NAF: the NLP Annotation Format

Technical Report NWR-2014-3

Version 1.1

Antske Fokkens,[1] Aitor Soroa,[2] Zuhaitz Beloki,[2] German Rigau,[2]
Willem Robert van Hage[3] and Piek Vossen[1]

(1) The Network Institute, VU University Amsterdam
Amsterdam, The Netherlands
{antske.fokkens,piek.vossen}@vu.nl
(2) IXA NLP group, University of the Basque Country UPV/EHU
Donostia, Basque Country
{a.soroa, zuhaitz.beloki, german.rigau}@ehu.es
(3) Inovation Lab, SynerScope
Horsten 1, 5612 AX Eindhoven
willem.van.hage@synerscope.com

Building structured event indexes of large volumes of financial and economic data for decision making
ICT 316404

1 Introduction

The NLP Annotation Format (NAF) is designed to represent linguistic annotations in complex NLP architectures. It follows the recommendations set out by Ide *et al.* (2003) for the Linguistic Annotation Framework (LAF). Because of its layered extensible format, it can easily be incorporated in a variety of NLP modules that may require different linguistic information as their input. NAF furthermore follows the footsteps of the NLP Interchange Format (Hellmann *et al.*, 2013, NIF) in making its representations conform to the principles of Linked Data as much as possible. This property facilitates communication between resources represented in Linked Data and NLP modules, both while making use of external resources in NLP modules and while analyzing text and represent retrieved information as Linked Data. Representing information in the Resource Description Framework (Manola and Miller, 2004, RDF) has the additional advantage that we can use Semantic Web technology to search and link information represented in NAF and offers a straightforward way to model provenance.

Using existing formats as a basis to define NAF has the additional advantage that it is (relatively) easy to integrate information represented in these formats in NAF. NAF is used in projects involving tools developed at different sites. These tools are intended to be used in future projects and are available to third parties as well. It is therefore essential that they can easily be integrated in various structures. We therefore looked at formats used in other projects such as the Knowledge Annotation Format (Bosma *et al.*, 2009, KAF), used in KYOTO[1] and OpeNER,[2] among others) and the Terence Annotation Format (Moens *et al.*, 2011, TAF), used in TERENCE.[3] Especially KAF contributed much to the basis of NAF.

In this report, we describe NAF. We elaborate on the motivation for designing the format, provide a detailed overview of the desiderata of a standardized format for linguistic representation and explain how NAF fulfills these requirements. The report is structured as follows. Section 2 will elaborate on the background and motivation behind NAF. This section will also present the desiderata of the framework. A global overview of the main ideas behind NAF and how this approach fulfills our desiderata will be given in Section 3. This is followed by a description of the NAF-RDF (the RDF version of NAF) in Section 4. We then describe the NewsReader architecture as an example of how NAF is used in an architecture involving several NLP tasks. Finally, we conclude and present future work in Section 6. The current

[1] `http://kyoto-project.eu/xmlgroup.iit.cnr.it/kyoto/index.html`
[2] `http://www.opener-project.org/`
[3] `http://terenceproject.eu/web/guest`

information layers are described in the appendix of this report.

2 Background and Motivation

Research involving computational linguistics and Linked Data has increased in popularity. The Semantic Web community is looking into NLP to include information from text to the Semantic Web. At the same time, more and more NLP applications make use of Linked Data. These research directions call for a format that both meets the requirements posed by NLP modules and can be interpreted as Linked Data. The main motivation behind NAF was to create a format that has these properties, but is close enough to formats already used in NLP tools to facilitate conversion and integration of these tools. It was designed in the context of NewsReader[4] and BiographyNet,[5] two projects that involve both research directions where Linked Data and NLP meet. They thus provide a clear context to define requirements for linguistic representations that facilitate interaction between NLP and Linked Data. We will mainly use NewsReader as a use case for explaining decisions on NAF's design throughout this document.

The remainer of this section provides a more elaborate overview of the motivation behind NAF. It is structured as follows. First, we briefly describe the main goals of NewsReader and how they influenced the requirements for NAF. This is followed by an overview of requirements posed and fulfilled in related work. Previously formulated requirements together with the requirements posed by recent projects result in a list of desiderata.

2.1 Desiderata

In NewsReader, we work on a "history recorder" that follows the daily newsstreams in English, Spanish, Italian and Dutch. We aim to determine what happend to whom, when and where, removing duplication, complementing information, registering inconsistencies and keeping track of original sources. Incoming information is integrated with data from the past, distinguishing new information from old and unfolding storylines. Extracted information is represented in RDF triples and stored in a central repository called KnowledgeStore (KS), which also performs automated reasoning to derive new facts from asserted ones. A decision-support tool visualizes storylines exploiting their explanatory power as well as their structural implications.

