NIF Combinator: Combining NLP Tool Output

Sebastian Hellmann[1], Jens Lehmann[1], Sören Auer[2], Marcus Nitzschke[1]

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

Abstract. The NLP Interchange Format (NIF) is an RDF/OWL-based format that provides interoperability between Natural Language Processing (NLP) tools, language resources and annotations by allowing NLP tools to exchange annotations about text documents in RDF. 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. NIF wrappers have to be only created once for a particular tool and can subsequently interoperate with a potentially large number of other tools. We present (1) the currently implemented NIF wrappers, which are available as free web services and (2) a GUI called the NIF Combinator, which allows to combine the output of the implemented NIF web services.

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 *Named Entity Recognition* (NER) 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, as the programming interfaces and result formats often differ to a great extent. Also, once a particular set of tools is integrated, this integration is usually *not reusable* by others. 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 and implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist; a public web demo, the NIF Combinator, is available at `http://nlp2rdf.lod2.eu/demo.php` NIF-aware applications will produce output (and possibly also consume input) adhering to the NIF URI Scheme and the String Ontology as REST services (access layer). Other than more centralized solutions such as UIMA[3] and GATE[4], NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. Another benefit is that a NIF wrapper has to be only created once for a particular tool, but enables the tool to interoperate with a potentially large number of other tools without additional adaptations. Ultimately, we envision an ecosystem of NLP tools and services to emerge using NIF for exchanging and integrating rich annotations. This paper describes the accompanying demo for [2].

NIF Example NIF provides two URI schemes, which can be used to represent strings as RDF resources. In the example below, "Berlin" in the sentence "Berlin has a new mayor!" was annotated by two different tools, a part of speech tagger using an OLiA identifier and an entity linking tool connecting the string with DBpedia. In this case, a simple string offset based URI scheme was used [2].

```
@prefix : <http://prefix.given.by/theClient#> .
@prefix str: <http://nlp2rdf.lod2.eu/schema/str/> .
@prefix sso: <http://nlp2rdf.lod2.eu/schema/sso/> .
#the whole sentence is given a URI and typed as context
:offset_0_23  a    str:Context , sso:Sentence;
    str:isString   "Berlin has a new mayor!" .
#a substring is given a URI and is annotated
:offset_0_6   a    str:StringInContext, sso:Word ;
    str:referenceContext     :offset_0_23 ;
#part of speech annotation
    sso:oliaLink   <http://purl.org/olia/penn.owl#ProperNoun> ;
#link to dbpedia
    sso:oen                  dbpedia:Berlin .
```

2 NIF Wrappers and Combinator

Access via REST Services: The structural and conceptual interoperability layers of NIF are built upon the RDF standard, Linked Data principles and existing ontologies such as OLiA. To improve interoperability and accessibility of NIF components, NIF provides a normative *access layer*, which facilitates easier integration and off-the-shelf solutions by specifying REST parameters. Of special importance is the *prefix* parameter as it enables the client to influence the RDF output. The RDF in Figure 3 is produced by different tools, but can be merged directly under the condition that the URI prefixes and offsets are the same.

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. The NIF Combinator shows one possible implementation of such a workflow. 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

[3] http://uima.apache.org/
[4] http://gate.ac.uk/

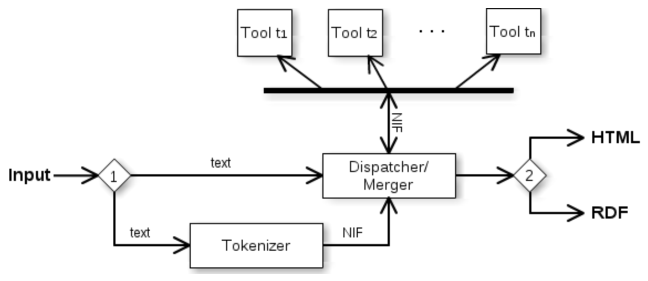


Fig. 1. Overview of the NIF Combinator workflow.

