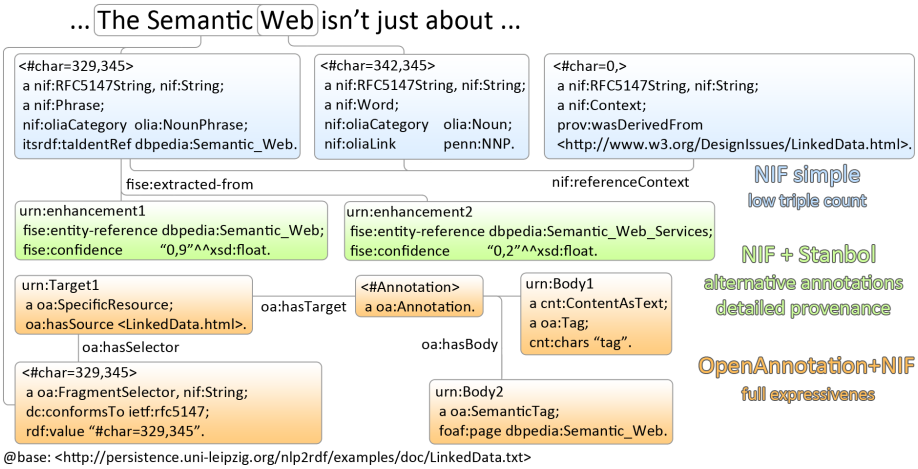

Fig. 3. Three of the four granularity profiles of NIF. Open annotation is able to use NIF identifiers as `oa:Selector`.

and not only the best estimate. The FISE ontology[14] is integrated into NIF as a vocabulary module and a NIF implementation is provided by the project(cf. Section 5.2).

Open Annotation Data Model (OA[15], formerly the annotation ontology[4]) was originally devised as an 'open ontology in OWL-DL for annotating scientific documents on the web' and is now advanced by the Open Annotation W3C Community Group. OA provides structural mechanisms to annotate arbitrary electronic artifacts and resources (including images, websites, audio and video). OA is a generic approach that succeeds in creating an annotation framework for a plethora of use cases and distinguishes between the *body*, the *target* and the *annotation* itself by creating URNs for each of the parts. As NLP has special requirements regarding scalability, NIF offers two more granularities targeting reduced overhead and three different levels of reasoning. Furthermore, OA is domain-agnostic, while NIF defines best practices for annotations as well as a community infrastructure to agree on common domain annotations and reference ontologies to create interoperability in the NLP domain.

Especially noticeable is the fact that all three main granularities are complementary and can be kept together. A client could keep token and POS tags in NIF simple to reduce triple count, encode entity linking in NIF Stanbol to keep the alternatives and then have user tags and comments in NIF OA, because OA allows to reply to previous comments (annotations on annotations). An implementation is for example provided in the OpenPHACTS system.[16]

[14] `http://fise.iks-project.eu/ontology/`
[15] `http://www.openannotation.org`
[16] `http://ubo.openphacts.org/index.php?id=4684`

4 Use Cases for NIF

4.1 Internationalization Tag Set 2.0

The *Internationalization Tag Set* (ITS) Version 2.0 is a W3C working draft, which is in the final phase of becoming a W3C recommendation. Among other things, ITS standardizes HTML and XML attributes which can be leveraged by the localization industry (especially language service providers) to annotate HTML and XML nodes with processing information for their data value chain. In the standard, ITS defines 19 *data categories*[17], which provide a shared conceptualization by the W3C working group and its community of stakeholders. An example of three attributes in an HTML document is given here:

```
<html><body><h2 translate="yes">Welcome to <span
   its-ta-ident-ref="http://dbpedia.org/resource/Dublin" its-within-text="yes"
   translate="no">Dublin</span> in
   <b translate="no" its-within-text="yes">Ireland</b>!</h2></body></html>
```

As an outreach activity, the working group evaluated *RDFa*[18] to create a bridge to the RDF world, but concluded that the format was not suitable to serve as a best practice for RDF conversion. The main problem was that the defined ITS attributes annotate the text within the HTML nodes, but RDFa only has the capability to annotate resources with the text in the node as an object. RDFa lacks subject URIs, which refer to the text within the tags. Although it is theoretically possible to extract provenance information (i.e. offsets and position in the text), the RDFa standard does not include this use case and current RDFa parsers (with the exception of *viejs.org*) do not implement such an extraction.