[4] `http://www.newsreader-project.eu/`
[5] `http://www.biographynet.nl/`

In order to follow (all) the news in four languages, massive processing is required involving a range of NLP technologies. Ideally, existing state-of-the-art tools should be integrated in the NewsReader pipeline where possible. Moreover, whenever new tools are developed, they should be easily integrated in other systems so that they may be used in other projects of members of the NewsReader consortium as well as third parties. Those requirements clearly suggest using a common representation format that guarantees the correct interoperability among the modules.

KAF has shown to be suitable for a complex pipeline combining tools developed at different sites in the KYOTO project and OpeNER. For more recent projects such as NewsReader, however, some additional properties are required. First, as explained above, within NewsReader we aim to extract storylines and store them as RDF triples in the KS. Using rich representation formalisms such as RDF in this step is necessary as it allows the KS to perform reasoning. The information in the KS will also be used to support linguistic processing (e.g. for disambiguation). Second, we have more than one tool for several of the steps in the pipeline. Ideally the output of these tools should be combined. Third, we delay decisions as much as possible. That is, instead of only providing the output that received the highest score, we will include several possible outcomes per tool with their confidence scores. Fourth, NewsReader requires processing of massive amounts of data. In KAF, the same information can be repeated within a representation, leading to (partially) unnecessary increase in size which may become problematic when large amounts of data are tackled. Such repetitions are therefore avoided in NAF.

Based on these requirements, we define the following properties and desiderata for NAF:

1. NAF should properly represent linguistic information focusing on two kind of linguistic processes (LPs):
   (a) within document processing: LPs whose granularity is the document
   (b) cross document processing, for (event) coreference, etc.
2. NAF should be simple
3. NAF should work for existing NLP modules developed by the partners in NewsReader, i.e. it should be easy and little afford to adapt existing tools to use NAF.
4. All elements in NAF will be identified with URIs (not document/XML-object internal ids)

LAF requirements	NIF requirements
Expressive adequacy	Compatibility with RDF
Media independence	Coverage
Semantic adequacy	Structural interoperability
Incrementality	Conceptual interoperability
Uniformity	Granularity
Openness	Provenance and Confidence
Extensibility	Simplicity
Human readability	Scalability
Processability	
Consistency	

Table 1: Requirements defined for LAF and NIF.

5. NAF should be flexible so that it can contain additional information and alternative representations:
   (a) It should be possible (and preferably easy) to integrate alternative modules (that may be developed by third parties) in the pipeline
   (b) It should be possible to represent other RDF-based layers.

The following section describes LAF and NIF a framework and, respectively, format that each fulfill some of the desiderata.

2.2 LAF + NIF = NAF

As the number of available NLP tools increases and they are used in more and more complex architectures, awareness of the importance of standardization rises (Ide *et al.* (2003), Bosma *et al.* (2009), Hellmann *et al.* (2013), among others). One of the main challenges lies in the fact that linguistic annotations as well as the output of NLP tools can be based on different theories or insights which each may have their own strengths. Standardization efforts must therefore bring these variations together without compromising the richness of the individual output of different tools.

This section will explain how LAF-based formats and NIF support standardization. We will outline the strengths of both approaches and provide a first indication of how NAF takes advantage of them. Table 1 lists the requirements defined for LAF and NIF. In fact, NAF fulfills all of these requirements as we will explain in the remainder of this report. We will mark properties of NAF related to individual requirements in **bold font** in this section.

NAF is based on Kyoto Annotation Format (Bosma *et al.*, 2009, KAF) as mentioned above. This format follows the main principles of LAF as outlines in Ide *et al.* (2003). Like LAF, KAF aims at **maximum flexibility**, **processing efficiency** and **reusability**. It is a layered, **extensible** format where each tool **incrementally** adds its output while maintaining all information that was present in its input (Bosma *et al.*, 2009).

NAF follows KAF in maintaining these properties, but is inspired by NIF to meet three of the four missing requirements in KAF defined above. These requirements are producing RDF triples and using information presented in RDF as part of the NLP pipeline, combining output of alternative tools and delaying decisions where possible. These three desiderata can be addressed by using **RDF conform** representations as shown by NIF. NIF is a RDF compliant format for linguistic information that is designed to accommodate the constantly increasing wide variety of NLP tools (Hellmann *et al.*, 2013) and can include information on provenance and confidence. Both provenance and confidence score indications are essential when combining the output of different tools.[6] A drawback of NIF is that it does not seem to be a practical format for internal use of NLP tools. This assumption is confirmed by Hellmann *et al.* (2013)'s own user evaluation of the format.

