Event Detection, version 2

Deliverable D4.2.2

Version FINAL

Authors: Rodrigo Agerri[1], Itziar Aldabe[1], Zuhaitz Beloki[1], Egoitz Laparra[1], Maddalen Lopez de Lacalle[1], German Rigau[1], Aitor Soroa[1], Antske Fokkens[2], Ruben Izquierdo[2], Marieke van Erp[2], Piek Vossen[2], Christian Girardi[3], Anne-Lyse Minard[3]

Affiliation: (1) EHU, (2) VUA, (3) FBK

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	Deliverable D4.2.2
Status & Version	FINAL
Contractual Date of Delivery	September 2014
Actual Date of Delivery	September 2014
Type	Report
Security (distribution level)	Public
Number of Pages	107
WP Contributing to the Deliverable	WP4
WP Responsible	EHU
EC Project Officer	Susan Fraser
Authors: Rodrigo Agerri[1], Itziar Aldabe[1], Zuhaitz Beloki[1], Egoitz Laparra[1], Maddalen Lopez de Lacalle[1], German Rigau[1], Aitor Soroa[1], Antske Fokkens[2], Ruben Izquierdo[2], Marieke van Erp[2], Piek Vossen[2], Christian Girardi[3], Anne-Lyse Minard[3]	
Keywords: Event detection, EN pipelines, NL pipeline, ES pipeline, IT pipeline, Scaling of text processing	
Abstract: This deliverable describes the second prototype for event detection. It focuses on English, Dutch, Italian and Spanish. It uses an open architecture which works with generic NLP modules that perform different tasks for event detection. Each task is executed by one module, which allows custom pipelines to be used for text processing.	

Table of Revisions

Version	Date	Description and reason	By	Affected sections
0.1	01 September 2014	Structure of the deliverable set	EHU	All
0.1	25 September 2014	Dutch, Italian, Spanish pipelines added	EHU, FBK, VUA	3, 4, 5
0.1	30 September 2014	Event detection section added	EHU	2
0.1	30 September 2014	Draft of the deliverable set	EHU, FBK, VUA	All
1.0	28 November 2014	Completed appendix and added references, reviewed draft	VUA,FBK, EHU	All
1.0	30 January 2015	Checked by project coordinator	VUA	-

Executive Summary

This deliverable describes the second cycle of event detection, developed within the European FP7-ICT-316404 "Building structured event indexes of large volumes of financial and economic data for decision making (NewsReader)" project. The prototype and results presented are part of the activities performed in tasks T4.2 Event Detection, T4.3 Authority and factuality computation and T4.5 Scaling of text processing of Work Package WP4 (Event Detection).

The second prototype on event detection includes improved and new modules in the English pipeline. We have improved the modules that perform tokenization, POS-tagging, parsing, time recognition, named entity recognition, word sense disambiguation, named entity disambiguation, coreference resolution, semantic role labeling, event classification, and opinion mining. We have integrated new modules to perform temporal and causal relation extraction. We have also implemented a new module for text classification. Each task is executed by one module, which allows us to custom different pipeline topologies for text processing.

In the second cycle of event detection, we have worked on the adaptation of the English pipeline to the financial domain. We have also developed Dutch, Italian and Spanish pipelines. So far, we have focused on generic NLP modules that perform the necessary tasks for event detection. The multilingual interoperable semantic interpretation of the information has also been studied.

Finally, we have continued collecting and processing different datasets. Five datasets covering different topics have been processed by the NLP pipeline. Together these sets consists of approximately 326K articles in English and 7K articles in Spanish.

Contents

1 Introduction

This deliverable describes the second version of the **Event Detection** framework developed in NewsReader to process large and continuous streams of English, Dutch, Spanish and Italian news articles. In this period, we have worked on the improvement of the systems for event detection. Event detection addresses the development of text processing modules that detect mentions of events, participants, their roles and the time and place expressions in the four project languages.

NewsReader uses an open and modular architecture for Natural Language Processing (NLP) as a starting point. The system uses the NLP Annotation Framework[1] [Fokkens *et al.*, 2014] (NAF) as a layered annotation format for text that can be shared across languages, and separate modules have been developed to add new interpretation layers using the output of previous layers. Text-processing requires basic and generic NLP steps such as tokenization, lemmatization, part-of-speech tagging, parsing, word sense disambiguation, named-entity and semantic role recognition, etc. for all the languages within the project.

Semantic interpretation involves the detection of event mentions and those named entities that play a role in these events, including time and location relations. This implies covering all expressions and meanings that can refer to events, their participating named entities, place and time relations. It also means resolving coreference relations for these named entities and relations between different event mentions. As a result of this process, the text is enriched with semantic concepts and identifiers that can be used to access lexical resources and ontologies. For each unique event, we will also derive its factuality score based on the textual properties and its provenance.

Moreover, in order to achieve cross-lingual semantic interoperability, entity and event mentions should be projected to language independent knowledge representations. Thus, named entities are linked as much as possible to external sources such as DBpedia entity identifiers while event mentions are aligned to abstract event representations thanks to the Predicate Matrix [López de Lacalle *et al.*, 2014]. We are developing new techniques and resources to achieve interoperable semantic interpretation for English, Dutch, Spanish and Italian thanks to DBpedia cross-lingual links and multilingual semantic projections through local wordnets of the Predicate Matrix. In the second year, we have continued improving existing modules and creating new ones.

NewsReader provides an abstraction layer for large-scale distributed computations, separating the *what* from the *how* of computation and isolating NLP developers from the details of concurrent programming.

This deliverable presents the main NLP processing modules for English, Dutch, Italian and Spanish addressed by the NewsReader project in order to process event across documents. The evaluation results of the pipelines will be included in deliverable D3.3.2 *Annotated data*.

The remainder of the document consists of the following sections. Section 2 presents the event detection task designed in the NewsReader project. It explains the event de-

[1] `http://wordpress.let.vupr.nl/naf`

tection task and the efforts to achieve cross-lingual semantic interoperability. Section 3 presents the main NLP processing modules for English. Sections 4, 5 and 6 describe the Dutch, Italian and Spanish processing pipelines, respectively. Our pipelines include several modules that can be used for more than one language in the project. In order to make sure that descriptions of pipelines for individual languages are self-contained, the description of the module is repeated for each individual language for which it can be used. Section 7 presents a process applied to all the pipelines in which the topic of each document is detected. Section 8 describes the processed data in this second year. Finally, Section 9 presents the main conclusions of this deliverable and a summary of each module.

2 Event Detection

This section introduces the main NLP tasks addressed by the NewsReader project in order to process events across documents in four different languages: English, Dutch, Spanish and Italian. NewsReader Deliverable D4.1[2] provides a detailed survey about the current availability of resources and tools to perform event detection for the four languages involved in the project.

Event Detection (WP04) addresses the development of text processing modules that detect mentions of events, participants, their roles and the time and place expressions. Thus, text-processing requires basic and generic NLP steps, such as tokenization, lemmatization, part-of-speech tagging, parsing, word sense disambiguation, named entity and semantic role recognition for all the languages in NewsReader. Named entities are as much as possible linked to external sources (Wikipedia, DBpedia, JRC-Names, BabelNet, Freebase, etc.) and entity identifiers. Furthermore, event detection involves the identification of event mentions, event participants, the temporal constraints and, if relevant, the location. It also implies the detection of expressions of factuality of event mentions and the authority of the source of each event mention.

Moreover, NewsReader is developping:

- New techniques for achieving interoperable Semantic Interpretation of English, Dutch, Spanish and Italian.

- Wide-coverage linguistic processors adapted to the financial domain.

- New scaling infrastructures for advanced NLP processing of large and continuous streams of English, Dutch, Spanish and Italian news articles.

During the second cycle of the NewsReader project (Event Detection, version 2), we focused on improving the existing modules of the English pipeline presented in NewsReader Deliverable 4.2.2[3] as the “IXA-pipeline”. We have also worked on the adaptation of the

[2] http://www.newsreader-project.eu/files/2012/12/NewsReader-316404-D4.1.pdf

[3] http://www.newsreader-project.eu/files/2012/12/NewsReader-316404-D4.2.pdf

pipeline to the financial domain. By the end of the year, we will evaluate and improve the event detection system based on the benchmark evaluations that are planned in WP03.

In the second year, we have worked on generic pipelines with new functionalities for Dutch, Italian and Spanish. These pipelines will also be evaluated by the gold-standards generated in WP03.

Although we already use NAF to harmonize the different outcomes, during the first cycle of the project, we detected the need to establish a common semantic framework for representing event mentions. For instance, SEMAFOR[4] uses FrameNet [Baker *et al.*, 1997] for semantic role labelling (SRL), while Mate-tools[5] uses PropBank [Palmer *et al.*, 2005] for the same task. As a backup solution, we process the text with Word Sense Disambiguation modules to match WordNet [Fellbaum, 1998] identifiers across predicate models and languages. Predicates in WordNet are linked to predicates in FrameNet and PropBank in the Predicate Matrix allowing us to derive FrameNet and PropBank roles. However, it is clear that in order to properly address interoperable semantic processing our processing tools should produce alignned semantic annotations across predicate models and languages.

During the first year of the project we addressed the problem of an interoperable Semantic Interpretation by:

- including into the English pipeline a first version of the Predicate Matrix for verbs [de Lacalle *et al.*, 2014] and nouns.
- including into the Spanish and Dutch pipelines a first version of the Predicate Matrix for verbs.

In the second cycle of the project we have continued working on this problem by:

- including a second version of the Predicate Matrix for verbs [Lacalle *et al.*, 2014] and nouns into the English pipeline.
- including a second version of the Predicate Matrix for verbs and nouns into the Spanish and Dutch pipelines.
- including a new version of the ixa-pipes-ned based on DBpedia Spotlight which also returns (if they covered in the English version of DBpedia) English identifiers into the Spanish, Dutch and Italian pipelines.

