NAF: NLP Annotation Format.

NAF Deliverable

Version DRAFT

Authors: Newsreader NAF team
Affiliation: (1) EHU, (2) VUA, (3) FBK, (4) SynerScope

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

Grant Agreement No.	316404
Project Acronym	NEWSREADER
Project Full Title	Building structured event indexes of large volumes of financial and economic data for decision making.
Funding Scheme	FP7-ICT-2011-8
Project Website	http://www.newsreader-project.eu/
Project Coordinator	Prof. dr. Piek T.J.M. Vossen VU University Amsterdam Tel. + 31 (0) 20 5986466 Fax. + 31 (0) 20 5986500 Email: piek.vossen@vu.nl
Document Number	NAF Deliverable
Status & Version	DRAFT
Contractual Date of Delivery	–
Actual Date of Delivery	July 6, 2015
Type	Report
Security (distribution level)	Public
Number of Pages	49
WP Contributing to the Deliverable	WP2
WP Responsible	Aitor Soroa
EC Project Officer	Susan Fraser
Authors: Newsreader NAF team	
Keywords: –	
Abstract: This document describes the specification of NAF.	

Contents

List of Tables

1 Introduction

This document presents the first draft of NAF: NLP Annotation Format to be used within the NewsReader project. This version of NAF evolves for the format used in Kyoto, described in [Bosma *et al.*, 2009].

The following properties and desiderata are used as guidelines for defining 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)
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 that link to the URIs used in the SEM annotation layer or to background knowledge

The general approach for creating NAF will be to start with KAF, which already supports a number of desiderata mentioned above. An overview of properties taken from KAF and proposed changes is given below:

1. NAF will follow the stand-off/multi-layer architecture as also used in related formats such as KAF
2. NAF will be presented in XML, using the KAF schema as a starting point
3. The current ids in KAF will be converted to URIs
4. Additional elements may be added, for instance, to allow for references to the SEM layer or background knowledge
5. Elements may need to be reordered to turn KAF into a proper RDF graph

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 overall 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.

2 Root element

All NAF documents have a root element `<NAF>` which has the following attributes:

- `xml:lang` (**required**): language identifier .
- `version` (**required**): the version of NAF. For newsreader, we will use version **v1**

Example:

```
<NAF xml:lang="en" version="v3">
  <!--- ... --->
</NAF>
```

3 NAF header

NAF documents may have a header for describing information about the document, such as its original name, URI or a list of the linguistic processors which generated the NAF document. The NAF header is represented within the `<nafHeader>` element, which is optional but highly recommended. The header element has the following sub-elements:

3.1 fileDesc element

`<fileDesc>` is an empty element containing information about the original document itself. It has the following attributes:

- `title` (optional): the title of the document.
- `author` (optional): the author of the document.
- `creationtime` (optional): a timestamp following the *xs:dateTime*[1] format that specifies the time when the document was created.
- `filename` (optional): the original file name.
- `filetype` (optional): the original format (PDF, HTML, DOC, etc).
- `pages` (optional): number of pages of the original document.

Example:

[1] See `http://www.w3.org/TR/xmlschema-2/#isoformats`. In summary, the date is specified following the form "YYYY-MM-DDThh:mm:ss" (all fields required). To specify a time zone, you can either enter a dateTime in UTC time by adding a "Z" behind the time ("2002-05-30T09:00:00Z") or you can specify an offset from the UTC time by adding a positive or negative time behind the time ("2002-05-30T09:00:00+06:00").

```
<fileDesc creationtime="2014-01-01T00:00:00Z"
          title="The best residence in the world."
          author="casa400"
          filename="residence_hostal"
          filetype="PDF" pages="19"/>
```

3.2 public element

<public> is an empty element which stores public information about the document, such as its URI. It has the following attributes:

- publicId (optional): a public identifier (for instance, the number inserted by the capture server, or the MD5 hash number of the original document).
- uri (optional): a public URI of the document.

Example:

```
<public publicId="50ee45d106f9caf2d1cf38f29419efa8"
        uri="http://casa400.com/docs/residence.pdf"/>
```

3.3 Linguistic Processors

The header also stores the information about which linguistic processors produced the NAF document, described under <linguisticProcessors> elements. There can be several <linguisticProcessors> elements, one per NAF layer. NAF layers correspond to the top-level elements of the documents, such as "text", "terms", "deps" etc. Each <linguisticProcessors> element contains one or several <lp> elements, each one describing one specific linguistic processor.

The <lp> element, if present, has the following attributes:

- name (**required**): the name of the processor
- version (optional): processor's version
- timestamp (optional): a timestamp, denoting the date/time at which the processor was launched. It follows the XML Schema *xs:dateTime* format.
- beginTimestamp (optional): a timestamp, denoting the date/time at which the processor started the process. It follows the XML Schema *xs:dateTime* format.
- endTimestamp (optional): a timestamp, denoting the date/time at which the processor ended the process. It follows the XML Schema *xs:dateTime* format.
- hostname (optional): The name of the machine where the processor was ran..

Example:

```
<linguisticProcessors layer="text">
   <lp name="Freeling" version="2.1"
       timestamp="2012-06-25T10:05:00Z"
       beginTimestamp="2012-06-25T10:05:00Z"
       endTimestamp="2012-06-25T10:12:00Z"/>
 </linguisticProcessors>
 <linguisticProcessors layer="terms">
  <lp name="Freeling" version="2.1"
      timestamp="2012-06-25T10:10:19Z"
      beginTimestamp="2012-06-25T10:10:19Z"
      endTimestamp="2012-06-25T10:15:19Z"/>
  <lp name="ukb" version="0.1.2"
      timestamp="2012-06-25T16:10:19Z"
      beginTimestamp="2012-06-25T16:10:19Z"
      endTimestamp="2012-06-25T16:20:19Z"/>
</linguisticProcessors>
<linguisticProcessors layer="namedEntities">
  <lp name="Standfort_NE"
      version="0.1"
      timestamp="20090626_00:10:19Z"
      beginTimestamp="20090626_00:10:19Z"
      endTimestamp="20090626_00:14:19Z"/>
</linguisticProcessors>
```

