1 Introduction

This document describes our implementation of RESTful NLP web services conforming to the NLP Interchange Format (NIF).
„NIF is an RDF/OWL-based format that aims to achieve interoperability between NLP tools language resources and annotations.“
We implemented NIF wrappers for the Stanford POS tagger[1] and Stanford parser[2] to demonstrate such integration.

2 The NLP Interchange Format

NIF Core Ontology

NIF is an RDF-based format. The classes to represent linguistic data are defined in the NIF Core Ontology[3].
All ontology classes are derived from the main class nif:String which respresents strings of Unicode characters.
One important subclass of nif:String is nif:Context. It represents a text in its entirety and holds the characters of this text in the nif:isString property. There are several classes (e.g. nif:Word, nif:Phrase, nif:Sentence) for representing partitions of a text, their choice depends on the unit of annotation. All such subunits have a property nif:referenceContext pointing to their respective nif:Context instance.
Furthermore, their position inside the context is specified using the nif:beginIndex and nif:endIndex properties. The actual substring, which these units represent can be specified using the nif:anchorOf property.
Annotations like POS tags or relation types (see below) can be added as properties to the respective individual.
NIF individuals are identified by URIs following a nif:URIScheme which restricts the URI's syntax. E.g. a URI following RFC 5147 consists of a prefix string followed by „#char=x,y“, where x and y are the start and end positions of the string in its context. For nif:Context URIs y can be omitted or set to the total number of characters in the text.

NIF service parameters

Our web services conform to the NIF 2.0 public API specification[4].
The following parameters are supported by a specification compliant service:

- input/i: The input to be processed by the service.
- informat/f: The format in which the input is given. Supported argument values are *text*, *turtle* (default) and *json-ld*.
- intype/t: Specifies how the input is retrieved. Supported argument values are *direct* (default), *file* and *url*.
- outformat/o: The format in which the output will be serialized. Supported argument values are *turtle* (default) and *json-ld*.
- urischeme/u: the URI scheme the service must use to create new URIs
- prefix/p: the service must use this as the prefix part of new URIs. A UUID will be generated if no prefix is specified

1 http://vtentacle.techfak.uni-bielefeld.de/~bsiemone/index.php/NifStanfordPOSTagger
2 http://vtentacle.techfak.uni-bielefeld.de/~bsiemone/index.php/NifStanfordParser
3 http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html
4 http://persistence.uni-leipzig.org/nlp2rdf/specification/api.html

Furthermore, our services allow to specify the path/url of a trained model to be used by the service via the option „m“. If such a model file is not available the input language can be specified using the parameter “l”. A corresponding model will then be selected by the service. Supported arguments are *english*, *french*, and *german*.

3 NIF service implementations

NIF wrapper for the Stanford POS tagger

Our web service wrapping the Stanford POS tagger can be invoked via curl using the following example call.

Example call 1:

```
curl vtentacle.techfak.uni-bielefeld.de/~bsiemone/index.php/NifStanfordPOSTagger
-d f="text" -d i="This is a sample sentence"
```

If the input is given as plain text like in the above example, an RDF model is constructed containing a nif:Context element with the input text in its nif:isString property and one nif:Word element for each word in the input.
If the input is already in NIF format, it is expected to contain at least one nif:Context element. Right now all input elements except the instances of nif:Context are ignored. It would be possible to tag instances of nif:Sentence or sets of nif:Word though.

The service then reads the nif:isString values of all nif:Context elements found in the input and passes them to the Stanford NLP tools where it it tokenized and POS tagged.
Each word is annotated by adding a nif:posTag property with the POS tag as a literal value to the corresponding nif:Word element. If a nif:Word instance is missing in the input it will be created.

Ouput of example call 1:

```
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

<936ea662-8613-4aae-968e-166715c4d529#char=10,16>
        a                       nif:RFC5147String , nif:Word ;
        nif:anchorOf            "sample" ;
        nif:beginIndex          "10"^^xsd:nonNegativeInteger ;
        nif:endIndex            "16"^^xsd:nonNegativeInteger ;
        nif:posTag              "NN"^^xsd:string ;
        nif:referenceContext    <936ea662-8613-4aae-968e-166715c4d529#char=0,25> .

<936ea662-8613-4aae-968e-166715c4d529#char=0,25>
        a               nif:Context , nif:RFC5147String , nif:String ;
        nif:isString    "This is a sample sentence" .

<936ea662-8613-4aae-968e-166715c4d529#char=17,25>
        a                       nif:RFC5147String , nif:Word ;
        nif:anchorOf            "sentence" ;
        nif:beginIndex          "17"^^xsd:nonNegativeInteger ;
        nif:endIndex            "25"^^xsd:nonNegativeInteger ;
        nif:posTag              "NN"^^xsd:string ;
        nif:referenceContext    <936ea662-8613-4aae-968e-166715c4d529#char=0,25> .

<936ea662-8613-4aae-968e-166715c4d529#char=8,9>
        a                       nif:RFC5147String , nif:Word ;
        nif:anchorOf            "a" ;
        nif:beginIndex          "8"^^xsd:nonNegativeInteger ;
        nif:endIndex            "9"^^xsd:nonNegativeInteger ;
        nif:posTag              "DT"^^xsd:string ;
        nif:referenceContext    <936ea662-8613-4aae-968e-166715c4d529#char=0,25> .

<936ea662-8613-4aae-968e-166715c4d529#char=0,4>
        a                       nif:RFC5147String , nif:Word ;
        nif:anchorOf            "This" ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "4"^^xsd:nonNegativeInteger ;
        nif:posTag              "DT"^^xsd:string ;
        nif:referenceContext    <936ea662-8613-4aae-968e-166715c4d529#char=0,25> .

<936ea662-8613-4aae-968e-166715c4d529#char=2,4>
        a                       nif:RFC5147String , nif:Word ;
        nif:anchorOf            "is" ;
        nif:beginIndex          "2"^^xsd:nonNegativeInteger ;
        nif:endIndex            "4"^^xsd:nonNegativeInteger ;
        nif:posTag              "VBZ"^^xsd:string ;
        nif:referenceContext    <936ea662-8613-4aae-968e-166715c4d529#char=0,25> .
```