NAF combines the strengths of KAF and NIF by using **Uniform Resource Identifiers** as much as possible in a representation that in other aspects follows the LAF recommendations. It is suitable to be used in NLP tools as is shown by the fact that we produced a wide variety of NLP modules using NAF that form an elaborate pipeline for event extraction and coreference in a relatively short time. At the same time, it offers the means to combine outcome of alternative tools while indicating **provenance and confidence scores**, as also offered by NIF. We even take the idea of using RDF conform representations a step further than NIF and also encourage to use URIs to refer to linguistic properties and values. The next subsection will elaborate on the main advantages of this extensive use of RDF compliant representations.

2.3 RDF in linguistic representations

RDF is a useful data model for NAF due to several reasons. This section will list the main reasons and explain how they support the desiderata outlined in the previous section.

First, RDF is by nature a graph model, which makes declarative speci-

[6]The full version of this paper will contain a section addressing the importance of modeling provenance and how this is achieved in NAF.

fication of dependency patterns easy, for instance in SPARQL. Triple stores are typically optimized for queries that require multiple joins. That makes evaluation of dependency graph queries, which are typically long branched chains, efficient. This facilitates the communication between the KS and linguistic processing tools.

Second, RDF uses URIs for identification and URIs are not limited to the scope of a document, but have a global validity. This makes it easy to represent coreference relations across documents as done in the Grounded Annotation Framework (Fokkens *et al.*, 2013, GAF). In GAF, formal representations of instances can be linked to one or more mentions of this instance in text, hence indicating which mentions corefer to this instance. A similar approach is taken to model the relations between instances: e.g., we can indicate that a semantic role label between the mention of a participant and the mention of an event is the mention of the relation between the event and this participant.

Third, RDF forms the basis on which RDFS and OWL ontology reasoning is possible. This allows for some very useful operations, such as subclass, subproperty and property chain reasoning. This last property forms the main motivation to use URIs more extensively than is done in NIF. Schuurman and Windhouwer (2011) note the challenges involved in defining standardized sets of linguistic properties. ISOcat provides standards with useful definition, but because of differences in linguistic theory or properties it is not always possible to use existing sets. New, sometimes closely related, categories will be introduced as linguistic annotations. If we can represent linguistic properties as ontologies, we can define how output of different tools relate to each other. If, for instance, there are differences in **granularity** between output of different tools, it can be used to generalize over linguistic information (e.g., NNS $\subseteq$ NN $\subseteq$ NP). It is also possible to define equivalence or near equivalence. RELcat (Schuurman and Windhouwer, 2011; Windhouwer, 2012) provides a set of basic relations specifically designed for this purpose. We can make use of these relations in NAF. If, for instance, users want to use URIs that refer to ISOcat definitions that are still under construction, they can define their own tool-specific ontology of properties and values that can be linked to the ISOcat standards later on.

It should be noted that NAF strongly encourages the use of URIs, but does not enforce it. New layers of information can thus be integrated in NAF representations easily by using strings to represent properties and values. Ontologies that support the full reasoning and comparison advantages of RDF can be defined at a larger stage, when the need to compare or combine the information rises. The following section will describe the general structure of NAF.

3 The general structure of NAF

In this section, we will provide a more detailed description of what NAF look like. NAF comes in two forms: "standard NAF" which is (typically) used to pass information through NLP modules and NAF-RDF a highly similar variation of NAF that conforms completely to RDF principles. We will first describe standard NAF, which is followed by a description of the differences found in NAF-RDF and a brief explanation of why these two are separate.

The following sections provide general properties of NAF and a basic description of some of the main layers of linguistic information. The strong ties between KAF and NAF have been mentioned at various places in this document. In these sections, we will highlight commonalities and differences between NAF and KAF. Descriptions of properties shared by the two formats based on Bosma *et al.* (2009).

3.1 General properties of NAF

Like KAF, NAF comprises several annotations over a text at different linguistic levels (morphosyntactic, syntactic, semantic) and adopts a stand off strategy for annotating the source text. The following general rules are followed in all layers:

- `<span>` elements are used for grouping linguistic elements.
- Linguistic annotations of a particular level always span elements of previous levels.
- Linguistic annotations of different levels are not mixed.

The "levels" in the general rules refer to different types of linguistic information, which can be different groupments of linguistic entities (e.g. tokens vs. terms vs. chunks) or relations between linguistic entities (e.g. dependencies, semantic roles) or information about a specific linguistic entity (e.g. disambiguated word sense). The most basic level in NAF is the text layer which assigns identifiers to tokens in the text. The term layer defines basic terms which can consist of one or more tokens, chunks typically consist of one or more terms (but can in principle also consist of tokens), etc. The span elements are used to refer to specific elements in the other layer, i.e. the ids of the tokens that make up a specific term define the span of the term. Relations are defined between elements of another layer, e.g. dependency relations between terms. In relations, the span consists of a source

and a target. Section 3.2 will provide more information on some of the basic components of NAF and illustrate how they are represented with basic examples.