Here is a full example of a NAF header:

```
<nafHeader>
 <fileDesc creationtime="2014-01-01T00:00:00Z"
           title="The best residence in the world."
           author="casa400"
           filename="residence_hostal"
           filetype="PDF" pages="19"/>
  <public publicId="3_3012"
          uri="http://casa400.com/docs/residence.pdf" />
  <linguisticProcessors layer="text">
     <lp name="Freeling" version="2.1"
         timestamp="2012-06-25T10:05:00Z"
         beginTimestamp="2012-06-25T10:05:00Z"
         endTimestamp="2012-06-25T10:12:00Z"/>
   </linguisticProcessors>
   <linguisticProcessors layer="terms">
```

```
    <lp name="Freeling" version="2.1"
        timestamp="2012-06-25T10:10:19Z"
        beginTimestamp="2012-06-25T10:10:19Z"
        endTimestamp="2012-06-25T10:15:19Z"/>
    <lp name="ukb" version="0.1.2"
        timestamp="2012-06-25T16:10:19Z"
        beginTimestamp="2012-06-25T16:10:19Z"
        endTimestamp="2012-06-25T16:20:19Z"/>
  </linguisticProcessors>
  <linguisticProcessors layer="namedEntities">
    <lp name="Standfort_NE"
        version="0.1"
        timestamp="2009-06-26T00:10:19Z"
        beginTimestamp="2009-06-26T00:10:19Z"
        endTimestamp="2009-06-26T00:14:19Z"/>
  </linguisticProcessors>
</nafHeader>
```

4 Raw layer

NAF document can encode a "raw" layer with the actual document to be processed. This layer is entirely optional, and is encoded as a `CDATA` section.

Given this input document:

> John taught mathematics 20 minutes every Monday in New York.
>
> He liked it a lot!

The NAF raw layer would look like:

```
  <raw><![CDATA[John taught mathematics 20 minutes every Monday in New York.

He liked it a lot!]]>
  </raw>
```

5 Topics layer

This layer encodes the topics for the document. The topics correspond to the whole document, and are usually assigned by authomatic methods, such as text classification processors. The topics are annotated within the `<topics>` element, and each topic is enclosed by a `<topic>` element.

The `<topic>` element has the following attributes:

- `source` (optional): A reference to the entity responsible for creating the annotation.
- `method` (optional): The name of the method used to create the annotation. This attribute is usually used in conjunction with the `source` attribute.
- `confidence` (optional): this attribute is optional but its presence is highly recommended. It gives a value of "confidence" for the annotation. The confidence value can be in fact almost anything (similarity score, the value of the margin on a SVM based classification, etc), as long as it can be used to sort all annotations sharing the `source` and `method` attributes.
- `uri` (optional): if the topic is a resource from an external reference, an URI to this resource. For instace, it could be a URI pointing to a Wikipedia category page, etc.

The content of the `<topic>` element is a string with the topic.

Here is an example of topic annotation:

```
<topics>
  <topic source="newsreader"
         confidence="0.8"
         method="textclassification-svm"
         URI="http://en.wikipedia.org/wiki/Category:Tower_mills">
   Tower Mills</topic>
  <topic confidence="0.2"
         source="newsreader"
         method="textclassification-svm"
         URI="http://en.wikipedia.org/wiki/Category:Windmills_in_Somerset">
   Windmills in Somerset</topic>
</topics>
```

6 Word forms

After tokenization step, all word forms are annotated within the `<text>` element, and each form is enclosed by a `<wf>` element.

The `<wf>` element has the following attributes:

- `id` (**required**): the unique id for the word form, starting with the prefix "w".
- `offset` (**required**): The offset (in characters) of the original word form.
- `lenght` (**required**): The lenght (in characters) of the original word form.
- `sent` (**required**): sentence id of the token.
- `para` (optional): paragraph id.
- `page` (optional): page id.
- `xpath` (optional): in case of source XML files, the xpath expression identifying the original word form.

NAF requires that input documents are encoded in UTF-8 and offsets are described in characters. The purpose of the offset is to identify the source string within the input document referred by the token, which is the string starting at character `offset` and ending at `offset + length` (exclusive).

The identifiers associated with sentences, paragraphs and pages (`sent`, `para` and `page` attributes) have to be numeric positive values in increasing order. The motivation is that the NAF has to allow queries such as "give me the words of the next (previous) sentence", which requires sentence identifiers to be numbers.

The NAF tokenization looks like:

```
<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>
  <wf id="w12" offset="62" length="2" sent="2" para="2">He</wf>
  <wf id="w13" offset="65" length="5" sent="2" para="2">liked</wf>
  <wf id="w14" offset="71" length="2" sent="2" para="2">it</wf>
```

```
  <wf id="w15" offset="74" length="1" sent="2" para="2">a</wf>
  <wf id="w16" offset="76" length="3" sent="2" para="2">lot</wf>
  <wf id="w17" offset="80" length="1" sent="2" para="2">!</wf>
</text>
```

7 Terms

Terms are lexical units which refer to word forms (and groups multi word forms). Terms are attached to several morphosyntactic information (such as part of speech, lemma, etc), and may be linked to external resources like WordNet senses, etc.

The `<term>` element has the following attributes:

- `id` (**required**): unique identifier starting with the prefix "t". If the term is a compound term (see Section 7.3), the prefix is "t.mw".
- `type` (optional): type of the term. Currently, 2 values are possible:
  - `open`: open category term
  - `close`: close category term
- `lemma` (optional): lemma of the term
- `pos` (optional): part of speech. The first letter of the pos attribute must be one of the following:

  N common noun
  R proper noun
  G adjective
  V verb
  P preposition
  A adverb
  C conjunction
  D determiner
  O other

- `morphofeat` (optional): morphosyntactic feature encoded as a single attribute.
- `head` (optional): if the term is a compound, the id of the head component (see Section 7.3).
- `case` (optional): declension case of the term.