In a joint effort, the ITS 2.0 RDF ontology[19] was developed using NIF, which was included within the proposed standard alongside an algorithm for a round-trip conversion of ITS attributes to NIF[20] (simple granularity). Provenance can be kept with an XPointer/XPath fragment identifier.

```
@base <http://example.com/exampledoc.html#> .
<char=0,> a nif:Context , nif:RFC5147String ;
<char=11,17>
    nif:anchorOf          "Dublin" ;
    itsrdf:translate      "no";
    itsrdf:taIdentRef     dbpedia:Dublin ;
    # needed provenance for round-tripping
    prov:wasDerivedFrom   <xpath(/html/body[1]/h2[1]/span[1]/text()[1])> ;
    nif:referenceContext <char=0,> .
```

NIF successfully creates a bridge between ITS and RDF and a round-trip conversion was recently implemented as a proof-of-concept. Therefore, NIF can be expected to receive a wide adoption by machine translation and industrial language service providers. Additionally, the ITS Ontology provides well modeled and accepted properties, which can in turn be used to provide best practices for NLP annotations.

[17] http://www.w3.org/TR/its20/#datacategory-description
[18] http://www.w3.org/TR/rdfa-syntax/
[19] http://www.w3.org/2005/11/its/rdf#
[20] ttp://www.w3.org/TR/its20/#conversion-to-nif

4.2 OLiA

The *Ontologies of Linguistic Annotation* (OLiA) [2][21] provide stable identifiers for morpho-syntactical annotation tag sets, so that NLP applications can use these identifiers as an interface for interoperability. OLiA provides *Annotation Models (AMs)* for fine-grained identifiers of NLP tag sets, such as *Penn*[22]. The individuals of these annotation models are then linked via `rdf:type` to coarse-grained classes from a *Reference Model (RM)*, which provides the interface for applications. The coverage is immense: OLiA comprises over 110 OWL ontologies for over 34 tag sets in 69 different languages, the latest addition being the Korean *Sejong tagset*. The benefit for application developers is three-fold:

1. **Documentation.** OLiA allows tagging with URIs (e.g. `http://purl.org/olia/penn.owl#DT`) instead of just short cryptic strings such as "DT". Developers who are unfamiliar can open the URL in an ontology browser and read the included documentation collected from the literature.
2. **Flexible Granularity.** For a wide range of NLP tools who built upon POS tags, very coarse-grained tags are sufficient. For example for keyword extraction, entity recognition and lemmatization, it is often not necessary to distinguish between singular/plural or common/proper noun. OLiA maps all four tags to a common class `olia:Noun`. Such a mapping exists for almost all tags and can be easily reused by developers for a wide range of tag sets.
3. **Language Independence.** AMs for different languages are mapped to the common RM providing an abstraction across languages.

NIF provides two properties: `nif:oliaLink` links a `nif:String` to an OLiA-AM. Although a reasoner could automatically deduce the abstract type of each OLiA individual from the RM, it was a requirement that the coarse-grained types should be linked redundantly to the strings as well in case reasoning services are not available or would cause high overhead. Therefore, an OWL annotation property `nif:oliaCategory` was created as illustrated in the following example.

```
<char=342,345> a nif:String, nif:RFC5147String ;
    nif:oliaLink        penn:NNP  ;
    nif:oliaCategory    olia:Noun , olia:ProperNoun .
# deducable by a reasoner:
penn:NNP       a olia:Noun,  olia:ProperNoun .
```

The NLP2RDF project provides conversions of the OLiA OWL files to CSV and Java HashMaps for easier consumption.[23] Consequently, queries such as 'Return all strings that are annotated (i.e. typed) as `olia:PersonalPronoun` are possible, regardless of the underlying language or tag set.