In order to reduce unnecessary duplication of information and facilitate conversion to NAF-RDF, the following additional rules were defined for NAF:

- Fixed properties of a specific linguistic annotation are not repeated if the annotation occurs more than once in a NAF representation.
- The structure of different linguistic layers should be consistent.
- URIs should be used whenever possible to refer to linguistic properties.

Some effort was made to use a consistent schema within KAF and Bosma *et al.* (2009) explicitly looked at possibilities to integrate existing standards for linguistic annotation, including some that can be represented by URIs. However, these three rules were not standardly respected in the design of KAF.

For instance, information on sentiment consists of various properties (resource, polarity, strength, subjectivity, etc.), all this information tends to be the same for a term with a specific meaning. Everytime this term occurs with this meaning in the text, this information is repeated in KAF. Furthermore, the resource tends to be stable for a given NLP module. The representation can thus be made more concise if we add information about the resource to the general provenance layer of the module and create assign group all stable information under a specific sentiment value. The individual terms only need to point to the sentiment value which leads to the specific properties of the term.

Consistency is important so that generic rules can be used to convert standard NAF to NAF-RDF. The `<span>` element should be used to point to other linguistic entities in the NAF file and `<source>` and `<target>` should be used define a relation from one linguistic entity (the source) to another (the target). Coreference (which can be seen as a bidirectional relation between terms) is modeled by relating the terms that corefer to the same entity.

Finally, URIs should be used as much as possible so that we can make use of the advantages of RDF conform representations, as outlined above. We will illustrate the rules outlined above through some of the basic layers of NAF in the next subsection.

3.2 Principle Levels of NAF

In this section, we illustrate how NAF works through basic descriptions of some of the mostly used levels of NAF. The descriptions are kept general

and examples simple. The NAF website[7] provides information on current NLP modules using NAF, as well as information on the current status of NAF. It should be noted that the examples illustrated here first and foremost illustrate a general annotation format for linguistic information following the principles of LAF.

The <text> layer can be seen as the most basic level in NAF. NAF also includes a level where the raw text can be represented (<raw>), but the linguistic entity that is represented in this level generally does not function as a span in other levels. The <text> layer provides the tokens of the text and assigns them an identifier. In addition, offset and length of the token are indicated and optionally the sentence or paragraph. Figure 1 provides an example of a text layer.

```
<text>
  <wf id="w1" offset="0" length="4" sent="1" para="1">John</wf>
  <wf id="w2" offset="5" length="6" sent="1" para="1">taught</wf>
  <wf id="w3" offset="12" length="11" sent="1" para="1">mathematics</wf>
  <wf id="w4" offset="24" length="2" sent="1" para="1">20</wf>
  <wf id="w5" offset="27" length="7" sent="1" para="1">minutes</wf>
  <wf id="w6" offset="35" length="5" sent="1" para="1">every</wf>
  <wf id="w7" offset="41" length="6" sent="1" para="1">Monday</wf>
  <wf id="w8" offset="48" length="2" sent="1" para="1">in</wf>
  <wf id="w9" offset="51" length="3" sent="1" para="1">New</wf>
  <wf id="w10" offset="55" length="3" sent="1" para="1">York</wf>
  <wf id="w11" offset="59" length="1" sent="1" para="1">.</wf>
</text>
```

Figure 1: Basic example of a text layer

The second most basic level is the terms layer. Terms span over one or more tokens defined in the <text> layer. The <target> of the indicates which tokens are part of the term. Information can be added indicating whether the term is an open or closed class term, the Part-of-Speech of the term, its lemma, morphological features, case or which of the tokens makes up the head of the term (in multiword expressions). Figure 2 provides a basic example of a term layer.

We will provide examples of two more complex levels to illustrate how NAF works. The first example represents linguistic chunks (the <chunks> layer). Chunks span one or more terms forming a syntactic phrase (integrated in the structure as the <target> of a span). A chunk always has a head. The term that makes up the head of the chunk must also be included in the span. The chunk layer can furthermore optionally indicate the type of phrase and case value of the phrase. Even though we currently only have modules

[7] http://wordpress.let.vupr.nl/naf/

```
<terms>
  <term id="t1" lemma="John" pos="R">
    <span>
      <target id="w1"/>
    </span>
  </term>
  <term id="t2" type="open" lemma="teach" pos="V">
    <span>
      <target id="w2"/>
    </span>
  </term>
  <term id="t3" lemma="mathematics" pos="N">
    <span>
      <target id="w3"/>
    </span>
  </term>
  <term id="t4" lemma="20" pos="N">
    <span>
      <target id="w4"/>
    </span>
  </term>
  <term id="t5" lemma="minute" pos="N">
    <span>
      <target id="w5"/>
    </span>
  </term>
  <term id="t5" lemma="every" pos="D">
    <span>
      <target id="w6"/>
    </span>
  </term>
  <term id="t6" lemma="Monday" pos="N">
    <span>
      <target id="w7"/>
    </span>
  </term>
  <term id="t7" lemma="in" pos="P">
    <span>
      <target id="w8"/>
</span>
</term>
  <term id="t.mw8" lemma="New_York" pos="R">
    <span>
      <target id="w9"/>
      <target id="w10"/>
    </span>
  </term>
</terms>
```