In that way, the English, Spanish, Dutch and Italian NLP processing chains are providing harmonized cross-lingual identifiers for predicates and entities.

[4] `http://code.google.com/p/semafor-semantic-parser/wiki/FrameNet`

[5] `http://code.google.com/p/mate-tools/`

2.1 Multilingual and Interoperable Predicate Models

Predicate models such as FrameNet [Baker *et al.*, 1997], VerbNet [Kipper, 2005] or PropBank [Palmer *et al.*, 2005] are core resources in most advanced NLP tasks such as Question Answering, Textual Entailment or Information Extraction. Most of the systems with Natural Language Understanding capabilities require a large and precise amount of semantic knowledge at the predicate-argument level. This type of knowledge allows us to identify the underlying typical participants of a particular event independently of its realization in the text. Thus, using these models, different linguistic phenomena expressing the same event, such as active/passive transformations, verb alternations, nominalizations, implicit realizations can be harmonized into a common semantic representation. In fact, lately, several systems have been developed for shallow semantic parsing an explicit and implicit semantic role labeling using these resources [Erk and Pado, 2004], [Shi and Mihalcea, 2004], [Giuglea and Moschitti, 2006], [Laparra and Rigau, 2013]. However, although all these systems cover the same kind of task, if they are based in different resources it becomes too difficult to establish concordances and discrepancies between their outputs over the same documents.

To allow for interoperable semantic interpretation of texts in multiple languages and predicate models, we have developed the *Predicate Matrix* [Lacalle *et al.*, 2014; de Lacalle *et al.*, 2014],[6] a new lexical resource resulting from the integration of multiple sources of predicate information including FrameNet [Baker *et al.*, 1997], VerbNet [Kipper, 2005], PropBank [Palmer *et al.*, 2005] and WordNet [Fellbaum, 1998]. By using the *Predicate Matrix*, we provide a more robust interoperable lexicon by discovering and solving inherent inconsistencies among the resources. We are now working to extend the coverage of current predicate resources (by including from WordNet morphologically related nominal and verbal concepts), to enrich WordNet with predicate information, and also to extend predicate information to languages other than English (by exploiting the local wordnets aligned to the English WordNet). The first version of the *Predicate Matrix* (v1.0) was integrated in the first prototype of the event detection system, and during this second cycle, we have integrated a new version of the *Predicate Matrix* (v1.1).

2.1.1 Sources of Predicate Information

Currently, we are considering the following sources of predicate information:

SemLink[7] [Palmer, 2009] is a project that aims to link together different predicate resources via set of mappings. These mappings make it possible to combine different information provided by these different lexical resources for tasks such as inferencing, consistency checking, interoperable semantic role labelling. We can also use these mappings to aid semi-automatic or fully automatic extensions of the current coverage of each of the resources, in order to increase the overall overlapping coverage. SemLink currently provides partial mappings between FrameNet [Baker *et al.*, 1997], VerbNet [Kipper, 2005], PropBank

[6] The Predicate Matrix can be obtained from `http://adimen.si.ehu.es/web/PredicateMatrix`

[7] `http://verbs.colorado.edu/semlink/`

[Palmer *et al.*, 2005] and WordNet [Fellbaum, 1998].

FrameNet[8] [Baker *et al.*, 1997] is a rich semantic resource that contains descriptions and corpus annotations of English words following the paradigm of Frame Semantics [Fillmore, 1976]. In Frame Semantics, a Frame corresponds to a scenario that involves the interaction of a set of typical participants, playing a particular role in the scenario. FrameNet groups words or lexical units (LUs hereinafter) into coherent semantic classes or frames, and each frame is further characterized by a list of participants or lexical elements (LEs hereinafter). Different senses for a word are represented in FrameNet by assigning different frames.

PropBank[9] [Palmer *et al.*, 2005] aims to provide a wide corpus annotated with information about semantic propositions, including relations between the predicates and their arguments. PropBank also contains a description of the frame structures, called framesets, of each sense of every verb that belong to its lexicon. Unlike other similar resources, as FrameNet, PropBank defines the arguments, or roles, of each verb individually. In consecuence, it becomes a hard task obtaning a generalization of the frame structures over the verbs.

VerbNet[10] [Kipper, 2005] is a hierarchical domain-independent broad-coverage verb lexicon for English. VerbNet is organized into verb classes extending [Levin, 1993] classes through refinement and addition of subclasses to achieve syntactic and semantic coherence among members of a class. Each verb class in VN is completely described by thematic roles, selectional restrictions on the arguments, and frames consisting of a syntactic description and semantic predicates.

WordNet[11] [Fellbaum, 1998] is by far the most widely-used knowledge base. In fact, WordNet is being used world-wide for anchoring different types of semantic knowledge including wordnets for languages other than English [Gonzalez-Agirre *et al.*, 2012]. It contains manually coded information about English nouns, verbs, adjectives and adverbs and is organized around the notion of a *synset*. A synset is a set of words with the same part-of-speech that can be interchanged in a certain context. For example, <*learn*, *study*, *read*, *take*> form a synset because they can be used to refer to the same concept. A synset is often further described by a gloss, in this case: *"be a student of a certain subject"* and by explicit semantic relations to other synsets. Each synset represents a concept that are related with an large number of semantic relations, including hypernymy/hyponymy, meronymy/holonymy, antonymy, entailment, etc.

2.1.2 Creating the Predicate Matrix

For the first version of the *Predicate Matrix*, we have focused on the links that connect VerbNet, FrameNet and WordNet. The process, explained in this section, starts from SemLink. Then, a set of sequential steps try to complete the alignments. The whole

[8] http://framenet.icsi.berkeley.edu/

[9] http://verbs.colorado.edu/~mpalmer/projects/ace.html

[10] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[11] http://wordnet.princeton.edu/

process can be divided into three consecutive steps, fully described in [de Lacalle *et al.*, 2014]:

1. Complete and extend the mappings between the lexicons of WordNet, VerbNet and FrameNet. Following [Laparra and Rigau, 2009; Laparra *et al.*, 2010; Laparra and Rigau, 2013], we apply knowledge–based Word Sense Disambiguation (WSD) algorithms that use a large-scale graph of concepts derived from WordNet to disambiguate the verbs (and also nouns, adjectives and adverbs corresponding to the FrameNet LUs) from both lexicons. Then, for each WordNet verb sense, we collect disambiguations (and alignments) to FrameNet frames and VerbNet classes.

2. Complete the mappings between VerbNet thematic-roles and FrameNet frame elements. For many cases, although there is a mapping between lemmas, the corresponding links between the roles are missing. We apply some methods that use the existing mappings and knowledge from the resources to complete these gaps.

3. We finally extend the mappings via WordNet synonyms.

2.1.3 Predicate Matrix v1.1

The current version of the *Predicate Matrix* is contained in a single file. Each row of that file represents the mapping of a role over the different resources and includes all the aligned knowledge about its corresponding WordNet verb sense.

As shown in table 1, the new version of the Predicate Matrix (PM v1.1) contains many more aligments than SemLink. First, it provides more verb aligments between VerbNet and FrameNet (from 3,285 to 9,952). Second, it doubles the WordNet verb sense aligments (from 7,620 to 14,900 VerbNet verbs and from 4,342 to 11,391 FrameNet verbs). Third, it covers more joint aligments between VerbNet, FrameNet and its corresponding WordNet verb sense (from 5,168 to 12,267). Finally, the new version of the Predicate Matrix also contains around five more VerbNet to FrameNet role aligments at a WordNet verb sense level (from 6,201 to 31,889).

	lexicon				roles
	VN-FN	VN-WN	FN-WN	VN-WN-FN	
SemLink	3,285	7,620	4,342	5,168	6,201
PMv1.1	9,952	14,900	11,391	12,267	31,889

Table 1: Number of new lexicon and role alignments in *Predicate Matrix 1.1*.

Moreover, as a side effect while creating the Predicate Matrix, we are enriching both VerbNet and FrameNet. For instance, we are enlarging the number of different verbs aligned to both VerbNet classes (from 4,394 to 8,706) and FrameNet frames (from 2,867 to 4,932).

Additionally, as the Predicate Matrix uses the verbal part of WordNet as a backbone, we now have a clear indication of how much of WordNet is still not covered by VerbNet, FrameNet or PropBank. For instance, from the total number of 25,148 WordNet verbal senses, the new version of the Predicate Matrix only contains 11,629 WordNet verb senses aligned to VerbNet classes. That is, there are 13,997 WordNet verb senses still without mappings to VerbNet classes. Similarly, the Predicate Matrix now only contains 7,573 WordNet senses aligned to FrameNet frames. Thus, there are 18,672 WordNet word senses without mappings to FrameNet frames.

Finally, although we have mainly focused on the mapping between VerbNet and FrameNet, and their connection to WordNet, we can extend the resulting new connections for those cases from PropBank that are currently mapped to any VerbNet and FrameNet element affected by the processes described here. Table 2 shows the old and new mappings to PropBank.

	lexicon			roles	
	PB-VN	PB-WN	PB-FN	PB-VN	PB-FN
SemLink	5,292	6,174	2,462	12,164	1,461
PMv1.1	11,990	12,249	2,901	23,498	1,923

Table 2: Number of new lexicon and role alignments for PropBank.

2.1.4 Spanish Predicate Matrix

The semantic resources that are part of SemLink only contain an English lexicon. Thus, the *Predicate Matrix* that extends the mappings between those resources just covers English predicates. However, if any resource in another language is linked to any of the resources included in the *Predicate Matrix* the projection to that language can be done straightforwardly.

This is the case of AnCora [Taulé *et al.*, 2008],[12] a multilevel copora that includes both for Spanish and Catalan, annotations of lemmatization, syntactic constituents, WordNet senses, coreference, named entities and also semantic roles. AnCora also develops a semantic resource called AnCoraVerb [Juan Aparicio and Martí, 2008] that contains Spanish verbal predicates and their corresponding arguments structures (see Table 3).