The `<term>` element have the following sub-element:

- `span` (**obligatory**): used to identify the tokens that the term spans. A single term may refer to one or more words, in case of multiword expressions. The `<span>` element has as many `<target>` sub-elements as spanned tokens (see example below).

Example of term level annotations:

```
<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>
```

7.1 Sentiment features

The term layer represents sentiment information which is context-independent and that can be found in a sentiment lexicon. It is related to concepts expressed by words/ terms (e.g. "beautiful") or multi-word expressions (e. g. "out of order"). NAF provides possibilities to store sentiment information at term level and at sense/synset level. In the latter case, the sentiment information is included in the `<externalReference>` section (Section 7.2) and a word sense disambiguation process may identify the correct sense along its sentiment information.

The `<sentiment>` element has the following attributes:

- `Resource` (**required**): identifier and reference to an external sentiment resource.
- `Polarity` (**required**): Refers to the property of a word/sense to express positive, negative or no sentiment. These values are possible:
  - positive
  - negative
  - neutral
  - Or a numerical value.
- `Strength` (optional): refers to the strength of the polarity.
  - weak
  - average
  - strong
  - Or a numerical value
- `Subjectivity` (optional): refers to the property of a words to express an opinion (or not).
  - subjective
  - objective
  - factual
  - opinionated

- `Sentiment_semantic_type` (optional): refers to a sentiment-related semantic type. Possible values:
  - Aesthetics_evaluation
  - Moral_judgment
  - Emotion
  - etc.
- `Sentiment_modifier` (optional): refers to words which modify the polarity of another word. Possible values:
  - intensifier polarity shifter
  - weakener polarity shifter
- `Sentiment_marker` (optional): refers to words which themselves do not carry polarity, but are kind of vehicles of it. Possible values:
  - Find, think, in my opinion, according to ...
- `Sentiment_product_feature` (optional): refers to a domain; mainly used in feature-based sentiment analysis. Values are related to specific domain. These are some possible values for for the tourist domain:
  - staff, cleanliness, beds, bathroom, transportation, location, etc..

Table 1 shows sentiment values for some words.
Example:

```
<!-- term level sentiment annotation -->
<term id="t2" lemma="nice" pos="G">
  <sentiment resource="VUA_polarityLexicon_word" polarity="positive"
    strength="average" subjectivity="subjective"
    sentiment_semantic_type="behaviour/trait" />
  <span><target id="w2"/></span>
<!-- sense level sentiment annotation -->
</term>
<term id="t5" lemma="warm" pos="G">
  <sentiment/>
  <span><target id="w5"/></span>
  <externalReferences>
    <externalRef resource="WN-ENG" reference="c_1009" conf="0.38">
      <sentiment resource="VUA_polarityLexicon_synset" polarity="positive"
          strength="average" subjectivity="subjective"
         sentiment_semantic_type="behaviour/traitEvaluation"/>
    </externalRef>
```

Word	Sentiment attributes
beatiful	polarity = "positive" strength = "average" subjectivity = "subjective" sentiment_semantic_type = "aesthetics_evaluation" sentiment_modifier = "" sentiment_product_feature = "general"
valley	polarity = "negative" strength = "average" subjectivity = "factual" sentiment_semantic_type = "" sentiment_modifier = "" sentiment_product_feature = "beds"
very	polarity = "" strength = "" subjectivity = "" sentiment_semantic_type = "" sentiment_modifier = "intensifier" sentiment_product_feature = ""

Table 1: Sentiment attributes.

```
    <externalRef resource="WN-ENG" reference="c_1008" conf="0.31">
      <sentiment resource="VUA_polarityLexicon_synset" polarity="positive"
         strength="average" subjectivity="objective"
         sentiment_semantic_type="temperature"/>
    </externalRef>
  </externalReferences>
</term>
```

7.2 External References

The `<externalReferences>` element is used to associate terms to external resources, such as elements of a Knowledge base, an ontology, etc. It consists of several `<externalRef>` elements, one per association. The `<externalRef>` elements may be nested, meaning that each externalRef refines the relation expressed by the parent element.

The `<externalRef>` element has the following attributes:

- `resource` (**required**): indicates the identifier of the resource referred to.
- `reference` (**required**): code of the referred element. If the element is a synset of some version of WordNet, it is recommended to follow the pattern:

```
[a-z]3-[0-9]2-[0-9]+-[nvars]
```

which is a string composed by four fields separated by a dash. The four fields are the following:

- Language code (three characters, lowercase).
- WordNet version (two digits).
- Synset identifier composed by digits.
- POS character:

  n noun
  v verb
  a adjective
  r adverb

examples of valid patterns are: "eng-20-12345678-n", "spa-16-017403-v", etc.

- `reftype` (optional): indicates the kind of relation the externalRef is expressing.
- `status` (optional): indicates the status of the relationship.
- `source` (optional): the name of the process which created the external reference.
- `confidence` (optional): a floating number between 0 and 1. Indicates the confidence weight of the association.

Example of term level annotations:

```
<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>
    <externalReferences>
      <externalRef resource="WN-1.7" reference="eng-17-00861095-v"
                   confidence="0.80">
        <externalRef resource="ontology" reference="Teach"
                     reftype="SubClassOf">
        </externalRef>
      </externalRef>
      <externalRef resource="WN-1.7" reference="eng-17-00859568-v"
                   confidence="0.20"/>
    </externalReferences>
```

```
</term>
<term id="t3" lemma="mathematics" pos="N">
  <span>
    <target id="w3"/>
  </span>
  <externalReferences>
    <externalRef resource="WN-1.7" reference="eng-17-04597590-n"
                 confidence="1.0"/>
  </externalReferences>
</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>
<externalReferences>
  <externalRef resource="WN-1.7" reference="eng-17-12621100-n"
               confidence="0.80"/>
  <externalRef resource="WN-1.7" reference="eng-17-12631889-n"
               confidence="0.20"/>
</externalReferences>
<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>
  <externalReferences>
    <externalRef resource="WN-1.7" reference="eng-17-12557842-n"
                 confidence="1.0"/>
  </externalReferences>
</term>
<term id="t7" lemma="in" pos="P">
  <span>
    <target id="w8"/>
  </span>
</term>
```

```
  <term id="mw8" lemma="New_York" pos="R">
    <span>
      <target id="w9"/>
      <target id="w10"/>
    </span>
  </term>
</terms>
```