All the ontologies are available under an open license.[24]

[21] `http://purl.org/olia`
[22] `http://purl.org/olia/penn.owl`
[23] `http://olia.nlp2rdf.org/owl/{Penn.java|penn.owl.csv|penn-link.rdf.csv}`
[24] `http://sourceforge.net/projects/olia/`

4.3 RDFaCE

RDFaCE (RDFa Content Editor)[25] [10] is a rich text editor that supports WYSIWYM (What-You-See-Is-What-You-Mean) authoring including various views of the semantically enriched textual content. One of the main features of RDFaCE is combining the results of different NLP APIs for automatic content annotation. The main challenge here is the heterogeneity of the existing NLP APIs in terms of API access, URI generation and output data structure. Different NLP APIs use different URL parameter identifiers such as content, text, lookupText etc. to indicate the input for the REST API. Furthermore, for identifying the discovered entities they use either their own URI schemes such as:
`http://d.opencalais.com/genericHasher-1/e7385008-0856-3afc-a40f-0000dcd27ded`
`http://api.evri.com/v1/organization/university-of-leipzig-0xbdb4d`
or external URIs such as:
`http://dbpedia.org/resource/University_of_Leipzig`
`http://mpii.de/yago/resource/University_of_Leipzig`
Another important issue is that each API returns different properties with different identifiers and in a different structure.

To cope with these heterogeneity issues, RDFaCE uses a server-side proxy. At first, the proxy handled the access heterogeneity by hard coding the input parameters and connection requirements of each individual API. After implementing NIF, the integration process was simplified to a great extent by abstracting the diversity of different NLP APIs and introducing an interoperability layer. Adding new NLP APIs to RDFaCE became straightforward and additional efforts to handle heterogeneity between different data formats were removed.

5 Evaluation

5.1 Quantitative Analysis with Google Wikilinks Corpus

To evaluate NIF against other formats for modeling NLP annotations as RDF, we converted the *Wikilinks Corpus* [16] to linked data using NIF.

The Wikilinks Corpus. The Google Wikilinks Corpus[26] is a large scale corpus, which collects found hyperlinks to Wikipedia from text fragments gathered from over 3 million web sites. Every item consist of the *website URI* of the crawled sites and a number of *mentions*, including the English Wikipedia link, the hyperlink anchor text, its byte offset and in most cases a *context string*, i.e. suffix and prefix around the anchor of variable length. With over 3 million items and 40 million mentions it surpasses most free corpora by far and serves as a very good testbed for measuring scalability of RDF as well as performance of NER Disambiguation tools in a noisy and multi-domain environment.

Conversion to NIF and Hosting as Linked Data. 15% of the items did not contain any mention with context strings and where therefore omitted.

[25] `http://aksw.org/Projects/RDFaCE`
[26] we used the `https://code.google.com/p/wiki-link/wiki/ExpandedDataset`

	NS	NSI	NSTAN	NSTANI	OA	UC
# triples	477 250 589	316 311 355	511 220 514	350 281 280	577 488 725	607 563 176
# generated URIs	76 850 241	42 880 316	110 820 166	76 850 241	169 849 625	189 342 046
# percentage	100%	66.28%	107.12%	73.40%	121.00%	127.30%
# percentage URIs	100%	55.79%	144.2%	100%	221.01%	246.38%

Table 1. Comparison of triple count and minted URIs. Percentage relative to NS. (NS=NIF Simple, NSI=NIF Simple Ideal, NSTAN=NIF Stanbol, NSTANI=NIF Stanbol Ideal, OA = Open Annotation, UC = UIMA Clerezza).