AnCoraVerb is based in PropBank and both resources are linked by a wide set of mappings called AncoraNet. The Spanish predicate *"verb.vender.1.default"* shown in Table 4 is for instance mapped to the English predicate *"sell.01"* and the Spanish predicate *"verb.dialogar.1.default"* is mapped to the English *"talk.01"*.

Unless AncoraNet states otherwise, the correspondance between the arguments in AncoraVerb and PropBank is direct. For the Spanish predicate *"verb.vender.1.default"*, the arguments *"arg0"*, *"arg1"* and *"arg2"* correspond to the arguments *"0"*, *"1"* and *"2"* respectively of the English predicate *"sell.01"*. The Spanish predicate *"verb.dialogar.1.default"* has its *"arg0"* linked directly to PropBank's argument *"0"*, but AncoraNet also explicitly establishes that the *"arg1"* corresponds to the argument *"2"* of PropBank.

[12] http://clic.ub.edu/corpus/ancora

```
<lexentry lemma=”vender” lng=”es” type=”verb”>
  <sense id=”1”>
    <frame default=”yes” lss=”A32.ditransitive-patient-benefactive” type=”default”>
      <argument argument=”arg0” function=”suj” thematicrole=”agt”/>
      <argument argument=”arg1” function=”cd” thematicrole=”pat”/>
      <argument argument=”arg2” function=”ci” thematicrole=”ben”>
      . . .

<lexentry lemma=”dialogar” lng=”es” type=”verb”>
  <sense id=”1”>
    <frame default=”yes” lss=”A22.transitive-agentive-theme” type=”default”>
      <argument argument=”arg0” function=”suj” thematicrole=”agt”/>
      <argument argument=”arg1” function=”creg” thematicrole=”tem”>
      . . .
```

Table 3: Examples of argument structures defined in AnCoraVerb for the predicates *“vender.1.default”* and *“dialogar.1.default”*.

```
<link ancoralexid=”verb.vender.1.default” propbankid=”sell.01”>
</link>

<link ancoralexid=”verb.dialogar.1.default” propbankid=”talk.01”>
  <arglink ancoralexarg=”arg1” propbankarg=”2”/>
</link>
```

Table 4: Examples of mappings between AnCoraVerb and PropBank predicates.

We use these aligments to replicate the entries in the *Predicate Matrix*. For this, we must define an identifier to distinguish between lines for English and Spanish predicates. Currently this identifier is based in both PropBank and AnCora predicates and arguments. For example, according to Table 5, the following lines correspond to the arguments *“0”*, *“1”* and *“2* of the predicate *“sell.01”*. They are thus identified by *“id:sell.01 id:0”*, *“id:sell id:1”* and *“id:sell id:2”* respectively.

The corresponding lines for the Spanish predicate *“vender.1.default”* are identified by *“id:vender.1.default id:arg0”*, *“id:vender.1.default id:arg1”* and *“id:vender.1.default id:arg2”*, as shown in Table 6.

Establishing such identifiers allows us to maintain the whole *Predicate Matrix* for both languages in the same file and helps to enrich the annotations of SRL systems such as the mate-tools which are based in PropBank and AnCora.

id:sell.01 id:0 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal vn:Agent wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell fn:sell.v fn:Seller pb:sell.01 pb:0 mcr:0 mcr:economy mcr:Selling mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024 SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:sell.01 id:1 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal vn:Theme wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell fn:sell.v fn:Goods pb:sell.01 pb:1 mcr:0 mcr:economy mcr:Selling mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024 SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:sell.01 id:2 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal vn:Recipient wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell fn:sell.v fn:Buyer pb:sell.01 pb:2 mcr:0 mcr:economy mcr:Selling mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024 SEMLINK;PREDICATE_MAPPING;FN_FE;SYNONYMS

Table 5: Some lines for the arguments of the predicate *"sell.01"* in the *Predicate Matrix.*

2.1.5 Nominal Predicate Matrix

Extending the *Predicate Matrix* to nominal predicates follows the same strategy for the projection to Spanish explained previously, provided the existance of semantic resources containing the argument structures for the nominalizations of the verbal predicates. In the case of English predicates, this knowledge can be obtained from NomBank [Meyers *et al.*, 2004][13] that includes nominalizations of the PropBank predicates, like *"sale.01"* aligned to the source verbal predicate *"sell.01"* (see Table 7).

For Spanish, AnCora also includes the nominalizations of its verbal predicates in a resource called AnCora-Nom. For example, *"venta.1.default"*, the nominalization of the predicate *"vender.1.default"*, is described in AnCora-Nom as Table 8 shows.

Once again, unless these resources say otherwise, the correspondance between arguments of the verbal and nominal predicates is direct. Hence, the lines we showed previously for *"sell.01"* can be replicated for its nominalization with their corresponding identifiers as shown in Table 9.

Table 10 shows how the same process can be used for replicating the lines corresponding to the Spanish predicate *"venta.default.1"*.

2.1.6 Interoperability through the Predicate Matrix

Tables 11, 12, 13 and 14 present the resulting annotations provided by the current Semantic Role Labelling system for the same event expressed using English (Tables 11 and 12) and Spanish (Tables 13 and 14) predicates in both verbal (Tables 11 and 13) and nominal

[13] http://nlp.cs.nyu.edu/meyers/NomBank.html

```
id:vender.1.default id:arg0 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Agent wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Seller pb:sell.01 pb:0 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:vender.1.default id:arg2 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Recipient wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Buyer pb:sell.01 pb:2 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;FN_FE;SYNONYMS

id:vender.1.default id:arg1 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Theme wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Goods pb:sell.01 pb:1 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;SYNONYMS
```

Table 6: Some examples of the resulting lines for the arguments of the predicate *"vender.1.default"*.

forms (Tables 12 and 14). The values included within the *externalReferences* have been obtained from the new *Predicate Matrix*. As shown, their values are the same for all of them. Obviously, now there is clear semantic interoperability between the different nominal and verbal predicates and their arguments in both Spanish and English. This information, combined with proper interoperable entity correference, disambiguation or named entity linking will be exploited to match the diverse mentions of a single event in documents in any form and language.

```
<roleset id=”sale.01” name=”commerce: seller” source=”verb-sell.01” vncls=”13.1-1”>
  <roleset id=”sale.01” name=”commerce: seller” source=”verb-sell.01” vncls=”13.1-1”>
    <roles>
      <role descr=”seller” n=”0”>
        <vnrole vncls=”13.1-1” vntheta=”Agent”/></role>
      <role descr=”thing sold” n=”1”>
        <vnrole vncls=”13.1-1” vntheta=”Theme”/></role>
      <role descr=”buyer” n=”2”>
        <vnrole vncls=”13.1-1” vntheta=”Recipient”/></role>
```

Table 7: Examples of argument structures defined in NomBank for the predicate *“sale.01”*.

```
<lexentry lemma=”venta” lng=”es” origin=”deverbal” type=”noun”>
  <sense cousin=”no” denotation=”result” id=”1” lexicalized=”no”
  originlemma=”vender” originlink=”verb.vender.1” wordnet-
  synset=”16:00721968”>
    <frame appearsinplural=”yes” type=”default”>
      <argument argument=”arg0” thematicrole=”agt”>
        <constituent frequency=”5” preposition=”de” type=”sp”/>
        <constituent frequency=”1” preposition=”por” type=”sp”/>
        <constituent frequency=”4” postype=”possessive” type=”determiner”/>
      </argument>
      <constituent frequency=”19” type=”sp”/>
      </argument>
      <constituent frequency=”7” preposition=”a” type=”sp”/>
      </argument>
```

Table 8: Description of the nominal predicate *“venta.1.default”* in AnCora-Nom.

```
id:sale.01 id:0 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal vn:Agent
wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell fn:sell.v fn:Seller pb:sell.01
pb:0 mcr:0 mcr:economy mcr:Selling mcr:Agentive;Dynamic;Social; mcr:possession
mcr:act%2:41:00 wn:4 wn:024 SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:sale.01 id:1 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal
vn:Theme wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Goods pb:sell.01 pb:1 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:sale.01 id:2 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1 vn:deal
vn:Recipient wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Buyer pb:sell.01 pb:2 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;FN_FE;SYNONYMS
```

Table 9: Some examples of the resulting lines for the arguments of the predicate *"sale.01"*.

```
id:venta.1.default id:arg0 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Agent wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Seller pb:sell.01 pb:0 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;SYNONYMS

id:venta.1.default id:arg2 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Recipient wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Buyer pb:sell.01 pb:2 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;FN_FE;SYNONYMS

id:venta.1.default id:arg1 vn:give-13.1 vn:13.1 vn:give-13.1-1 vn:13.1-1
vn:deal vn:Theme wn:deal%2:40:00 mcr:ili-30-02244956-v fn:Commerce_sell
fn:sell.v fn:Goods pb:sell.01 pb:1 mcr:0 mcr:economy mcr:Selling
mcr:Agentive;Dynamic;Social; mcr:possession mcr:act%2:41:00 wn:4 wn:024
SEMLINK;PREDICATE_MAPPING;SYNONYMS
```

Table 10: Some examples of the resulting lines for the arguments of the predicate *"venta.1.default"*.

```
Volkswagen sells its actions to Porche.
  <predicate id=”pr2”>
  <!–sells–>
    <externalReferences>
      <externalRef reference=”sell.01” resource=”PropBank”/>
      <externalRef reference=”sell.01” resource=”PropBank”/>
      <externalRef reference=”give-13.1” resource=”VerbNet”/>
      <externalRef reference=”give-13.1-1” resource=”VerbNet”/>
      <externalRef reference=”Commerce_sell” resource=”FrameNet”/>
      <externalRef reference=”contextual” resource=”EventType”/>
    </externalReferences>
    <role id=”rl2” semRole=”A0”>
    <!–by Volkswagen–>
      <externalReferences>
        <externalRef reference=”sell.01@0” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Agent” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Seller” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A1”>
    <!–its actions–>
      <externalReferences>
        <externalRef reference=”sell.01@1” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Theme” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Goods” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A2”>
    <!–to Porche–>
      <externalReferences>
        <externalRef reference=”sell.01@2” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Recipient” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Buyer” resource=”FrameNet”/>
      </externalReferences>
    </role>
  </predicate>
```