7.3 Compound terms

Compound terms can be represented in NAF by including `<component>` elements within `<term>` elements. For example, the term landbouwbeleid (English: agriculture policy) would look like this:

```
<term id="t.mw7" head="t.mw7.1" lemma="landbouwbeleid" pos="N" type="open">
  <span>
    <target id="w7"/>
    <target id="w8/>
  </span>
  <component id="t.mw7.1" lemma="landbouw" pos="N" type="open">
    <span>
     <target id="w7"/>
    </span>
    <externalReferences>...</externalReferences>
  </component>
  <component id="t.mw7.2" lemma="beleid" pos="N" type="open">
    <span>
     <target id="w8"/>
    </span>
    <externalReferences>...</externalReferences>
  </component>
  <externalReferences>...</externalReferences>
</term>
```

Note: the `id` attribute of compound terms start with the prefix "t.mw".

The `<component>` element has the same attributes and structure as the `term` element, the only difference being that components can not contain more `component` elements.

8 Markable layer

NAF includes a "markable" layer, whose purpose is to group tokens and attach information to them. The layer is repsesented under a `<markables>` element.

The `<markable>` layer comprises one or more `<mark>` elements, one per token group. The `<mark>` element has the following attributes:

- `id` (**required**): unique identifier starting with the prefix "m".
- `source` (optional): the source (or purpose) of the layer.

Apart from this, `<mark>` element has the same attributes and sub-elements as the `<term>` element described in section 7[2].

Below is an example of a markable layer produced by a tool which links token groups to DBpedia entities:

```
<markables>
  <!--Football Championship Subdivision-->
  <mark id="m42" lemma="Football Championship Subdivision"
        source="DBpedia">
    <span>
      <target id="w128"/>
      <target id="w129"/>
      <target id="w130"/>
    </span>
    <externalReferences>
      <externalRef resource="spotlight"
      reference="http://dbpedia.org/resource/Division_I_(NCAA)"
      confidence="1.0"/>
    </externalReferences>
 </mark>
</markables>
```

[2]Except that `<mark>` elements do not contain component sub-elements.

9 Dependency relations

Dependencies represent dependency relations among terms. Each dependency is represented by an empty `<dep>` element and span previous terms.

The `<dep>` element has the following attributes:

- `from` (**required**): term id of the source element
- `to` (**required**): term id of the target element
- `rfunc` (**required**): relational function. Among others, it can be one of:
  - `mod`: indicates the word introducing the dependent in a head- modifier relation. For instance:

`mod(by,gift,Peter)`	the gift of a book by Peter
`mod(of,examination,patient)`	the examination of the patient

  - `subj`: indicates the subject in the grammatical relation Subject-Predicate. For instance:

`subj(arrive,John,_)`	John arrived in Paris
`subj(employ,Microsoft,_)`	Microsoft employed 10 C programmers
`subj(employ,Paul,obj)`	Paul was employed by Microsoft

  - `csubj, xsubj, ncsubj`: The Grammatical Relations (RL) s csubj and xsubj may be used for clausal subjects, controlled from within, or without, respectively. ncsubj is a non-clausal subject. For instance:

`xsubj(win,require,_)`	to win the America's Cup requires heaps of cash

  - `dobj`: Indicates the object in the grammatical relation between a predicate and its direct object. For instance:

`dobj(read,book,_)`	read books

  - `iobj`: The relation between a predicate and a non-clausal complement introduced by a preposition; type indicates the preposition introducing the dependent. For instance:

`iobj(in,arrive,Spain)`	arrive in Spain
`iobj(into,put,box)`	put the tools into the box
`iobj(to,give,poor)`	give to the poor

  - `obj2`: The relation between a predicate and the second non-clausal complement in ditransitive constructions. For instance:

`obj2(head,dependent)`	
`obj2(give,present)`	give Mary a present
`obj2(mail,contract)`	mail Paul the contract

- `case` (optional): declension case

Example of dependency relation annotations:

```
<deps>
<!-- subj(teach, John) -->
<dep from="t1" to="t2" rfunc="subj" />
<!-- dobj(teach, Mathematics) -->
<dep from="t3" to="t2" rfunc="dobj" />
<!-- iobj(teach, New_York) -->
<dep from="t8" to="t2" rfunc="iobj" />
</deps>
```

10 Constituency parsing

This layer represent the output of constituency analysis parsers, i.e., full syntactic tree of sentences. The top element of the layer is `<constituency>`, and each sentence (parse tree) is represented by a `<tree>` element.

The `<tree>` element has one optional attribute:

- `type` (optional): the type of the tree.

Sometimes `<tree>` elements can be used to represent sentence structures that are not proper syntactic constituent trees. One typical example is grouping nested clauses inside a sentence. In those cases, it is useful to mark the `<tree>` element with a type (for instance, `type="clause"`).

Inside each `<tree>`, there are three types of elements:

- `<nt>` elements representing non-terminal nodes.
- `<t>` elements representing terminal nodes.
- `<edge>` elements representing in-tree edges.

The `<nt>` element represents the internal nodes of the parse tree. It has the following attribute:

- `id` (**required**): unique identifier starting with the prefix “nter”;
- `label` (**required**): the category label of the node (for instance, ’S’, ’NP’, ’VP’, etc).

The `<t>` element represents the leaf nodes of the parse tree. It has the following attributes:

- `id` (**required**): unique identifier starting with the prefix “ter”;

The `<t>` element contains a `<span>` element pointing to the *term* layer. Each `<span>` contains one or more `<target>` elements, with the following attributes:

- `id` (**required**): id of the target term.