Figure 2: Basic example of a terms layer

running NAF that build chunks from terms, it is also possible to have chunks that span tokens or other chunks. In principle, the chunk layer can be used to define a phrase structure tree. An example of a chunks layer is given in Figure 3.

The final representation layer we will present as part of this illustration

```
<chunks>
  <!-- John -->
  <chunk id="c1" head="t1" phrase="NP">
    <span>
      <target id="t1"/>
    </span>
  </chunk>
  <!-- taught -->
  <chunk id="c2" head="t2" phrase="V">
    <span>
      <target id="t2"/>
    </span>
  </chunk>
  <!-- Mathematics -->
  <chunk id="c3" head="t3" phrase="NP">
    <span>
      <target id="t3"/>
    </span>
  </chunk>
  <!-- 20 minutes -->
  <chunk id="c5" head="t5" phrase="NP">
    <span>
      <target id="t4"/>
      <target id="t5"/>
    </span>
  </chunk>
  <!-- every -->
  <chunk id="c6" head="t6" phrase="R">
<span>
      <target id="t6"/>
    </span>
  </chunk>
  <!-- every Monday -->
  <chunk id="c7" head="t7" phrase="NP">
    <span>
      <target id="t6"/>
      <target id="t7"/>
    </span>
  </chunk>
  <!-- in New York -->
  <chunk id="c9" head="t9" phrase="PP">
    <span>
      <target id="t8"/>
      <target id="t9"/>
    </span>
  </chunk>
</chunks>
```

Figure 3: Basic example of chunks

is the semantic role layer (`<srl>`). In addition to the usual `<span>` element (which typically points to a term), semantic role labels may include references to external resources. The reference attribute points to the URI of a possible interpretation of the term, where reftype provides the possibility of indicating a more general class of this reference. Resource indicates the general resource that provided the general resource. The `<role>` element is

```
<srl>
  <predicate id="pr1" uri="http://framenet.net/Make" confidence=0.9>
    <!-- making -->
     <externalReferences>
        <externalRef reference="cognition" reftype="mcr:cognition" resource="2"/>
        <externalRef reference="cognition" reftype="mcr:cognition" resource="mc"/>
        <externalRef reference="eng-30-00690614-v" resource="wn:eng-30"/>
      </externalReferences>
    <span>
      <target id="t3"/>
    </span>
    <role id="rl1" semRole="Agent">
      <!-- She -->
      <externalReferences>
        <externalRef reference="participant" resource="1"/>
      </externalReferences>
      <span>
<target id="t1"/>
      </span>
    </role>
    <role id="rl2" semRole="Theme">
      <!-- apple jam -->
      <span>
<target id="t4"/>
<target id="t5"/>
      </span>
    </role>
  </predicate>
</srl>
```

Figure 4: Basic example of semantic roles

used to indicate which terms are related to the predicate, where the semRole attribute provides the semantic role as assigned by the NLP module used for semantic role labelling. External references can also be used in these elements to indicate corresponding roles found in other resources.

4 NAF-RDF

The structures presented in the previous section are not RDF representations. As explained above, it is advantageous to also have a variation of NAF that in RDF, because we can take advantage of Semantic Web technologies to extract information from our representations efficiently. Furthermore, communication with information represented in RDF (as is the case in the KnowledgeStore in NewsReader) is facilitated when we have representations that represent all linguistic objects as URIs. The examples in the previous section merely assigned IDs that are unique within a specific NAF representation. We want to link specific information in our KnowledgeStore to the specific source(s) of this information in the text using the `denotedBy`

relation provided by GAF. This can only be achieved if the linguistic information identified in text has truly unique IDs (i.e. the need an ID that is unique beyond the domain of the NAF representation they are in). Unique IDs of elements in standard NAF can be deducted by combining the ID of the element with the unique ID of the document, but it is more straightforward if the ID need not be deducted, but is directly defined in a RDF representation.

In this section, we will briefly outline the requirements for a RDF representation of NAF and illustrate what this looks like by representing the examples of the text and terms layer in NAF-RDF. We will first elaborate on differences in structure in Section 4.2. Section **??** will briefly elaborate on using ontologies to represent linguistic attributes and values.