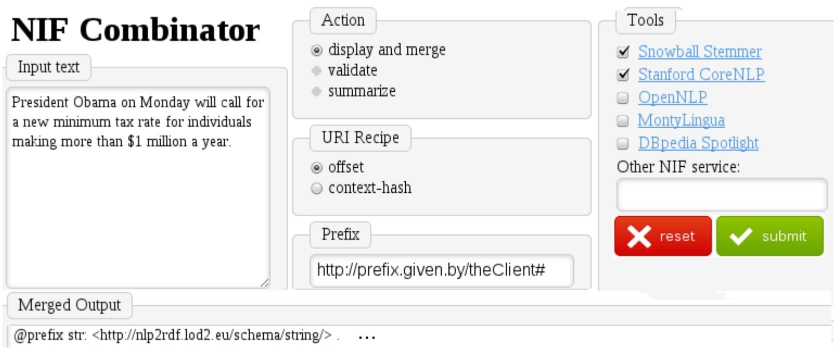
added without using additional formalisms. For acquiring and merging external data from knowledge bases, existing Semantic Web tools can be used.

The main **interface** are wrappers that provide NIF web services. A NIF web service must be *stateless*, *HTTP method agnostic respective POST and GET* 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.
– *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`). Enables NIF output for existing web services (i.e. deploy NIF in parallel to legacy output).
– *Format* (optional, `format = rdfxml | ntriples | n3 | turtle | json`). The RDF serialisation format.
– *Prefix* (optional, `prefix = uriprefix`). An 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).

NIF Combinator Demo: Figure 1 describes the workflow of the NIF Combinator. The given input (normally text) can be forwarded directly to the dispatcher or optionally prepared by a tokenizer (see Diamond 1 in Figure 1). The tokenizer already outputs the results in NIF and provides tokenization for the remaining components. The dispatcher then calls the selected NLP tools (see checkboxes in Figure 2) which can read as well as write NIF. The NIF output from all the tools is then merged (see Figure 3). Merged results can be shown in HTML format for users. Another option (Diamond 2) is to output RDF directly. This way, the NIF Combinator can be used as an aggregator web service itself, by simply executing a GET/POST request with the parameters of the HTML forms.

Conclusions and Future Work: In this demo paper, we briefly presented NIF wrappers and the NIF Combinator tool. All implemented NIF wrappers are able to produce NIF output adhering to the NIF 1.0 specification. The additional integration of a tokenizer has, however, not yet been fully implemented for all



NIF Combinator

Input text

President Obama on Monday will call for a new minimum tax rate for individuals making more than $1 million a year.

Action: [✓ display and merge] validate summarize

URI Recipe: [✓ offset] context-hash

Prefix: http://prefix.given.by/theClient#

Tools: [✓ Snowball Stemmer] [✓ Stanford CoreNLP] OpenNLP MontyLingua DBpedia Spotlight

Other NIF service:

Merged Output

```
@prefix str: <http://nlp2rdf.lod2.eu/schema/string/> .   ...
```

Fig. 2. Screenshot of the NIF Combinator user interface.

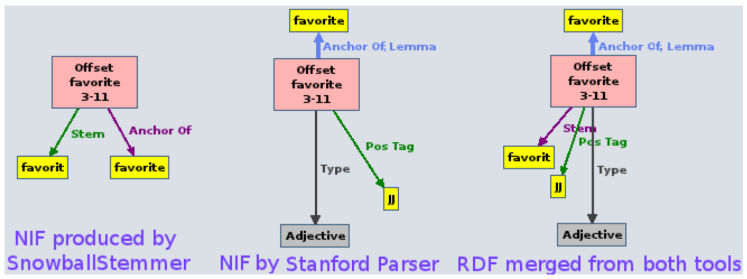


Fig. 3. Example of merged RDF from two NLP tools.

wrappers. Tokenization conflicts will be resolved either by providing tokenization for the tools or by implementing resolution strategies for tokenization conflicts[1] in the dispatcher/merger component. Future work comprises the creation of a new version NIF 2.0 based on community feedback. All presented resources are openly available at `http://nlp2rdf.org`

Acknowledgments We would like to thank our colleagues from AKSW research group and the LOD2 project for their helpful comments during the development of NIF. Especially, we would like to thank Christian Chiarcos for his support while using OLiA and the students that helped implement the wrappers: Markus Ackermann, Martin Brümmer, Didier Cherix, 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).

References

1. C. Chiarcos, J. Ritz, and M. Stede. By all these lovely tokens... merging conflicting tokenizations. *LRE*, 46(1):53–74, Mar. 2012.
2. S. Hellmann, J. Lehmann, and S. Auer. Linked-data aware uri schemes for referencing text fragments. In *EKAW 2012*, LNAI. Springer, 2012.