The `<edge>` attribute links child nodes with its parent. It has the following attribute:

- `id` (optional) unique identifier starting with the prefix “tre” (tree edge);
- `from` (**required**): id of the child node. The child node can be a terminal or a non-terminal.
- `to` (**required**): id of the parent node. The parent node is a non-terminal.
- `head` (optional): a ”yes” value indicates that the child node is the head constituent of the sub-tree pointed by the parent node.

Note that the order in which the `<edge>` elements appear in the XML document is important. In particular, it has to maintain the relative order among the sub-clauses of constituents. Therefore, in NAF the order of the edges pointing to the same parent (having the same value for the `to` attribute) has to be preserved.

Given the sentence:

> The dog ate the cat.

Its lispified parse tree is (heads are marked with an asterisk):

> (S (NP (DET The) *(NN dog)) *(VP *(V ate) (NP ((DET the) *(NN cat)))) (. .))

The NAF representation would be as follows:

```
<constituency>
  <tree>
    <!-- Non-terminals -->
    <nt id="nter0"  label="ROOT"/>
    <nt id="nter1"  label="S"/>
    <nt id="nter2"  label="NP"/>
    <nt id="nter3"  label="VP"/>
    <nt id="nter4"  label="V"/>
    <nt id="nter5"  label="NP"/>
    <nt id="nter6"  label="DET"/>
    <nt id="nter7"  label="NN"/>
    <nt id="nter8"  label="DET"/>
    <nt id="nter9"  label="NN"/>
    <nt id="nter10" label="."/>
    <!-- Terminals -->
    <!-- The -->
    <t id="ter1"><span><target id="t1"/></span></t>
    <!-- dog -->
    <t id="ter2"><span><target id="t2"/></span></t>
    <!-- ate -->
    <t id="ter3"><span><target id="t3"/></span></t>
    <!-- the -->
    <t id="ter4"><span><target id="t4"/></span></t>
    <!-- cat -->
    <t id="ter5"><span><target id="t5"/></span></t>
    <!-- . -->
    <t id="ter6"><span><target id="t6"/></span></t>

    <!-- tree edges. Note: order is important! -->
    <edge id="tre1" from="nter1" to="nter0"/>            <!-- ROOT <- S -->
    <edge id="tre2" from="nter2" to="nter1"/>            <!-- S <- NP -->
```

```
    <edge id="tre3" from="nter6" to="nter2"/>             <!-- NP <- DET -->
    <edge id="tre4" from="ter1" to="nter6"/>              <!-- DET <- The -->
    <edge id="tre5" from="nter7" to="nter2" head="yes"/>  <!-- NP <- NN (head) -->
    <edge id="tre6" from="ter2" to="nter7"/>              <!-- NN <- dog -->
    <edge id="tre7" from="nter3" to="nter1" head="yes"/>  <!-- S  <- VP (head) -->
    <edge id="tre8" from="nter4" to="nter3" head="yes"/>  <!-- VP <- V (head) -->
    <edge id="tre9" from="ter3" to="nter4"/>              <!-- V  <- ate -->
    <edge id="tre10" from="nter5" to="nter3"/>            <!-- VP <- NP -->
    <edge id="tre11" from="nter8" to="nter5"/>            <!-- NP <- DET -->
    <edge id="tre12" from="ter4" to="nter8"/>             <!-- DET <- the -->
    <edge id="tre13" from="nter9" to="nter5" head="yes"/> <!-- NP <- NN (head) -->
    <edge id="tre14" from="ter5" to="nter5"/>             <!-- NN <- cat -->
    <edge id="tre15" from="ter6" to="nter10"/>            <!-- . <- . -->
    <edge id="tre16" from="nter10" to="nter1"/>           <!-- S <- . -->
  </tree>
</constituency>
```

11 Chunks

Chunks are noun or prepositional phrases, spanning terms.

The `<chunk>` element has the following attributes:

- `id` (**required**): unique identifier, starting with the prefix "c".
- `head` (**required**): the id of the chunk's head.
- `phrase` (optional): type of the phrase.
- `case` (optional): declension case.

Example of chunk annotations:

```
<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>
```

12 Named Entities

Named entities are information units such as names, including person, organization and location names, and numeric expressions including time, date, money and percent expressions. NAF describes named entity mentions present in text in a separate layer (`<entities>`), described in this section. The `<entity>` element is used to represent a named entity in the document. It may be used either for referencing single mentioned entities or multi-mentioned entities: in the latter case, the element contains clusters of term spans, which we call mentions, each mention referencing the same entity. The text anchor is described by means of the `<span>` elements, which always refer to term elements. `<entity>` elements also annotates the type of the entity mention. Besides, it can be linked to an external resource (such as Wikipedia) using the `<externalRef>` element.

The `<entity>` element has the following attributes:

- `id` (**required**): unique id, starting with the prefix "e".
- `type` (optional): named entity type. Some possible values:
  - `Time`
  - `Location`
  - `Organization`
  - `Person`
  - `Money`
  - `Percent`
  - `Date`
  - `Misc`

The `<entity>` element has the following sub-elements:

- `<references>` (**required**): this element contains one or more `<span>` element, each one spaning terms.
- `externalReferences` (optional): contains one or more `<externalRef>` element.

A `<span>` sub-element can be used to reference the different occurrences of the same named entity in the document or mentions of it, using `<target>` elements to refer to the terms it spans. If the entity is composed by multiple words, multiple target elements are used.

The `<target>` element has the following attributes:

- `id` (**required**): id that refers to the target term.
- `head` (optional): a "yes" value indicates that the term is the head of the mention.

The optional <externalRef> sub-element is used to associate terms to external lexical or semantic resources. See Section 7.2.