Every mention was then converted into two resources, a `nif:Context` resource for each context string and the mention resource itself with `nif:beginIndex`, `nif:endIndex`, `itsrdf:taIdentRef` and `nif:referenceContext`. The created context resource was then linked via `nif:broaderContext` to a URI of the form:[27] `http://wiki-link.nlp2rdf.org/api.php?uri=$websiteURI#char=0,` The corpus resulted in 10,526,423 files hosted in an Apache2 file system[28] and a 5.6 GB turtle dump (37.8 GB uncompressed, original size 5.3 GB / 18 GB). Table 1 gives a comparison of created triples and URIs by different profiles as well as OA and UIMA Clerezza[29] . Because we only have text snippets for each mention, we were forced to create one context resource per mention. If the whole plain text of the website were available (as according to the creators is planned in the near future), NIF could further reduce the number of triples to 66.28% (NSI), by using the whole document text as context. This is not the underspecified variant, which would even cause another large reduction of triples.

5.2 Questionnaire and Developers Study for NIF 1.0

With NLP2RDF[30], we provide reference implementations and demo showcases to create a community around NIF and support its adoption.

NLP tools can be integrated using NIF, if an adapter is created, that is able to parse a *NIF Model* into the internal data structure and also to output the NIF as a serialization. The effort for this integration is usually very low; just a parser and a serializer have to be written. An NLP pipeline can then be formed by either passing the NIF RDF Model from tool to tool (sequential execution) or passing the text to each tool and then merge the NIF output to a large model (parallel execution). After the release of NIF version 1.0 in November 2011[31] a total of **30** implementations for different NLP tools and converters were created (8 by the authors, including Wiki-link corpus, 13 by people participating in our survey and 9 more, we have heard of). In 2011, we performed a first round of the NIF developer study by assigning the task of developing NIF 1.0 wrappers

[27] e.g.`http://wiki-link.nlp2rdf.org/api.php?uri=http://phish.net/song/on-green-dolphin-street/history#char=0,`

[28] `http://wiki-link.nlp2rdf.org/`

[29] NS was generated, all others calculated based on `http://persistence.uni-leipzig.org/nlp2rdf/doc/wikilink-stats.txt`

[30] `https://github.com/NLP2RDF`

[31] `http://nlp2rdf.org/nif-1-0/`

for 6 popular NLP tools to 6 postgraduate students at our institute. Wrappers were developed for UIMA, GATE-ANNIE, Mallet, MontyLingua, OpenNLP and DBpedia Spotlight (first six lines of Table 2). The remaining entries were created in 2012 and 2013 by adopters of NIF 1.0, some even already implementing a draft version of 2.0. Table 2 summarizes the results of our NIF developer study.

The first columns contain 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 middle columns summarize 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 displays 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. Both UIMA, GATE and Stanbol 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 the UIMA Clerezza consumer an RDF serializer already exists and we have compared potential output in Section 5.1.

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 (low) to 5 (very high). The results[32] suggest, that NIF lives up to its promise of ease-of-use and increased interoperability and is generally perceived positive by developers.

5.3 Qualitative Comparison with other Frameworks and Formats

In [9], 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) and has recently been standardized by ISO. GrAF is meant to serve as a pivot format for conversion of different annotation formats and is able to allow a structural mapping between annotation structures. GrAF is similar to the Open Annotation effort. *Extremely Annotational RDF Markup* (EARMARK, [12]) is a stand-off format to annotate text with markup (XML, XHTML) and represent the markup in RDF including overlapping annotations. The main method to address content is via ranges that are similar to the NIF