Table 11: Annotation example of the verbal predicate in English *“sell.01”*.

```
The sale of actions to Porche by Volkswagen was aproved.
  <predicate id=”pr2”>
  <!–sale–>
    <externalReferences>
      <externalRef reference=”sale.01” resource=”NomBank”/>
      <externalRef reference=”sell.01” resource=”PropBank”/>
      <externalRef reference=”give-13.1” resource=”VerbNet”/>
      <externalRef reference=”give-13.1-1” resource=”VerbNet”/>
      <externalRef reference=”Commerce_sell” resource=”FrameNet”/>
      <externalRef reference=”contextual” resource=”EventType”/>
    </externalReferences>
    <role id=”rl2” semRole=”A0”>
    <!–by Volkswagen–>
      <externalReferences>
        <externalRef reference=”sell.01@0” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Agent” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Seller” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A1”>
    <!–of actions–>
      <externalReferences>
        <externalRef reference=”sell.01@1” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Theme” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Goods” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A2”>
    <!–to Porche–>
      <externalReferences>
        <externalRef reference=”sell.01@2” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Recipient” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Buyer” resource=”FrameNet”/>
      </externalReferences>
    </role>
  </predicate>
```

Table 12: Annotation example of the nominal predicate in English *“sale.01”*.

```
Volkswagen vende sus acciones a Porche.
  <predicate id=”pr1”>
  <!–vende–>
    <externalReferences>
      <externalRef reference=”vender.1.default” resource=”AnCora”/>
      <externalRef reference=”sell.01” resource=”PropBank”/>
      <externalRef reference=”give-13.1” resource=”VerbNet”/>
      <externalRef reference=”give-13.1-1” resource=”VerbNet”/>
      <externalRef reference=”Commerce_sell” resource=”FrameNet”/>
      <externalRef reference=”contextual” resource=”EventType”/>
    </externalReferences>
    <role id=”rl2” semRole=”A0”>
    <!–Volkswagen–>
      <externalReferences>
        <externalRef reference=”sell.01@0” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Agent” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Seller” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A1”>
    <!–sus acciones–>
      <externalReferences>
        <externalRef reference=”sell.01@1” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Theme” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Goods” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A2”>
    <!–a Porche–>
      <externalReferences>
        <externalRef reference=”sell.01@2” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Recipient” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Buyer” resource=”FrameNet”/>
      </externalReferences>
    </role>
  </predicate>
```

Table 13: Annotation example of the verbal predicate in Spanish *“vender.1.default”*.

```
Se aprueba la venta de acciones a Porche por parte de Volkswagen.
  <predicate id=”pr2”>
  <!–venta–>
    <externalReferences>
      <externalRef reference=”venta.1.default” resource=”AnCora”/>
      <externalRef reference=”sell.01” resource=”PropBank”/>
      <externalRef reference=”give-13.1” resource=”VerbNet”/>
      <externalRef reference=”give-13.1-1” resource=”VerbNet”/>
      <externalRef reference=”Commerce_sell” resource=”FrameNet”/>
      <externalRef reference=”contextual” resource=”EventType”/>
    </externalReferences>
    <role id=”rl2” semRole=”A0”>
    <!–por parte de Volkswagen–>
      <externalReferences>
        <externalRef reference=”sell.01@0” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Agent” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Seller” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A1”>
    <!–de acciones–>
      <externalReferences>
        <externalRef reference=”sell.01@1” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Theme” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Goods” resource=”FrameNet”/>
      </externalReferences>
    </role>
    <role id=”rl2” semRole=”A2”>
    <!–a Porche–>
      <externalReferences>
        <externalRef reference=”sell.01@2” resource=”PropBank”/>
        <externalRef reference=”give-13.1@Recipient” resource=”VerbNet”/>
        <externalRef reference=”Commerce_sell@Buyer” resource=”FrameNet”/>
      </externalReferences>
    </role>
  </predicate>
```

Table 14: Annotation example of the nominal predicate in Spanish *“venta.1.default”*.

2.1.7 Interoperability of Named Entities

In the second year of the project, we have not only focused on cross-lingual interoperability of predicate information. We now also provide cross-lingual interoperability of the named entities occurring in the text. Several studies in previous research focused on the integration of resources targeted at knowledge about nouns and named entities. Well know examples are YAGO [Suchanek *et al.*, 2007], Freebase [Bollacker *et al.*, 2008], DBpedia [Bizer *et al.*, 2009], BabelNet [Navigli and Ponzetto, 2010] or UBY [Gurevych *et al.*, 2012].

Among the available tools to work with this type of resource, DBpedia Spotlight [Daiber *et al.*, 2013][14] is a Wikification tool for automatically annotating mentions of DBpedia resources in text, providing a solution for linking unstructured information sources to the Linked Open Data cloud through DBpedia. The tool also offers the option of performing only Named Entity Disambiguation given previously detected spots by another engine. It furthermore provides functionalities and models to work with the four languages of the project.

DBpedia is the Linked Data version of Wikipedia. The DBpedia data set currently provides information about more than 4 million *things*, including at least 1,445,000 persons, 735,000 places, 241,000 organizations classified in a consistent ontology. According to the DBpedia website, localized versions of DBpedia are available in 125 languages.[15] All these versions together describe 38.3 million things, out of which 23.8 million are localized descriptions of things that also exist in the English version of DBpedia. In addition, the data set is interlinked with many other data sources from various domains (life sciences, media, geographic government, publications, etc.), including the aforementioned Freebase and YAGO, among many others.[16]

Thus, we consider DBpedia and DBpedia Spotlight two valuable resources to work with when establishing the interoperable semantic interpretation for English, Dutch, Spanish and Italian nouns and named entities. Based on the language of the input text, the corresponding DBpedia is used to perform the semantic annotation. That is, if we process English text, we use the English version of DBpedia. If we process Spanish text, we use the Spanish version. Obviously, the external references to DBpedia produced by the existing English and Spanish DBpedia spotlight modules are different. For instance, a mention to *New York* in an English document produces as external reference the identifier `http://dbpedia.org/page/New_York`. Similarly, a mention to *Nueva_York* in a Spanish document produces as external reference the identifier `http://es.dbpedia.org/page/Nueva_York`. Obviously, both identifiers are interoperable because there are crosslingual links between both DBpedia entries. Now, we have modified our non English named entity disambiguation modules to also include as external reference (if exist) the corresponding identifier for English. For example, our Spanish document also produces as external reference the identifier `http://dbpedia.org/page/New_York` for mentions to *Nueva_York*.

[14] `https://github.com/dbpedia-spotlight`

[15] `http://wiki.dbpedia.org/Datasets`, accessed 30 November 2014.

[16] `http://wiki.dbpedia.org/Datasets`

For that, in the second cycle of the project, we have updated the NED module (cf. Sections 3.8 and 6.8) so that the module suggest a list of candidates for each entity. In addition, for languages other than English, the module obtains DBpedia entries in the localized and English versions. This new feature allows us to start working with cross-lingual links. For instance, given the entity *Ford* (referring to the organization) in a Spanish text, the module would return the following information:

```
<entities>
  <entity id="e1" type="ORGANIZATION">
    <references>
      <!--Ford-->
      <span>
        <target id="t3" />
      </span>
    </references>
    <externalReferences>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Ford_Motor_Company" confidence="0.999962" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Ford_Motor_Company" confidence="0.999962" reftype="en" source="es↩
            " />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Henry_Ford" confidence="3.455495E-5" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Henry_Ford" confidence="3.455495E-5" reftype="en" source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Ford_World_Rally_Team" confidence="2.9344005E-6" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Ford_World_Rally_Team" confidence="2.9344005E-6" reftype="en" ↩
            source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          John_Ford_(director_de_cine)" confidence="2.0449986E-7" reftype="es" source=↩
          "es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/John_Ford" confidence="2.0449986E-7" reftype="en" source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Ford_Falcon_(Argentina)" confidence="1.8881913E-7" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Ford_Falcon_(Argentina)" confidence="1.8881913E-7" reftype="en" ↩
            source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Fundacion_Ford" confidence="6.853231E-8" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Ford_Foundation" confidence="6.853231E-8" reftype="en" source="es↩
            " />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Gerald_Ford" confidence="6.199239E-8" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Gerald_Ford" confidence="6.199239E-8" reftype="en" source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Aiden_Ford" confidence="8.774065E-9" reftype="es" source="es">
        <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/↩
            resource/Aiden_Ford" confidence="8.774065E-9" reftype="en" source="es" />
      </externalRef>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Ford_Motors" confidence="3.4108938E-9" reftype="es" source="es" />
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Ford_TC_2000" confidence="3.3651093E-9" reftype="es" source="es" />
    </externalReferences>
  </entity>
```

The NED module decides that *Ford* refers most likely to the `http://es.dbpedia.org/resource/Ford_Motor_Company` DBpedia entry, but it also returns a list of candidates. In addition, the module shows the corresponding English DBpedia entry. As we already mentioned, this type of information is used used to link the crosslingual realizations of entities in different languages.

3 English NLP Processing

This section describes the English pipeline. Descriptions of the modules included in the pipeline are provided along with some technical information. This technical information includes: a) the description of the input and output that the modules require and obtain; b) the dependencies with other modules and third-party modules and libraries; c) level of operation of the module; d) if the module is language dependent or not; e) the required resources for a correct functioning of the module; f) the possible formats the module works with and feasibility of adapting the module to other formats g) the github address of the module. We will first describe the individual modules of the pipeline. An overview of the complete pipeline is provided at the end in Section 3.16. A complete specification of the output attributes and elements is provided in Appendix A.