Example of an entity mention:

```
<entities>
  <!-- John -->
  <entity id="e1" type="person">
    <references>
      <span>
        <target id="t1"/>
      </span>
    </references>
  </entity>
  <!-- New York -->
  <entity id="e2" type="location">
    <references>
      <span>
        <target id="t8"/>
      </span>
    </references>
    <externalReferences>
      <externalRef resource="Wikipedia"
                   reference="New_York"
                   confidence="0.85"/>
      <externalRef resource="Wikipedia"
                   reference="New_York,Lincolnshire"
                   confidence="0.15"/>
    </externalReferences>
  </entity>
  <!-- London -->
  <entity id="e3" type="location">
    <references>
      <!-- London -->
      <span>
        <target id="t12" head="yes"/>
      </span>
      <!-- The capital city of England -->
      <span>
        <target id="t1" />
        <target id="t2" />
        <target id="t3" head="yes"/> <!-- city is the head -->
        <target id="t4" />
        <target id="t5" />
      </span>
    </references>
    <externalReferences>
```

```
      <externalRef reference="ref01011020"
                   resource="GeoNames"/>
      <externalRef reference="http://wwww.wikipedia.com/London"
                   resource="Wikipedia"/>
    </externalReferences>
  </entity>
</entities>
```

13 Coreference

The coreference layer creates clusters of term spans (which we call mentions) which share the same referent. For instance, “London” and “the capital city of England” are two mentions referring to the same entity. It is said that those mentions corefer. A `<coref>` element represents a mention cluster, and within `<coref>` each mention is represented by a `<span>` element (which groups term mentions using `<target>` elements). Additionally, one `<target>` element within the `<span>` may have an attribute `head` with value “yes” to represent the fact that this particular term is the head of the mention. For instance, the head of the mention “the capital city of England” is “city”.

The `<coref>` element has the following attribute:

- `id` (**required**): unique id, starting with the prefix “co”.
- `type` (optional): type of the coreference set. It describes whether the coreference cluster refers to an entity instance, and event, etc.

The `<coref>` element contains as many `<span>` elements as elements in the coreference cluster. Each `<span>` contains one or more `<target>` elements, with the following attributes:

- `id` (**required**): id that refers to the target term.
- `head` (optional): a ”yes” value indicates that the term is the head of the coreference cluster.

The `<coref>` element may also contain one `<externalReferences>` elements (see section 7.2, which links the coreference cluster to some external entity (such as the lowest common subsumer synset in WordNet).

Example of a coreference cluster:

```
<coreferences>
  <coref id="co1" type="entity">
    <!-- London -->
    <span >
      <target id="t12" head="yes"/>
    </span>
    <!-- the capital city of England -->
    <span>
      <target id="t1"/>
      <target id="t2"/>
      <target id="t3" head ="yes"/> <!-- city is the head -->
      <target id="t4"/>
      <target id="t5"/>
    </span>
  </coref>
</coreferences>
```

14 Opinion layer

The sentiment related information in NAF is represented at two levels: the (already existing) term layer and a (new) opinion layer. The term layer represents information copied from an external source like the sentiment lexicon. Information can be stored at both word and sense-synset level. Sentiment information in the term layer is the building block for the sentiment analysis tool which then tries to generate information needed to fill the slots (opinion holder, opinion target, opinion epression) of an opinion triplet. These triplets are represented in the opinion layer. Starting from these triplets, final sentiment analysis results can be calculated.

Basic forms of sentiment analysis where polarity is aggregated at sentence or document level may ignore the opinion layer triplets and make use of the term level sentiment only. In these cases (for example, polarity classification of product reviews) , opinion target (i. e. the product) and opinion holder (i. e. the reviewer) are known on beforehand. However, when more complex sentiment analysis is performed, the opinion triplets are needed. An example of such an analysis, is product feature-based SA where the target of the opinion is not just the product but some specific feature of the product. Another example concerns the analysis of , for example, news or blogs where different people (i. e. opinion holders) may have different opinions about different targets. In those cases, aggregation of the polarity values found in the text, is not enough.

The `<opinion>` element has one attribute:

- `id` (**required**): the unique identifier of the opinion, starting with the prefix "o".

The `<opinion>` element contains of the following subelements:

- `<opinion_holder>`: the holder of the opinion, that is, the speaker or some actor in the text.
- `<opinion_target>` : the target of the opinion (about what).
- `<opinion_expression>`: the opinion expression, which spans the terms comprising the opinion proper.

These three elements contain a `<span>` sub-element spanning terms.

The `<opinion_holder>` element has the following attributes:

- `type` (optional): the type of the holder (for instance, "Speaker/Writer").

The `<opinion_expression>` element has the following attributes:

- `polarity` (optional): refers to the positive or negative orientation of the expression.
- `strength`: refers to the strength of the expression.

- `subjectivity`: refers to whether an expression is subjective or not.
- `sentiment_semantic_type`: refers to sentiment related semantic types like emotion, judgment, belief, speculation.
- `sentiment_product_feature` : refers to specific features of entities, to be used in feature/aspect-based sentiment analysis.

NAF Example:

```
<opinions>
  <!-- They had a nightmare with Hilton Hotel Paris. -->
  <opinion id="o1">
    <opinion_holder type="Speaker/Writer" >
      <span>
        <target id="t1"/>
      </span>
    </opinion_holder>
    <opinion_target>
      <span>
        <target id="t6"/>
        <target id="t7"/>
        <target id="t8"/>
      </span>
    </opinion_target>
    <opinion_expression polarity="negative" strength="strong"
                        subjectivity="subjectivity"
                        sentiment_semantic_type="evaluation"
                        sentiment_product_feature="">
      <span>
        <target id="t3"/>
        <target id="t4"/>
      </span>
    </opinion_expression>
  </opinion>
</opinions>
```

15 Attribution Layer

The attribution layer is meant to represent information about the source of a statement, when this is explicitly mentioned in the text. It consists of `<statement>` elements that contain, at least, the `<statement_target>` and, typically, the `<statement_source>`. If present, the `<statement_cue>` can also be represented. All elements contain a span that points to relevant terms in the term layer.