[32] more data at `http://svn.aksw.org/papers/2013/ISWC_NIF/public/devstudy.pdf`

Tool	Developer	Type	SW	NLP	Web Services	Frameworks	Effort (h)	Tool	NIF	Implementation	LoC	Lang	Adequacy	Coverage	NIF Extension	Scalability	Interoperability	Documentation	Reference Impl.	Entrance barr.	Perf. overhead
UIMA	MB	w	3	2	3	4	35	20	5	10	271	Java	✓	no (✓for POS)	n.a.	2	4	4	5	3	2
GATE	DC	w	4	1	4	4	20	3	5	14	445	Java	✓	no (✓for POS)	n.a.	4	5	4	5	3	2
Mallet	MA	w	1	4	2	3	40	4	8	28	400	Java	✓	no (NIF 1.0)	✓	3	4	3	5	4	3
MontyLingua	MN	w	4	1	4	2	25	4	3	18	252	Python	✓	✓	n.a.	4	4	5	-	3	3
Spotlight	RS	w	3	3	5	1	20	4	4	12	110	Node-JS	✓	no (NIF 1.0)	✓	4	5	4	5	4	3
OpenNLP	MB	w	3	2	3	4	3/8	1	0*	2	267	Java	✓	no (NIF 1.0)	✓	2	4	4	5	3	2
OpenCalais	AL	w	4	4	3	4	32	6	6	20	201	PHP	✓	no (NIF 1.0)	✓	3	3	4	5	4	3
Zemanta	MV	w	3	4	4	4	24	1	3	20	235	Python	✓	✓	n.a.	3	4	3	5	4	3
SemanticQuran	MS	w	4	3	2	2	25	1	6	18	500	Java	✓	✓	n.a.	5	5	4	5	4	2
ITS2NIF	FS	w	3	3	3	3	20	7	7	6	72	XSLT	✓	✓	n.a.	3	3	3	3	1	3
THD	MD	w	4	2	5	3	20	7	8	5	300	Java	✓	no	✓	3	4	2	2	3	3
STANBOL	RW	w/i	5	4	4	4	28	?	8	20	400	Java	✓	no	∼	?	?	3	?	2	2
Spotlight	MN	i	4	2	4	3	24	8	1	15	212	Scala	✓	✓	n.a.	4	4	3	4	3	2
Coat	SL	i	2	1	2	4	165	10	5	150	-	Java	✓	✓	n.a.	3	-	3	-	3	-
DrugExractor	AK	w	4	5	4	4	16	1	5	10	30	Java	∼	no	✓	3	3	4	-	1	-

Table 2. Results of the NIF developer case study.

URI scheme. *TELIX* [14] extends SKOS-XL[33] and suggests RDFa as annotation format. We were unable to investigate *TELIX* in detail, because neither an implementation nor proper documentation was provided. In Section 4.1, we have argued already that RDFa is not a suitable format for NLP annotations in general. The usage of SKOS-XL by TELIX only covers a very small part of NLP annotations, i.e. lexical entities. With the early *Tipster* and the more modern *UIMA* [6], *GATE* [5], *Ellogon*, *Heart-of-Gold* and *OpenNLP*[34] 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 [15]. 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, for example, already provided by *Zemanta* and *Open Calais*.

[33] `http://www.w3.org/TR/skos-reference/skos-xl.html`
[34] `http://opennlp.apache.org`

6 Lessons Learned, Conclusions and Future Work

Lessons Learned. Our evaluation of NIF since the publication of NIF 1.0 in the developers study has been accompanied by extensive feedback from the individual developers and it was possible to increase ontological coverage of NLP annotations in version 2.0, especially with the ITS 2.0 / RDF Ontology, NERD [13], FISE and many more ontologies that were available. Topics that dominated discussions were scalability, reusability, open licenses and persistence of identifiers. Consensus among developers was that RDF can hardly be used efficiently for NLP in the internal structure of a framework, but is valuable for exchange and integration. The implementation by Apache Stanbol offered a promising perspective on this issue as they increased scalability by transforming the identifiers used in OLiA into efficient Java code structures (enums). Hard-compiling ontological identifiers into the type systems of Gate and UIMA seems like a promising endeavour to unite the Semantic Web benefits with the scalability requirements of NLP. A major problem in the area remains the URI persistence. Since 2011 almost all of the mentioned ontologies either changed their namespace and hosting (OLiA and NIF itself) or might still need to change (Lemon, FISE), which renders most of the hard-coded implementations useless.

Conclusions. In this article, we presented the NIF 2.0 specification and how it was derived from NLP tool chain integration use cases. NIF is already used in a variety of scenarios, which we described in the article. We conducted an evaluation by applying NIF to a large NLP corpus, which we provide as Linked Data for further re-use. Furthermore, a developer use case study shows that NIF wrappers can be implemented in one week and the specification has sufficient coverage to wrap the output of existing NLP tools. Overall our study has also shown that ontologies are a good way to achieve interoperability across different programs and programming languages.