3.1 Tokenizer

- **Module**: ixa-pipe-tok

- **Description of the module**: This module provides Sentence Segmentation and Tokenization for English and other languages such as Dutch, German, French, Italian and Spanish. It implements a rule-based segmenter and tokenizer originally inspired by the Stanford English Penn Treebank tokenizer[17] but with several modifications and improvements. These include tokenization for other languages such as Italian, normalization according the Spanish Ancora Corpus [Taulé *et al.*, 2008], paragraph treatment, and more comprehensive gazetteers of non breaking prefixes. The tokenizer depends on a JFlex[18] specification file which compiles in seconds and performs at a very reasonable speed (around 250K word/second, and much quicker with Java multithreading). JFlex is a lexical analyser generator. The module is part of the IXA pipes [Agerri *et al.*, 2014],[19] a modular set of Natural Language Processing tools

[17] `http://www-nlp.stanford.edu/software/tokenizer.shtml`

[18] `http://jflex.de/`

[19] `http://ixa2.si.ehu.es/ixa-pipes/`

(or pipes) which provide easy access to NLP technology for English and Spanish. An illustration of its output is provided in Appendix A.1.

- **Input**: Raw text
- **Input representation**: NAF raw layer
- **Output**: Tokens and sentences.
- **Output representation**: NAF text layer
- **Required modules**: None
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: None
- **Dependencies**: Java, Maven, NAF Java library, JFlex
- **Flexible in- and output**: It takes plain text or raw text in NAF. It produces tokenized and segmented text in NAF, running text and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

3.2 POS tagging

- **Module**: ixa-pipe-pos
- **Description of the module**: This module provides POS tagging and lemmatization for English and Spanish. The module is part of the IXA pipes. We have obtained the best results so far with *Perceptron* models and the same featureset as in [Collins, 2002]. The models have been trained and evaluated on the WSJ treebank using the usual partitions (as explained in [Toutanova *et al.*, 2003]). We currently obtain a performance of 96.88% vs 97.24% in word accuracy obtained by [Toutanova *et al.*, 2003].

  Lemmatization is currently performed via 3 different dictionary lookup methods: a) *Simple Lemmatizer*: The simple lemmatizer is based on HashMap lookups on a plain text dictionary. Currently we use dictionaries from the LanguageTool project[20] under their distribution licenses; b) *Morfologik-stemming*:[21] The Morfologik library provides routines to produce binary dictionaries, from dictionaries such as the one used by the Simple Lemmatizer above, as finite state automata. This method is

[20] `http://languagetool.org/`

[21] `https://github.com/morfologik/morfologik-stemming`

convenient whenever lookups on very large dictionaries are required because it reduces the memory footprint to 10% of the memory required for the equivalent plain text dictionary; and c) *WordNet Lookup*: We also provide lemmatization based on lookup in WordNet-3.0 [Fellbaum, 1998] via the JWNL API.[22] By default, the module accepts tokenized text in NAF format as standard input and outputs NAF (see Appendix A.2).

- **Input**: Tokens
- **Input representation**: NAF text layer
- **Output**: Lemmas and POS-tags
- **Output representation**: NAF terms layer
- **Required modules**: Tokenizer module
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: POS model; Lemmatizer dictionaries: plain text dictionary and morfologik-stemming dictionary.
- **Dependencies**: Java, Maven, NAF Java library, JWNL API, Apache OpenNLP.
- **Flexible in- and output**: It accepts tokenized text in NAF. It outputs NAF or CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

3.3 Constituency Parser

- **Module**: ixa-pipe-parse
- **Description of the module**: This module provides statistical constituent parsing for English and Spanish. The module is part of the IXA pipes. Maximum Entropy models are trained to build shift reduce bottom up parsers [Ratnaparkhi, 1999] as provided by the Apache OpenNLP Machine Learning API. Parsing models for English have been trained using the Penn treebank. Furthermore, ixa-pipe-parse provides two methods of headword finders: one based on Collins's head rules as defined in his PhD thesis [Collins, 1999], and another one based on Stanford's parser Semantic Head Rules.[23] The latter are a modification of Collins's head rules according to lexical and semantic criteria. We obtain a F1 87.42%. The module accepts lemmatized and POS tagged text in NAF format as standard input and outputs NAF (see Appendix A.8).

[22] `http://jwordnet.sourceforge.net/`

[23] `http://www-nlp.stanford.edu/software/lex-parser.shtml`

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Constituents; Syntactic tree of sentences.
- **Output representation**: NAF constituency layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: Parsing model
- **Dependencies**: Java, Maven, NAF Java library, Apache OpenNLP
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. In addition to NAF output, ixa-pipe-parse also can allows output the parse trees into Penn Treebank bracketing style.
- **github address**: `https://github.com/newsreader/ixa-pipes`

3.4 Dependency Parser

- **Module**: ixa-pipe-srl
- **Description of the module**: This module is based on the MATE-tools [Björkelund *et al.*, 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. The core component of the module is the MATE-tools and the module provides a wrapper around it. As the input of the module is a NAF file that includes lemmatization and POS-tagging, the module only implements the dependency parser [Bohnet, 2010]. The module is ready to work with Spanish and English. For the latter, the dependency parser had the top score in the CoNLL shared task 2009, obtaining 90.24% labeled attachment score (LAS). The output is illustrated in Appendix A.9.
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Dependencies
- **Output representation**: NAF deps layer
- **Required modules**: Tokenizer and POS tagger modules

- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: mate-tools package, dependency parsing model[24]
- **Dependencies**: Java, Maven, NAF Java library, mate-tools
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. The modules can output dependencies trees in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipe-srl`

3.5 Time expression detection and normalization

- **Module**: fbk-timepro
- **Description of the module**: TimePro identifies the tokens corresponding to temporal expressions in English, assigns them to one of the 4 TIMEX classes defined in ISO-TimeML and normalizes them following TIDES specification ([Ferro *et al.*, 2002]). The temporal expressions recognizer is based on machine learning and it is trained on TempEval3 data. The average result for English is: 83.81% precision, 75.94% recall and 79.61% F1-measure values. The temporal expressions normalizer uses the library timenorm ([Bethard, 2013]), enhanced by some pre-processing and post-processing for the selection of the best normalization value. Timenorm is shown to be the best performing system for most evaluation corpora (it obtained 81.6% F1-measure on TempEval3 test corpus) compared with other systems such as HeidelTime ([Strötgen *et al.*, 2013]). This module has been integrated into TextPro pipeline but it also accepts a NAF input file with tokenization, POS-tagging and chunking information and provides a NAF file as output (see Appendix A.13).
- **Input**: token layer, term layer with lemma, POS, entity and constituents
- **Input representation**: NAF terms, entities and constituency layers
- **Output**: timex3
- **Output representation**: NAF time expression layer
- **Required modules**: tokenizer, POS-tagger, NERC, Parser
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language model, language dependent rules, English grammar for timenorm

[24] The module use additional resources to perform the semantic role labeling.

- **Dependencies**: timenorm ([Bethard, 2013]), YamCha[25]
- **Flexible in- and output**: can provide NAF and TextPro format (i.e. column format)
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/TimePro`

3.6 Named Entity Recognition and Classification

- **Module**: ixa-pipe-nerc
- **Description of the module**: This module is a multilingual Named Entity Recognition and Classification tagger. ixa-pipe-nerc is part of IXA pipes. The named entity types are based on: a) the CONLL 2002[26] and 2003[27] tasks which focused on language-independent supervised named entity recognition for four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. We provide very fast models trained on local features only (84.53 F1), similar to those of [Zhang and Johnson, 2003] with several differences: We do not use POS tags, chunking or gazetteers in our baseline models but we do use bigrams, trigrams and character ngrams. We also provide some models with external knowledge (87.11 F1); b) the Ontonotes 4.0 dataset. We have trained our system on the full corpus with the 18 NE types, suitable for production use. We have also used 5K sentences at random for testset from the corpus and leaving the rest (90K aprox) for training. The Ontonotes CoNLL 4 NE types with local features model obtains F1 86.21. The Ontonotes 3 NE types with local features configuration obtains F1 89.41. The module can format its output in CoNLL style tabulated BIO format as specified in the CoNLL 2003 shared evaluation task. The output is illustrated in Appendix A.6.
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Named entities
- **Output representation**: NAF entities layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes

[25] `http://chasen.org/~taku/software/yamcha/`
[26] `http://www.clips.ua.ac.be/conll2002/ner/`
[27] `http://www.clips.ua.ac.be/conll2003/ner/`

- **Resources**: CoNLL 2003 models, Ontonotes 4.0 models, properties file
- **Dependencies**: Java, Maven, NAF Java library, Apache OpenNLP
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. The modules can output both NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

3.7 Word Sense Disambiguation

- **Module**: wsd-ukb
- **Description of the module**: UKB is a collection of programs for performing graph-based Word Sense Disambiguation. UKB applies the so-called Personalized PageRank on a Lexical Knowledge Base (LKB) to rank the vertices of the LKB and thus perform disambiguation. UKB has been developed by the IXA group. The module has been evaluated on the general domain coarse grained all-words datasets (S07CG) [Navigli *et al.*, 2007]. The overall result obtained is F1 80.1. An analysis of the performance according to the POS shows that this module performs better particularly on nouns, obtaining F1 83.6 (results for the rest of POS: 71.1 for verbs, 83.1 for adjectives and 82.3 for adverbs). The module accepts lemmatized and POS tagged text in NAF format as standard input and outputs NAF (see Appendix A.4).
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Synsets
- **Output representation**: NAF terms layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: English WordNet
- **Dependencies**: C++, boost libraries
- **Flexible in- and output**: no
- **github address**: `https://github.com/ixa-ehu/ukb`