4.1 The structure of NAF-RDF

Several revisions are needed to convert standard NAF to RDF. As a convention in RDF, objects are marked in uppercase and properties are in lowercase. An object can have properties, but no objects (the XML structure cannot be read in as RDF if objects have objects). This is why at certain locations additional layers must be introduced. We will list additional steps in the conversion below.

There are some general principles and actions for NAF-RDF which apply to the entire NAF structure. They are the following:

1. use macros to define abbreviations for RDF locations
2. avoid using '#' in RDF
3. make NAF itself an RDF object
4. define xmlns
5. naf: is default prefix for items defined as part of NAF
6. use '%40' for '@'

Furthermore, it applies to all levels of linguistic annotation that:

1. the attribute 'id' should be 'rdf:about' and underscores from ids should be removed
2. all properties will be defined within NAF and receive the prefix "naf"

```
<text xml:lang="en">
      <Text>
        <wf>
          <WF rdf:about="&docId;w1" naf:offset="0" naf:length="9" naf:token="Followers">
            <sent rdf:resource="&docId;s1"/>
            <next rdf:resource="&docId;w2"/>
          </WF>
        </wf>
        <wf>
          <WF rdf:about="&docId;w2" naf:offset="10" naf:length="2" naf:token="of"/>
            <sent rdf:resource="&docId;s1"/>
            <next rdf:resource="&docId;w3"/>
          </WF>
        </wf>
        ...
      </Text>
</text>
```

Figure 5: Example of the text layer in NAF-RDF

In addition, there may be level-specific adaptations. We will describe the changes applied to the `<text>`, `<term>`, `<chunk>` and `<srl>` layer to illustrate what is involved in the conversion from standard NAF to NAF-RDF. We aim to include information about conversions of NAF layers in the NAF specifications, so please consult this resource for information about other layers.[8] We will list the main adaptations for the four levels of information presented in the previous section below.

Figure 5 presents an example of a (partial) text layer in NAF-RDF. After the general modifications outlined above are applied, three additional changes remain. The value of the element `<wf>` is now the value of attribute `naf:token`. This is necessary, because RDF represents information in triples. We can thus not simply indicate a value for an entity without defining what the relation between the entity is. The `sent` attribute is changed into an element. Finally, we added the element `<next>`. This element is based on the property `nif:nextWord`. It has the same functionality (it provides an easy way to loop through the tokens of the text), but is different in the sense that it can be added to any kind of linguistic entity. It is currently used for the text and term layer of NAF-RDF.

Figure 6 provides an example of a term represented in NAF-RDF. Most changes made to the terms layer follow from the general requirements outlined above, as well as the aforementioned addition of the property `<next>`. There only is one notable difference in the structure of the `<externalRef>`.

[8] Despite our efforts to keep the NAF specification up to date at all times, this information may not always be complete in the NAF specifications. Please contact the authors of this paper when they information you seek is not provided in the NAF specifications.

```
<terms>
     <Terms>
       <term>
         <!--Followers-->
         <Term rdf:about="&docId;t1" naf:type="open" naf:lemma="follower" naf:pos="N" naf:morphofeat="NNS">
          <span>
          <Span><target rdf:resource="&docId;w1"/><target rdf:resource="&docId;t32"/></Span>
        </span>
           <externalRef>
             <ExternalRef>
               <referenceTo rdf:resource="&wn30g;eng-30-10099375-n"/>
               <confidence>0.525004</confidence>
             </ExternalRef>
           </externalRef>
           <externalRef>
             <ExternalRef>
              <referenceTo rdf:resource="&wn30g;eng-30-10100124-n"/>
            <confidence>0.474996</confidence>
             </ExternalRef>
            </externalRef>
           <next rdf:resource="&docId;t2"/>
         </Term>
       </term>
       ...
    <Terms>
<terms>
```

Figure 6: Example of the terms layer in NAF-RDF

In standard NAF, the confidence score was an attribute of the `externalRef` element. We cannot leave it at this location in NAF-RDF. Whereas we can interpret the `confidence` attribute any way we like when it is part of a XML schema, in RDF it becomes an attribute of the external reference. This is incorrect, since the score indicates the confidence of this external reference being associated with the term and not some confidence score of the reference itself. This difference points to the most fundamental difference between a general XML schema and RDF: an RDF representation always has a specific interpretation.

4.2 Making full use of RDF

In section 2.3, we explained that one of the advantages of using RDF is that we can define relations between values from different tools. This is, however, only possible if these values are represented as URIs. The examples presented in the previous section all contain only strings or numbers as values of attributes. For the confidence value and lemma, it is correct that they are a number and, respectively, a string. All other values in these representation should, however, ideally follow the guidelines of Linked Data as defined by

Tim Berners-Lee. We quote:[9]

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.