A `<statement>` has the following subelements:

- statement_target (**required**): provides the span of terms that constitute the statement itself, i.e. where the attribution applies to (in the example below *Hilton Hotel Paris was a nightmare*).
- statement_source: provides the span that expresses the source of the statement (*They* in the example)
- statement_cue: provides the span that explicitly links the statement to the source (*said* in the example, but also expressions such as *according to*)

```
<attribution>
  <!-- They said Hilton Hotel Paris was a nightmare. -->
  <statement id="a1">
    <statement_target>
      <span>
        <target id="t3">
        <target id="t4">
        <target id="t5">
        <target id="t6">
        <target id="t7">
      </span>
    </statement_target>
    <statement_source>
      <span>
        <target id="t1">
      </span>
    </statement_source>
    <statement_cue>
      <span>
        <target id="t2">
      </span>
    </statement_cue>
    </statement>
</attribution>
```

16 Semantic Role Labeling

Semantic Role Labeling (SRL) is a shallow semantic analysis which detects semantic arguments associated with predicates. In NAF, the SRL information is stored on a separate layer, whose top element is `<srl>`.

Each annotated predicate is represented by an `<predicate>` element. The `<predicate>` element has the following attributes:

- `id` (**required**): the identifier of the predicate, starting with the prefix "pr".
- `uri` (optional): An URI to an external resource representing the model of an event/predicate. For instance, it could be a reference to a frame in *FrameNet* [Baker *et al.*, 1997], `PropBank` [Kingsbury and Palmer, 2002], etc.
- `confidence` (optional): a floating number between 0 and 1. Indicates the confidence weight of the association.

Inside `<predicate>`, there are the following sub-elements:

- One `<externalReferences>` element, which links the predicate with one or more external resources.
- One `<span>` element, spaning terms, and which point to the surface form of the predicate in the document.
- One or more `<role>` elements which fill the arguments of the predicate.

As always in NAF, the `<span>` element contains one or more `<target>` elements with the following attribute:

- `id` (**required**): id of the target term.

The `<role>` element represents filler of a particular argument of the predicate. It has the following attributes:

- `id` (**required**): the id of the role, starting with prefix "rl".
- `semRole` (**required**): the role type (for instance, "A0" or "A1" if the role belongs to a PropBank predicate).
- `uri` (**optional**): an URI pointing to an externar reference representing the particular role.
- `confidence` (**optional**): a confidence value.

Inside `<role>`, there are the following sub-elements:

- One `<externalReferences>` element, which links the predicate with one or more external resources.

- One `<span>` element, spaning terms, and which point to the surface form of the predicate in the document.

Each `<span>` contains one or more `<target>` elements, with the following attributes:

- `id` (**required**): id of the target term.

Given the following sentence:

> She was making apple jam for her friends and it was a surprise.

The SRL layer would be encoded as follows:

```
<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>
```

17 Time expressions and events

This section describes representing relations among events, events and time expressions, or relations among time expressions. We first describe how to represent events and time relations. Then, we describe how to represent relations among them.

17.1 Events

Within NAF, events are represented as coreference clusters in the coreference layer 13. Thus, the way to refer to those events is to just span to a coreference element (c.f. Section 13). For instance, the following example:

```
<coreferences>
  <coref id="coevent2" type="event">
    <span>
      <!--liked-->
      <target id="t13"/>
    </span>
  </coref>
  <coref id="coevent1" type="event">
    <span>
      <!--taught-->
      <target id="t2"/>
    </span>
  </coref>
</coreferences>
```

the coreference clusters `coevent2` and `coevent1` (of type "event") represent two events.

17.2 Time expressions

NAF follows TimeML schema for annotating time expressions under the `timeExpressions` layer. Inside `timeExpressions` there are many `<timex3>` elements, each one describing one time expression. The `<timex3>` in NAF have the same semantics as TimeML. The only difference is that TimeML inserts `<timex3>` elements along with the running text in the document, wheras NAF uses a stand-off approach. Please refer to TimeML specification document [Pustejovsky *et al.*, 2010] for further information.

The `<timex3>` element has the following attributes:

- `id` (**required**): unique identifier starting with the prefix "tmx" ("tmx1", "tmx2", etc.)
- `type` (**required**): type of the timex3 expression. Possible values:
  - DATE

  - TIME
  - DURATION
  - SET

- `beginPoint` (optional): term id of beginning point.
- `endPoint` (optional): term id of end point.
- `quant` (optional): used for specifying sets that denote quantified times. Generally a literal from the text that quelifies over the expression. Usual values are "EVERY", "SOME", etc.
- `freq` (optional): Used for specifying sets that denote quantified times. It contains an integer value and a time granularity to represent any frequency contained in the set. Usual values are "2X" (twice-a-month), "3D" (three-days), etc.
- `functionInDocument` (default value: "NONE"): indicates the function of the timex3 providing an anchor to other temporal expressions in the document. One of this values:

Value	Description
CREATION_TIME	date and time the identified resource where first created.
EXPIRATION_TIME	date and time when the right to publish material expires.
MODIFICATION_TIME	date and time the resource was last modified.
PUBLICATION_TIME	date and time when the resource is released to the public.
RELEASE_TIME	date and time when the resource may be distributed.
RECEPTION_TIME	date and time when the resource was received on current system.
NONE	—

- `temporalFunction` (default: "false"): whether the timex3 is used as a temporal function ("two weeks ago"). Possible values are "true" or "false".
- `value` (optional): XML datatype based on 2002 TIDES guideline (expanding ISO 8601) for representing dates, times and durations. Possible values are "Duration", "Time", "WeekDate", "Season", "PartOfYear", "ParPrFu", etc.
- `valueFromFunction` (optional): used when the value is taken from a temporal function timex3. The value is the identifier (tmx3id) of the timex3.
- `mod` (optional): Used for temporal modifiers that cannot be expressed either within value proper, or via links or temporal functions. Possible values:

  - BEFORE
  - AFTER
  - ON_OR_BEFORE
  - ON_OR_AFTER
  - LESS_THAN
  - MORE_THAN
  - EQUAL_OR_LESS
  - EQUAL_OR_MORE
  - START
  - MID
  - END
  - APPROX

- `anchorTimeID` (optional): point to another timex3 in case of expression such as "last week", which have a functional interpretation. `anchorTimeID` provides the reference point to which the function interpretation applies. The value is the identifier (tmx3id) of the timex3.