3.8 Named Entity Disambiguation

- **Module**: ixa-pipe-ned
- **Description of the module**: This module performs the Named Entity Disambiguation task based on DBpedia Spotlight. Assuming that a DBpedia Spotlight Rest server for a given language is locally running, the module will take NAF as input (containing elements) and perform Named Entity Disambiguation. The module accepts text with named entities in NAF format as standard input, it disambiguates them and outputs them in NAF. The module offers the "disambiguate" and "candidates" service endpoints. The disambiguate service takes the spotted text input and it returns the identifier for each entity. The candidates service is similar to disambiguate, but returns a ranked list of candidates. For the evaluation of the module, we used the 2010 and 2011 datasets from the TAC KBP editions and the AIDA corpus. Because we focus our study on NED systems, we discard the so-called NIL instances (instances for which no correct entity exists in the Reference Knowledge Base) from the datasets. As the module has several parameters, it was optimized in TAC 2010 dataset. Using the best parameter combination, the module has been evaluated on two datasets: TAC 2011 and AIDA. The best results obtained on the first dataset were 79.77 in precision and 60.68 in recall. The best performance on the second dataset is 79.67 in precision and 75.94 in recall. The output of this module is illustrated in Appendix A.7.
- **Input**: Named entities and sentences
- **Input representation**: NAF entities layer[28]
- **Output**: Disambiguated named entities
- **Output representation**: NAF entities layer
- **Required modules**: NERC module
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: DBpedia spotlight server
- **Dependencies**: Java, Maven, NAF Java library, DBpedia spotlight
- **Flexible in- and output**: no
- **github address**: `https://github.com/newsreader/ned-spotlight`

[28] Note: Linguistic annotations of a particular level always span elements of previous levels. In this particular case, the module also uses the terms layer to obtain the sentences of the given entities.

3.9 Coreference Resolution

- **Module**: corefgraph
- **Description of the module**: The module of coreference resolution included in the IXA pipeline is loosely based on the Stanford Multi Sieve Pass system [Lee *et al.*, 2013]. The system consists of a number of rule-based sieves. Each sieve pass is applied in a deterministic manner, reusing the information generated by the previous sieve and the mention processing. The order in which the sieves are applied favours a highest precision approach and aims at improving the recall with the subsequent application of each of the sieve passes. This is illustrated by the evaluation results of the CoNLL 2011 Coreference Evaluation task [Lee *et al.*, 2013; Lee *et al.*, 2011], in which the Stanford's system obtained the best results. The results show a pattern which has also been shown in other results reported with other evaluation sets [Raghunathan *et al.*, 2010], namely, the fact that a large part of the performance of the multi pass sieve system is based on a set of significant sieves. Thus, this module focuses for the time being, on a subset of sieves only, namely, Speaker Match, Exact Match, Precise Constructs, Strict Head Match and Pronoun Match [Lee *et al.*, 2013]. So far we have evaluated our module on the dev-auto part of the Ontonotes 4.0 corpus. We score 56.4 CoNLL F1, around 3 points worse than Stanford's system. Appendix A.12 illustrates the output representation of this module.
- **Input**: lemma, morphosyntactic information (morphofeat in NAF), named-entities, constituents
- **Input representation**: NAF entities, term, and constituency layers
- **Output**: coreferences
- **Output representation**: NAF coreferences layer
- **Required modules**: Tokenizer, POS-tagger and NERC modules
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: none
- **Dependencies**: pyKAF, pycorpus, networkx, pyYALM
- **Flexible in- and output**: It accepts lemmatized and POS tagged text, entities and constituents in NAF format. The modules can output coreference clusters in NAF and CoNLL formats.
- **github address**: `https://bitbucket.org/Josu/corefgraph`

3.10 Semantic Role Labeling

- **Module**: ixa-pipe-srl
- **Description of the module**: This module is based on the MATE-tools [Björkelund *et al.*, 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. They report on the CoNLL 2009 Shared Task a labelled semantic F1 of 85.63 for English and 79.91 for Spanish. As the input of the module is a NAF file that includes lemmatization, POS-tagging and dependency parsing, the module only implements the semantic role labeler [Björkelund *et al.*, 2009]. The module is ready to work with Spanish and English. By default, the module accepts parsed text in NAF format as standard input and outputs the enriched text in NAF. Original the output annotations are based on PropBank/NomBank or AnCora, but the module makes use of the *Predicate Matrix* as an external resource to enrich the semantic information of the annotation, including both for predicates and arguments their correspondances in FrameNet, VerbNet and, in case of spanish or nominal predicates, their sources in PropBank. The output representation of the semantic role labeler is illustrated in Appendix A.9.
- **Input**: Lemmatized and POS tagged text and syntactic dependencies
- **Input representation**: NAF terms, deps layers
- **Output**: Semantic roles
- **Output representation**: NAF SRL layer
- **Required modules**: Tokenizer, POS tagger and Dependency parsing modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: mate-tools package, PredicateMatrix
- **Dependencies**: Java, Maven, NAF Java library, mate-tools
- **Flexible in- and output**: It accepts lemmatized and POS tagged text and syntactic dependencies in NAF format. The modules can output semantic roles in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipe-srl`

3.11 Event coreference

- **Module**: vua-eventcoreference
- **Description of the module**: This module takes the predicates of the SRL layer as input and matches the predicates semantically. If the predicates are sufficiently similar, a coreference set is created with references to the predicates as coreferring expressions. If there is no match, predicates form a singleton set in the coreference layer. An example of the exact output can be found in Appendix A.12.
- **Input**: SRL
- **Input representation**: SRL predicates
- **Output**: Coreference sets for events
- **Output representation**: NAF coref layer
- **Required modules**: ixa-pipe-srl
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: wordnet-lmf
- **Dependencies**: Java, Maven, KAF/NAF Saxparser, WordnetTools
- **Flexible in- and output**: no: this module only works with KAF/NAF.
- **github address**: `https://github.com/cltl/EventCoreference`

3.12 Temporal relation extraction

- **Module**: fbk-temprel
- **Description of the module**: TempRelPro extracts and classifies temporal relations between two events, two time expressions, or an event and a time expression ([Mirza and Tonelli, 2014b]). The module is based on machine learning and is trained using yamcha tool on the TempEval3 data. It detects relations between the following: a) the document creation time and the main event of each sentence; b) the main events of two consecutive sentences; c) the time expressions and the events inside a sentence; d) all the events inside a sentence. All the events annotated by the Coreference module are considered and all the time expressions identified by the TimePro module. The result for relation classification (identification of the relation type given the relations) on TempEval3 test corpus is: 58.8% precision, 58.2% recall and 58.5% F1-measure. This module is part of TextPro pipeline, a multilingual NLP pipeline developed at FBK. Appendix A.14 provides an example of the module's output.

- **Input**: token layer, term layer with lemma, POS, entity, constituent, SRL, event coreference and time expression

- **Input representation**: NAF terms, entities, constituency, SRL, coref and time expression layers

- **Output**: tlink

- **Output representation**: NAF temporal relation layer

- **Required modules**: tokenizer, POS-tagger, NERC, Parser, ixa-pipe-srl, vua-eventcoreference, fbk-time

- **Level of operation**: document level

- **Language dependent**: yes

- **Resources**: language model, language dependent rules

- **Dependencies**: Explicit Discourse Connectives Tagger ([Pitler and Nenkova, 2009]), MorphoPro, YamCha[29]

- **Flexible in- and output**: use also TextPro format (i.e. column format)

- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/TempRelPro/`

3.13 Causal relation extraction

- **Module**: fbk-causalrel

- **Description of the module**: CausalRelPro extracts explicit causal relations between two events in the same sentence ([Mirza and Tonelli, 2014a]). All the events annotated by the Coreference module are considered. The module is based on machine learning and is trained using yamcha tool on the TimeBank corpus manually enriched with causal information (causal signals and causal relations). One model is trained for the annotation of causal signals (e.g. *as a result of*, *due to*) and another for the extraction of the causal links between events. The evaluation done on the test part of the corpus for the task of causal relation extraction gave a precision of 67.3%, a recall of 22.6% and a F1 measure of 33.9%. This module is part of TextPro pipeline, a multilingual NLP pipeline developed at FBK. An example of the NAF output is provided in Appendix A.16.

- **Input**: token layer, term layer with lemma, POS, entity, constituent, SRL, event coreference, time expression and temporal relation

[29] `http://chasen.org/~taku/software/yamcha/`

- **Input representation**: NAF terms, entities, constituency, SRL, coref, time expression and temporal relation layers
- **Output**: tlink
- **Output representation**: NAF causal relation layer
- **Required modules**: tokenizer, POS-tagger, NERC, Parser, ixa-pipe-srl, vua-eventcoreference, fbk-time, fbk-temprel
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language model, language dependent rules
- **Dependencies**: Explicit Discourse Connectives Tagger ([Pitler and Nenkova, 2009]), MorphoPro, YamCha[30]
- **Flexible in- and output**: can also use the TextPro format (i.e. column format)
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/CausalRelPro/`

3.14 Factuality

- **Module**: VUA-factuality
- **Description of the module**: The factuality module classifies for each event whether it is factual or not. The module uses the MAchine Learning for LanguagE Toolkit Mallet [McCallum, 2002] (version 2.0.7) trained on FactBank v1.0[31] to determine the factuality of an event in text. A new factuality module is currently under development which will provide the same information as the module currently used for Italian (see Appendix A.17). The output of the module described here is illustrated in Appendix A.18.
- **Input**: Tokenized and POS-tagged text
- **Input representation**: NAF terms layer
- **Output**: Factuality layer with token spans
- **Output representation**: NAF factuality layer
- **Required modules**: tokenizer, POS-tagger modules

[30] `http://chasen.org/~taku/software/yamcha/`

[31] `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T23`

- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: FactBank, Mallet
- **Dependencies**: Perl
- **Flexible in- and output**: no
- **github address**: `https://github.com/newsreader/Factuality-Classifier`