Ideally, these guidelines should apply to all linguistic values in NAF-RDF. In order to define a layer for representing linguistic annotations from some NLP tool, the values that this tool assigns must be defined in an ontology defined according to RDF standards. We can only formally define that the syntactic category NNS is a subclass of NP, if we can formally refer to these classes. The same applies to the attributes used in NAF. They currently are URIs in form, but they should also lead to an unambiguous definition of what they stand for.

Despite the fact that the principle ideas behind NAF only apply if we use URIs to refer to all (linguistic) things, it is possible to use strings as values in NAF. One of the main requirements of NAF is that it should be easy to add representations for the in- and output of new NLP tools to NAF. Defining an ontology forms an extra step in this integration. If it is not directly necessary to model the output in RDF (because it need not be linked to anything yet), new information can be added faster values are direct representations of the output of the NLP tools as strings. The ontology can (and should) be defined at a later time, but its absence does not form a problem for adding the information to NAF.

5 NAF in the NewsReader Pipeline

In Section 2.2, we pointed out that NIF seems to have the drawback that it is not easy to use for NLP tools. This was pointed out in the evaluation carried out by the designers of NIF themselves (Hellmann *et al.*, 2013). This drawback was also one of the reasons for us not to adapt NIF: we needed a format that could easily be read and produced by the NLP tools used in our projects. Even though NAF is still in its early stages of development, it is already being used on a distributed environment for NLP processing. In this

[9] following `http://www.w3.org/DesignIssues/LinkedData.html`, accessed 31 January 2014.

environment, NAF is used as *lingua franca* to communicate among linguistic processors: each NLP module understand NAF annotations as input and produce annotations into new NAF layers as a result of the linguistic analysis.

Three libraries for working with NAF annotations have been implemented.[10] Each library captures the structure and relations contained in the information to be manipulated, and is represented by classes which are encapsulated in several modules. These classes offer the necessary operations or methods required by the different tools to perform their tasks when recognizing the input and producing their output.

The linguistic processors and resources integrated so far integrate modules from several sources, such as the IXA-pipeline processing framework (Agerri *et al.*, 2014), or third party modules such as the MATE tool for Semantic Role labelingBjörkelund *et al.* (2010). Currently the pipeline comprises 12 modules and it is presented in Figure 7.

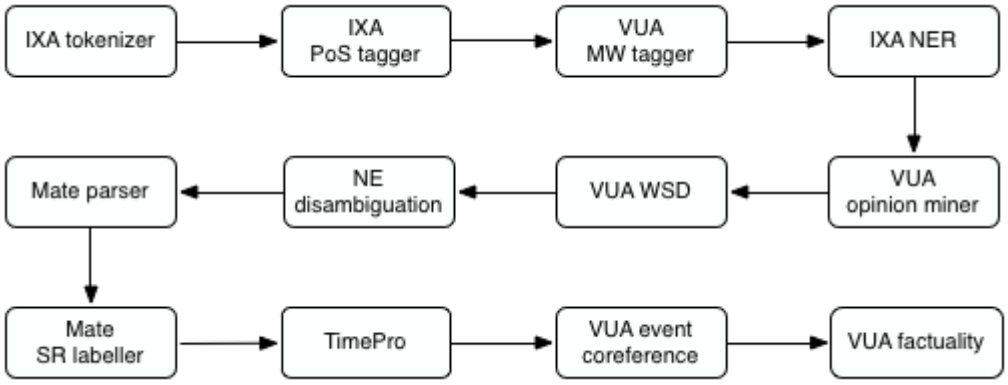

Figure 7: The IXA pipeline: the basic architecture for NewsReader

The fact that a pipeline involving several NLP modules that uses NAF for its linguistic annotations was set up in this short time indicates that NAF is suitable for internal use in NLP tools.

The pipeline presented in Figure 7 provides the first basic architecture used in NewsReader. We are currently working on a more advanced architecture that allows us to speed up processing of large amounts of data by running NLP modules in parallel. Figure 8 illustrates what such an architecture in NewsReader might look like.

The general processing environment still needs to be adapted to allow for parallel processing, but NAF is already set up to facilitate this. The

[10]The NAF website (`http://wordpress.let.vupr.nl/naf/`) provides links to NAF libraries as well as modules using NAF.

Figure 8: NewsReader architecture where processes are run in parallel

extensible layered representation NAF adopted from LAF is particularly well suited to work on a distributed environment. Processing modules represent their output in different NAF layers and they never modify annotations on the lower layers. Therefore, several processors can create new annotations to the same document in parallel.

In summary, our current pipeline shows that NAF can be used in a complex NLP pipeline and could easily be integrated in our tools. Furthermore, the extensible layered representation makes NAF suitable for the more complex pipelines we envision for NewsReader in future work.