Future Work. The NIF/NLP2RDF project can be seen as an umbrella project creating bridges between different communities to achieve interoperability in the NLP domain via ontologies. The currently active and fruitful collaborations such as Stanbol, Spotlight, Open Annotation, ITS, OLiA, NERD are yet mostly centered on stakeholders from the Semantic Web. With the soon-to-start LIDER EU project, NLP2RDF will outreach to core NLP projects such as CLARIN, ELRA and LanguageGrid.[35] Identifying incentives relevant for stakeholders outside the Semantic Web community remains an open challenge as in this initial phase NIF focused primarily on middleware interfaces and not directly on end user problems. We will investigate existing (and hopefully directly reusable) approaches on Semantic Web workflows such as SADI, Taverna and WSMO-Lite.[36] A NIF workflow, however, can obviously not provide any better performance (F-measure, efficiency) than a properly configured UIMA or GATE pipeline with the same components. NIF targets and benefits developers in terms

[35] `http://www.clarin.eu/node/3637`, `http://elra.info`, `http://langrid.org`

[36] `http://sadiframework.org`, `http://www.taverna.org.uk`, `http://www.w3.org/Submission/WSMO-Lite`

of entry barrier, data integration, reusability of tools, conceptualisation and off-the-shelf solutions. Early adoption of open-source as well as industry projects is manifesting, but an exhaustive overview and a machine-readable collection of available implementations and deployments is yet missing.

Acknowledgments. We especially thank all contributors to NIF. The list is really large and will be maintained at `http://persistence.uni-leipzig.org/nlp2rdf/` This work was supported by grants from the European Union's 7th Framework Programme provided for the projects LOD2 (GA no. 257943) and GeoKnow (GA no. 318159).

References

1. S. Auer and S. Hellmann. The web of data: Decentralized, collaborative, interlinked and interoperable. In *LREC*, 2012.
2. C. Chiarcos. Ontologies of linguistic annotation: Survey and perspectives. In *LREC*. European Language Resources Association, 2012.
3. C. Chiarcos, J. Ritz, and M. Stede. By all these lovely tokens... merging conflicting tokenizations. *Language Resources and Evaluation*, 46(1):53–74, 2012.
4. P. Ciccarese, M. Ocana, L. Garcia Castro, S. Das, and T. Clark. An open annotation ontology for science on web 3.0. *Biomedical Semantics*, 2:S4+, 2011.
5. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL*, 2002.
6. 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.
7. S. Hellmann, J. Lehmann, and S. Auer. Linked-data aware uri schemes for referencing text fragments. In *EKAW 2012*, LNCS 7603. Springer, 2012.
8. S. Hellmann, J. Lehmann, S. Auer, and M. Nitzschke. Nif combinator: Combining nlp tool output. In *EKAW 2012*, pages 446–449, 2012.
9. N. Ide and K. Suderman. Bridging the Gaps: Interoperability for Language Engineering Architectures using GrAF. *LRE Journal*, 46(1):75–89, 2012.
10. A. Khalili, S. Auer, and D. Hladky. The rdfa content editor - from wysiwyg to wysiwym. In *COMPSAC*, 2012.
11. P. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *I-Semantics*, 2011.
12. 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.
13. G. Rizzo, R. Troncy, S. Hellmann, and M. Bruemmer. NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In *LDOW*, 2012.
14. E. Rubiera, L. Polo, D. Berrueta, and A. E. Ghali. Telix: An rdf-based model for linguistic annotation. In *ESWC*, 2012.
15. M. Schierle. *Language Engineering for Information Extraction*. Phd thesis, Universität Leipzig, 2011.
16. S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, 2012.
17. S. Tobies. *Complexity results and practical algorithms for logics in knowledge representation*. PhD thesis, TU Dresden, 2001.