3.15 Opinions

- **Module**: opinion-miner
- **Description of the module**: This is a module that detects and extracts fine-grained opinions, where one single opinion contains three elements: the opinion expression (the subjective statement itself), the opinion target (what the opinion is about) and the opinion holder (who is stating the opinion). This module has been developed during the OpeNER project,[32] where it has been trained on different domains (hotel reviews, political news...), and for different languages (Dutch, English, Spanish, Italian, French and German) using corpora that were annotated manually also during the project. The extraction and tagging of opinions is divided into two steps. First, opinion entities (holder, target and expression) are detected using Conditional Random Fields, where tokens, lemmas and POS of the context are used as features, as well as syntactic features and features extracted from lexicons. The second step is the opinion entity linking (expression−target and expression−holder) using binary Support Vector Machines. In this step, all the single opinion entities detected are grouped into triples <expression,target,holder> according to the output of the SVM classifiers. In this case, besides the local context features, dependency features and features capturing the relative location of the opinion elements are included. The models have been trained with a rich set of features, but the opinion tagger can be used with a reduced subset of this features considering that the performance will be affected. The output representation is illustrated in Appendix A.19.
- **Input**: NAF text processed trough the pipeline, this module will use the information provided by all the rest
- **Input representation**: token, term, entity, dependency and constituency NAF layers
- **Output**: fine-grained opinion triples <expression, target, holder>
- **Output representation**: NAF opinion layer

[32] `http://www.opener-project.eu/`

- **Required modules**: tokenizer, lemmatizer, POS tagger, polarity tagger, named entity tagger, constituency parser and dependency parser
- **Level of operation**: document
- **Language dependent**: yes
- **Resources**: models trained on the OpeNER project
- **Dependencies**: CRF library,[33] SVM-Light library,[34] KafNafParser[35] and VUA_pylib library[36]
- **Flexible in- and output**: it takes NAF or KAF input text, and generates NAF or KAF output text enriched with the extracted opinions
- **github address**: `https://github.com/cltl/opinion_miner_deluxe`

3.16 English Pipeline

This section presents the English pipeline used to detect events in the second year of the project. The pipeline is described in two different ways. First, the dependencies among each task (module) are presented in Figure 1. Note that the current modules inherit the dependencies of previous modules. Then, the order of the linguistic processors is displayed in Figure 2.

[33] http://www.chokkan.org/software/crfsuite/
[34] http://www.chokkan.org/software/crfsuite/
[35] https://github.com/cltl/KafNafParserPy
[36] https://github.com/cltl/VUA_pylib

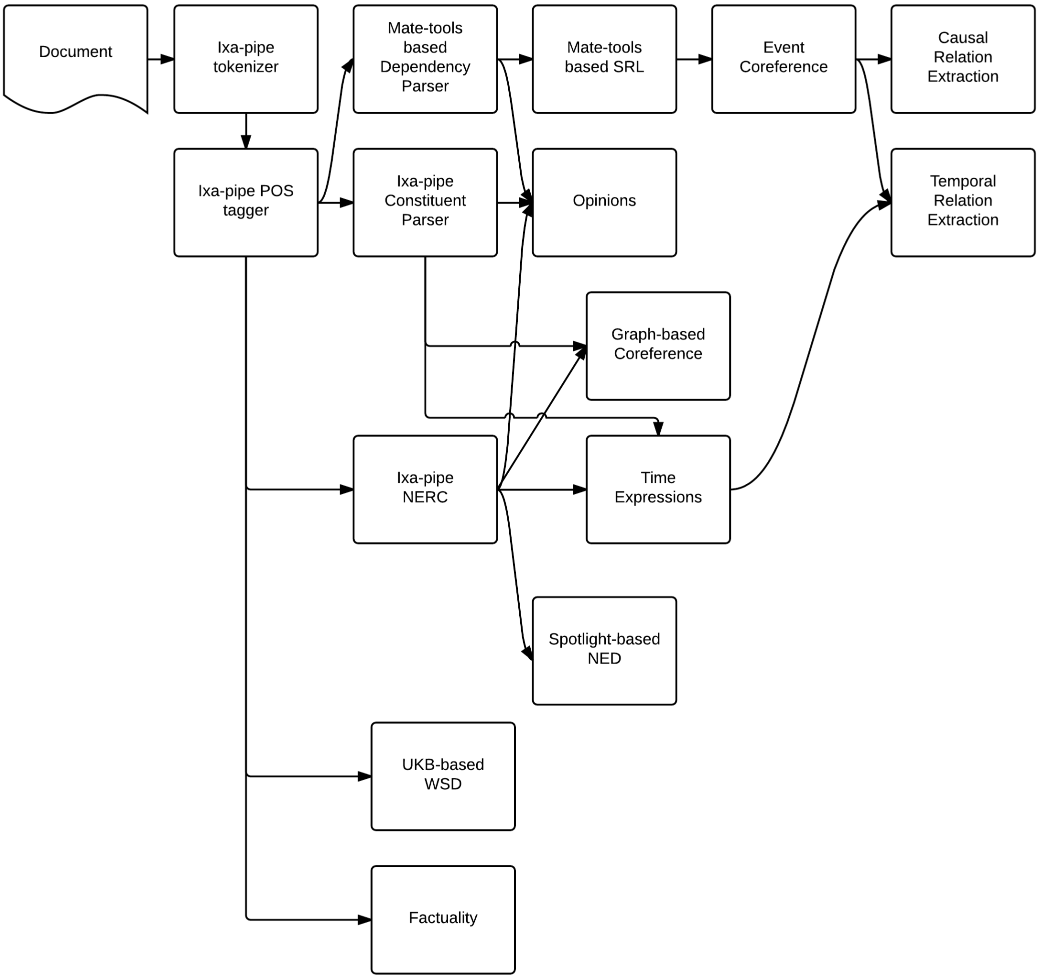

Figure 1: NewsReader pipeline. Dependencies among modules

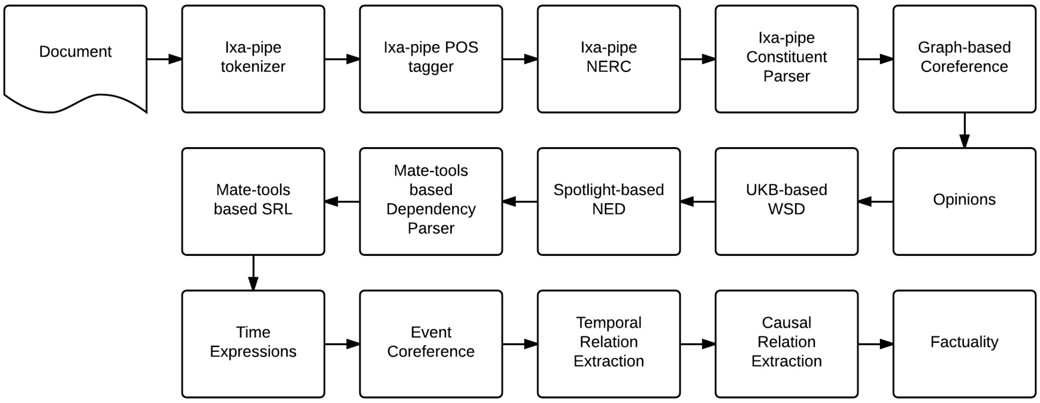

Figure 2: NewsReader pipeline. Linguistic processors' order

4 Dutch NLP Processing

The description of modules for the Dutch pipeline are structured similary to the English descriptions. Each description is accompanied with some technical information. This technical information includes: a) the description of the input and output that the modules require and obtain; b) the dependencies with other modules and third-party modules and libraries; c) level of operation of the module; d) if the module is language dependent or not; e) the required resources for a correct functioning of the module; f) the possible formats the module works with and g) the github address of the module. When output attributions and elements differ from the output of English modules for the same task, a complete specification is provided in Appendix A. After the description of individual modules in the pipeline, we will provide an overview of the complete Dutch pipeline in Section 4.13.

4.1 Tokenizer

- **Module**: ixa-pipe-tok
- **Description of the module**: This module provides Sentence Segmentation and Tokenization for English and Spanish and other languages such as Dutch, German, English, French, Italian and Spanish. It implements a rule-based segmenter and tokenizer originally inspired by the Stanford English Penn Treebank tokenizer[37] but with several modifications and improvements. These include tokenization for other languages such as Italian, normalization according the Spanish Ancora Corpus [Taulé *et al.*, 2008], paragraph treatment, and more comprehensive gazeteers of non breaking prefixes. The tokenizer depends on a JFlex[38] specification file which compiles in sec-

[37] http://www-nlp.stanford.edu/software/tokenizer.shtml

[38] http://jflex.de/

onds and performs at a very reasonable speed (around 250K word/second, and much quicker with Java multithreading). JFlex is a lexical analyser generator. The module is part of the IXA pipes [Agerri *et al.*, 2014],[39] a modular set of Natural Language Processing tools (or pipes) which provide easy access to NLP technology for English and Spanish. It was originally developed as part of the aforementioned OpeNER project. The output looks the same as the output for English and is illustrated in Appendix A.1.

- **Input**: Raw text
- **Input representation**: NAF raw layer
- **Output**: Tokens and sentences.
- **Output representation**: NAF text layer
- **Required modules**: None
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: None
- **Dependencies**: Java, Maven, NAF Java library, JFlex
- **Flexible in- and output**: It takes plain text or raw text in NAF. It produces tokenized and segmented text in NAF, running text and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

4.2 POS tagging, lemmatization and parsing

- **Module**: vua-alpino
- **Description of the module**: This module performs morphosyntactic and dependency anaylisis of Dutch text. It is based on the Alpino parser,[40] which is a dependency and constituency parser for Dutch. Therefore this module is a wrapper around the Alpino parser to deal with NAF files as input and output formats. Some special features of Alpino have been used in order to reduce the time of processing. It basically means that the parser is run just once to generate the XML files containing Alpino output, and then these files are processed by different extractors to obtain all the information including POS-tags, morphosyntactic information, constituents and dependencies. The output provided by Alpino slightly differs from the output of other

[39] `http://ixa2.si.ehu.es/ixa-pipes/`

[40] `http://www.let.rug.nl/vannoord/alp/Alpino/`

POS-taggers (different tags and morphological features) and parsers (notably richer dependencies and different decisions in structure). Examples of what the output looks like in NAF is provided in Appendix A.11.