6 Conclusion

This technical report has described the NLP Annotation Format (NAF). The format is designed to serve as a lingua franca between NLP tools developed

at different partner sites. The main motivation for designing NAF was that we needed a format that could easily be integrated in the tools we currently use and facilitates communication between the output of NLP tools and data represented in RDF. NAF fulfills these requirements by combining strengths from LAF (easy to integrate in current tools) with those from NIF (representations in RDF).

We have outlined the advantages of extensive use of RDF in linguistic annotations. Using URIs and ontologies to model linguistic annotations allows us to formally relate information from different tools facilitating the integration of alternative tools for the same task. In combination with GAF, it facilitates relating information represented in RDF to actual texts where this information is expressed as well as linguistic interpretations of this text. Finally, we can easily and efficiently query information using Semantic Web technology.

We have shown that NAF could indeed be integrated easily in our current tools and used as lingua franca in a complex pipeline. Even though it is in its initial stages, it is currently already been used in the NewsReader pipeline for event extraction and cross-document event coreference.

6.1 Discussion and Future Work

There have been several attempts to create formats that facilitate integration of a variety of NLP modules. Even though some of these efforts have found a wide resonance (notably UIMA and GATE), no clear standard for representing information has emerged. Even though the irony of introducing yet another format while aiming for standardization is not lost on us, we are of the opinion that NAF does improve interchangeability between NLP tools using different formats. Neither UIMA, GATE or any other formats we are aware of, apart from NIF, provide representations in RDF. Converting our tools to use any format that is not LAF-based would require a considerable effort. It does not make sense to invest time in converting our tools to another format that does not provide the RDF-related properties we needed for projects like NewsReader and BiographyNet.

NIF would satisfy our requirement of facilitating communication with other sources represented in RDF, but has the disadvantage that it is not easy to use within NLP tools. At least, this applies to NIF in its current form. More recent developments in NIF have extended the format making it possible to add statements about linguistic entities, rather than its current inline representations. This new direction opens the door to using NIF in a similar way as the layered extensible formats as proposed in LAF. Currently, it would still require too much effort to adapt our tools so that they directly

use NIF as in- and output. If NIF involves further in this direction, we may look into this option again.

It should be noted that our main goal is to find a format that is both easy to integrate in our tools as well as NLP tools developed by third parties and can be represented in RDF. NAF was the fastest and most straightforward way to achieve this. In future work, we hope to collaborate with researchers using other formats to further improve interoperability. One way to achieve this is to see how other formats may be converted to NAF, or how the information they provide may be integrated in our representations. However, if NIF becomes easier to integrate in our tools or more efforts are made to make UIMA and GATE (more) RDF compatible, our goals can also be met by adapting our tools to other formats.

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. IXA pipeline: Efficient and ready to use multilingual NLP tools. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014)*, Reykjavik, Iceland, 2014.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, COLING '10, pages 33–36, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Wauter Bosma, Piek Vossen, Aitor Soroa, German Rigau, Maurizio Tesconi, Andrea Marchetti, Monica Monachini, and Carlo Aliprandi. KAF: a generic semantic annotation format. In *Proceedings of the 5th International Conference on Generative Approaches to the Lexicon GL 2009*, Pisa, Italy, 2009.

Antske Fokkens, Marieke van Erp, Piek Vossen, Sara Tonelli, Willem Robert van Hage, Luciano Serafini, Rachele Sprugnoli, and Jesper Hoeksema. GAF: A grounded annotation framework for events. In *Proceedings of the first Workshop on Events: Definition, Dectection, Coreference and Representation*, Atlanta, USA, 2013.

Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP using Linked Data. In *Proceedings of the 12th International Semantic Web Conference (ISWC)*, 2013.

Nancy Ide, Laurent Romary, and Éric Villemonte de La Clergerie. International standard for a linguistic annotation framework. In *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS)*. Association for Computational Linguistics, 2003.

Frank Manola and Eric Miller, editors. *RDF Primer*. W3C Recommendation. World Wide Web Consortium, February 2004.

Marie-Francine Moens, Oleksandr Kolomiyets, Emanuele Pianta, Sara Tonelli, and Steven Bethard. D3.1: State-of-the-art and design of novel annotation languages and technologies: Updated version. Technical report, TERENCE project – ICT FP7 Programme – ICT-2010-25410, 2011.

Ineke Schuurman and Menzo Windhouwer. Explicit semantics for enriched documents. what do ISOcat, RELcat and SCHEMAcat have to offer. In *2nd Supporting Digital Humanities conference (SDH 2011), Copenhagen*, 2011.

Menzo Windhouwer. RELcat: a relation registry for isocat data categories. In *Proceedings of LREC 2012*, pages 3661–3664, 2012.