- `comment` (optional): optional comment.

The `<timex3>` element may have one sub-element, `<span>`, which groups the word forms the expression is referring to[3].

```
<timeExpressions>
  <timex3 id="tmx1" type="DURATION">
    <span>
      <target id="w17"/>
      <target id="w18"/>
      <target id="w19"/>
      <target id="w20"/>
    </span>
  </timex3>
  <timex3 id="tmx2" type="DATE">
    <span>
      <target id="w22"/>
    </span>
  </timex3>
</timeExpressions>
```

[3]Note that usually span elements refer to terms, but timex3 span elements refer to word forms.

17.3 Temporal relations

Temporal relations describe several relations that hold between events and time expressions of a document. In NAF, all temporal relations are encoded within the `temporalRelations` layer. The layer comprises elements of type `tlink`, one per temporal relation, which mimics the semantics of `tlink` elements of TimeML.

The `tlink` element is an empty element which contains the following attributes:

- `id` (**required**): unique identifier starting with the prefix "tlink" ("tlink1", "tlink2", etc.)
- `from` (**required**): the source of the temporal relation.
- `to` (**required**): the target of the temporal relation.
- `fromType` (`required`): the type of the element the `from` attribute points to. It can be one of the following:
  - event. The element is an event.
  - timex. The element is a time expression.
- `toType` (`required`): the type of the element the `to` attribute points to. The possible values are the same as the `fromType` attribute.
- `reltype` (`required`): the type of the relation. Usually it will have the same value as the `relType` attribute of TimeML's `tlink` element, namely:
  - BEFORE
  - AFTER
  - INCLUDES
  - IS_INCLUDED
  - DURING
  - SIMULTANEOUS
  - IAFTER
  - IBEFORE
  - IDENTITY
  - BEGINS
  - ENDS
  - BEGUN_BY
  - ENDED_BY
  - DURING_INV

Here is an example of temporal relations in NAF:

```
<temporalRelations>
  <tlink from="coevent46" fromType="event"
         relType="SIMULTANEOUS"
         to="coevent47" toType="event"/>
  <tlink from="tmx0" fromType="timex"
         relType="AFTER"
         to="tmx1" toType="timex"/>
  <tlink from="coevent48" fromType="event"
         relType="BEFORE"
         to="tmx0" toType="timex"/>
</temporalRelations>
```

The first `tlink` states that events `coevent46` and `coevent47` occur simultaneously. The second states that time expression `tmx0` occurred after time expression `tmx1`. The last `tlink` states that event `coevent48` occurred before time expression `tmx0`.

18 Factuality

Information about the veracity or factuality of events can be represented in the factuality layer. The information stored in this layer is relevant for finding out whether something happened or not according to a source. It includes information on whether something is certain, probable or possible, it is true or false and whether it is a statement about the future (implying speculation).

The information is represented pointing to ontologies and specific values, which provides flexibility to add different kinds of information related to factuality and to include information from various modules trained on differently annotated datasets. In the example below, we use the values of factbank and the NewsReader annotations.

The top element of the layer is `<factualities>`. It contains `<factuality>` elements for which we have information concerning their factuality. The `<factuality>` element has the following attribute and subelements:

- id (**required**): the identifier of the factuality element. It takes the prefix "f".
- one `<span>` element (**required**): which points to the terms the factuality has scope over. Typically, the terms will correspond to the terms making up a predicate which may be accompanied by one or more of its roles.
- one ore more `<factVal>` that provide information about the factuality of the expression in the span.

The `<factVal>` element has the following attributes:

- `value` (**obligatory**): indicates the output of the factuality module
- `resource` (**obligatory**): points to the ontology or theory that introduced the value
- `confidence` (optional): indicates the confidence of the module (if provided)
- `source` (optional): indicates the module that generated the result (if more than one module was used, else this information can be obtained unambiguously from the NAF header).

```
<factualities>
  <factuality id="f1">
    <span>
        <target id="t3"/>
        <target id="t4"/>
        <target id="t5"/>
        <target id="t6"/>
        <target id="t7"/>
    </span>
    <factVal value="CT+" resource="FactBank"
```

```
            confidence="0.83"/>
    <factVal value="CERTAIN" resource="nwr:AttributionCertainty"
            confidence="0.79"/>
    <factVal value="PROBABLE" resource="nwr:AttributionCertainty"
            confidence="0.11"/>
    <factVal value="NONFUTURE" resource="nwr:AttributionTime"
            confidence="0.91"/>
    <factVal value="POS" resource="nwr:AttributionPolarity"
            confidence="0.67"/>
  </factuality>
</factualities>
```

References

[Baker *et al.*, 1997] Colin Baker, Charles Fillmore, and John B. Lowe. The berkeley framenet project. In *COLING/ACL'98*, Montreal, Canada, 1997.

[Bosma *et al.*, 2009] 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.

[Kingsbury and Palmer, 2002] Paul Kingsbury and Martha Palmer. Proceedings of the third international conference on language resources and evaluation, lrec 2002, may 29-31, 2002, las palmas, canary islands, spain. In *LREC*. European Language Resources Association, 2002.

[Pustejovsky *et al.*, 2010] James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. ISO-TimeML: An international standard for semantic annotation. In *Proceedings o the Fifth International Workshop on Interoperable Semantic Annotation*, 2010.

[Saurí and Pustejovsky, 2009] Roser Saurí and James Pustejovsky. Factbank: a corpus annotated with event factuality. *Language resources and evaluation*, 43(3):227–268, 2009.