NIF wrapper for the Stanford dependency parser

Our web service wrapping the Stanford dependency parser can be invoked via curl using the following example call where the input is assumed to be given in a turtle file called input.tll.

Example call 2:

```
curl vtentacle.techfak.uni-bielefeld.de/~bsiemone/index.php/NifStanfordParser
i="input.ttl"
```

The service can be used to parse input that is already POS tagged. I.e. it expects the input to be in NIF format and contain
a) at least one nif:Context element
b) one nif:Word element for each word in the nif:isString property of its context containing a POS annotation in nif:posTag and the represented string in nif:anchorOf.

The words are ordered by context (using nif:referenceContext) and position (using nif:beginIndex) in order to reconstruct the original texts.
The service then passes the annotated input to the Stanford parser. For each dependency relation of the parse a nif:dependency property is added to the relation's head with the URI of the dependent word as object. As a word can only have one head, the type of the relation is annotated in the nif:dependencyRelationType property of the dependent word (as a literal).

Output of example call 2:

```
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=17,25>
        a                           nif:Word , nif:RFC5147String ;
        nif:anchorOf                "sentence" ;
        nif:beginIndex              "17"^^xsd:nonNegativeInteger ;
        nif:dependencyRelationType  "dep"^^xsd:string ;
        nif:endIndex                "25"^^xsd:nonNegativeInteger ;
        nif:posTag                  "NN"^^xsd:string ;
        nif:referenceContext        <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25> .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=8,9>
        a                           nif:Word , nif:RFC5147String ;
        nif:anchorOf                "a" ;
        nif:beginIndex              "8"^^xsd:nonNegativeInteger ;
        nif:dependencyRelationType  "det"^^xsd:string ;
        nif:endIndex                "9"^^xsd:nonNegativeInteger ;
        nif:posTag                  "DT"^^xsd:string ;
        nif:referenceContext        <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25> .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25>
        a             nif:String , nif:RFC5147String , nif:Context ;
        nif:isString  "This is a sample sentence" .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,4>
        a                           nif:Word , nif:RFC5147String ;
        nif:anchorOf                "This" ;
        nif:beginIndex              "0"^^xsd:nonNegativeInteger ;
        nif:dependencyRelationType  "dep"^^xsd:string ;
        nif:endIndex                "4"^^xsd:nonNegativeInteger ;
        nif:posTag                  "DT"^^xsd:string ;
        nif:referenceContext        <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25> .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=2,4>
        a                     nif:Word , nif:RFC5147String ;
        nif:anchorOf          "is" ;
        nif:beginIndex        "2"^^xsd:nonNegativeInteger ;
        nif:dependency        <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,4> ,
<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=17,25> , <61504c14-bcdf-4f84-b24f-
42ef77bec7be#char=10,16> ;
        nif:endIndex          "4"^^xsd:nonNegativeInteger ;
        nif:posTag            "VBZ"^^xsd:string ;
        nif:referenceContext  <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25> .

<61504c14-bcdf-4f84-b24f-42ef77bec7be#char=10,16>
        a                           nif:Word , nif:RFC5147String ;
        nif:anchorOf               "sample" ;
        nif:beginIndex              "10"^^xsd:nonNegativeInteger ;
        nif:dependency              <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=8,9> ;
        nif:dependencyRelationType  "nsubj"^^xsd:string ;
        nif:endIndex                "16"^^xsd:nonNegativeInteger ;
        nif:posTag                  "NN"^^xsd:string ;
        nif:referenceContext        <61504c14-bcdf-4f84-b24f-42ef77bec7be#char=0,25> .
```

Chaining NIF services

As one of the services described above (the tagger) produces output the other one (the parser) relies on, they can be used to demonstrate the integration of NIF compliant NLP services.
The following nested call combines example calls 1 and 2. It invokes the tagger which produces the output of example call 1 and passes this POS annotated NIF data to the parser. The output is the same as in example call 2.

Example call 3:

```
curl vtentacle.techfak.uni-bielefeld.de/~bsiemone/index.php/NifStanfordParser -d
i=$(curl vtentacle.techfak.uni-
bielefeld.de/~bsiemone/index.php/NifStanfordPOSTagger -d f="text" -d i="This is
a sample sentence")
```