- **Input**: Tokens
- **Input representation**: NAF text layer
- **Output**: Lemmas and POS-tags, constiuent and dependency trees
- **Output representation**: NAF term, constituency and dependency layer
- **Required modules**: Tokenizer module
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: None
- **Dependencies**: Python, NAF python parser and Alpino parser version of 17 June 2014 or later. Note that this version only runs on Linux.
- **Flexible in- and output**: Alpino is software developed by a third party. For more information on possible in- and output, checkout the Alpino website at `http://www.let.rug.nl/vannoord/alp/Alpino/`. Our wrapper accepts tokenized text in NAF or KAF. It outputs NAF or KAF formats.
- **github address**: Wrapper: `https://github.com/cltl/morphosyntactic_parser_nl`, Alpino (git address): `git://urd.let.rug.nl/Alpino.git`

4.3 Time Expressions

- **Module**: VUA-HeidelTime
- **Description of the module**: This module identifies time expressions in text and normalizes them. The core component of the module is HeidelTime [Strötgen and Gertz, 2013], a temporal tagger supporting English, German, Dutch, Vietnamese, Arabic, Spanish, Italian, French, Chinese and Russian.[41] HeidelTime identifies temporal expressions based on language specific patterns. Identified temporal expressions are normalized and represented according to TIMEX annotations [Sundheim, 1996]. Dutch patterns have been written by Matje van de Camp for a corpus of short biographies [van de Camp and Christiansen, 2013]. HeidelTime is open source and can be used as a standalone module using TreeTagger or it can be integrated in UIMA architectures. The current version of the module provides a NAF wrapper around

[41] HeidelTime source code is available here: `https://code.google.com/p/heideltime/`

HeidelTime's standalone version. It can take the raw text layer or NAF token layer as input, calls HeidelTime standalone and creates a time expression layer as output. The disadvantage of this approach is that the standalone version requires running TreeTagger to obtain POS-tags and lemmas, which is freely available but cannot be redistributed. Moreover, our vua-alpino module already provides POS-tags and lemmas. An alternative version of the module that uses the NAF term layer as input is currently under development. The output of time expressions for Dutch is modeled on the English representations (see Appendix A.13).

- **Input**: Raw text or tokens
- **Input representation**: NAF raw-text layer or NAF token layer
- **Output**: Normalized time expressions in timex
- **Output representation**: NAF time expression layer
- **Required modules**: Tokenizer producing NAF or KAF
- **Level of operation**: Document level
- **Language dependent**: yes (language specific rules and patterns)
- **Resources**: TreeTagger with resources for Dutch, language specific rules
- **Dependencies**: Python 2.7, TreeTagger for the correct operating system
- **Flexible in- and output**: HeidelTime is independently developed software that can either take raw text as input and produce marked-up text as output or be integrated in UIMA pipelines. Our wrapper can use NAF or KAF as in- and output.
- **github address**: Wrapper: `https://github.com/cltl/NAF-HeidelTime`

4.4 Named Entity Recognition and Classification

- **Module**: ixa-pipe-nerc
- **Description of the module**: This module is multilingual Named Entity Recognition and Classification tagger which is part of IXA pipes. The named entity types are based on the CONLL 2002[42] and 2003[43] tasks which were focused on language-independent supervised named entity recognition for four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. We provide very fast models trained on local features only (84.53 F1), similar to those of Zhang and Johnson (2003) with several

[42] `http://www.clips.ua.ac.be/conll2002/ner/`

[43] `http://www.clips.ua.ac.be/conll2003/ner/`

differences: POS tags, chunking or gazetteers are not used in our baseline models but we do use bigrams, trigrams and character ngrams. We also provide some models with external knowledge (87.11 F1); Furthermore, the Ontonotes 4.0 dataset has been also considered. We have trained our system on the full corpus with the 18 NE types, suitable for production use. We have also used 5K sentences at random for testset from the corpus and leaving the rest (90K aprox) for training. The Ontonotes CoNLL 4 NE types with local features model obtains F1 86.21. The Ontonotes 3 NE types with local features configuration obtains F1 89.41. The module can format its output in CoNLL style tabulated BIO format as specified in the CoNLL 2003 shared evaluation task.The module produces the same output as the English module (Appendix A.6).

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF token and term layers
- **Output**: Named entities
- **Output representation**: NAF entity layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: CoNLL 2003 models, Ontonotes 4.0 models, properties file
- **Dependencies**: Java, Maven, NAF Java library, Apache OpenNLP
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. The modules can output dependencies trees in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

4.5 Word Sense Disambiguation

- **Module**: vua-wsd
- **Description of the module**: WSD-VUA is a machine learning system that performs Word Sense Disambiguation for Dutch text. It is based on a supervised machine learning approach: Suppor Vector Machines. The library selected for the low level machine learning engine is lib-svm.[44] A bag-of-words feature model is used for representing the training instances, where tokens and lemmas are considered to build the

[44] `https://github.com/cjlin1/libsvm`

context for each target word. The training material used has been the corpus manually annotated on the DutchSemCor project.[45] One classifier is build for each target lemma, following the one-vs-all approach to deal with the multilabel classification of whe WSD task, as SVM is in principle a binary classifier. The relative frequency of a feature with respect to a certain classifier is considered to filter out to general features, and also for weighting each feature in the training phase. This classifier was evaluated by Cross-fold validation reaching an accuracy of 82.51 for nouns, 84.80 for verbs and 73.62 for adjectives. The output adds external references to the term layer as illustrated in Appendix A.3.

- **Input**: tokenized, lemmatized and POS tagged text
- **Input representation**: NAF token and term layers
- **Output**: word sense labels assigned to terms
- **Output representation**: NAF external references in the term layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: models trained on DutchSemCor
- **Dependencies**: lib-svm and KafNafParser python libraries
- **Flexible in- and output**: it accepts tokenized, lemmatized and POS tagged text in KAF/NAF format or plain text. The output can be KAF/NAF format of XML format as used in the SemCor corpus.
- **github address**: `https://github.com/cltl/svm_wsd`

4.6 PredicateMatrix tagging

- **Module**: vua-ontotagging
- **Description of the module**: This module takes a predicate matrix file with synsets and predicate matrix mappings and adds the predicate matrix mappings to the synsets listed in the term layer. A more detailed description of the idea behind this module can be found in Section 2. An illustration of the exact output can be found in Appendix A.5.
- **Input**: terms with word sense disambiguated output

[45] `http://www2.let.vu.nl/oz/cltl/dutchsemcor`

- **Input representation**: term layer
- **Output**: terms with predicate matrix mappings
- **Output representation**: term layer with external references
- **Required modules**: vua-wsd
- **Level of operation**: term level
- **Language dependent**: yes
- **Resources**: PredicateMatrix
- **Dependencies**: Java, Maven, KAF/NAF Saxparser
- **Flexible in- and output**: The module takes NAF input stream and NAF output stream only, but the Predicate Matrix can easily be integrated in other pipelines.
- **github address**: `git@github.com:cltl/OntoTagger`

4.7 Named Entity Disambiguation

- **Module**: ixa-pipe-ned
- **Description of the module**: This module performs the Named Entity Disambiguation task based on DBpedia Spotlight. Assuming that a DBpedia Spotlight Rest server for a given language is locally running, the module will take NAF as input and will perform Named Entity Disambiguation. The module accepts text with named entities in NAF format as standard input, it disambiguates them and outputs them in NAF. The output of this module corresponds to the English output illustrated in Appendix A.7.
- **Input**: Named entities and sentences
- **Input representation**: NAF entities layer[46]
- **Output**: Disambiguated named entities with dbpedia links
- **Output representation**: DBpedia links on the NAF entity layer
- **Required modules**: NERC module
- **Level of operation**: sentence level
- **Language dependent**: yes

[46] Note: Linguistic annotations of a particular level always span elements of previous levels. In this particular case, the module also uses the terms layer to obtain the sentences of the given entities.

- **Resources**: DBpedia spotlight server
- **Dependencies**: Java, Maven, NAF Java library, DBpedia spotlight
- **Flexible in- and output**: The module is a wrapper around DBpedia spotlight. The wrapper only handles NAF/KAF, but other wrappers can easily be created for DBpedia spotlight.
- **github address**: `https://github.com/newsreader/ned-spotlight`

4.8 Nominal Coreference Resolution

- **Module**: CorefGraph
- **Description of the module**: The module of coreference resolution included in the IXA pipeline that is used for English is also used for Dutch in an adapted form. For convenience, we repeat the description provided above before adding a few remarks on adaptations for Dutch. The coreference module is loosely based on the Stanford Multi Sieve Pass system [Lee *et al.*, 2013]. The system consists of a number of rule-based sieves. Each sieve pass is applied in a deterministic manner, reusing the information generated by the previous sieve and the mention processing. The order in which the sieves are applied favours a highest precision approach and aims at improving the recall with the subsequent application of each of the sieve passes. This is illustrated by the evaluation results of the CoNLL 2011 Coreference Evaluation task [Lee *et al.*, 2013; Lee *et al.*, 2011], in which the Stanford's system obtained the best results. The results show a pattern which has also been shown in other results reported with other evaluation sets [Raghunathan *et al.*, 2010], namely, the fact that a large part of the performance of the multi pass sieve system is based on a set of significant sieves. Thus, this module focuses for the time being, on a subset of sieves only, namely, Speaker Match, Exact Match, Precise Constructs, Strict Head Match and Pronoun Match [Lee *et al.*, 2013]. The algorithm uses constituent trees and morphosyntactic information to determine coreference. We have adapted the original implementation so that it can deal with the richer morphosyntactic output of Alpino that our current pipeline for Dutch requires. Furthermore, constituent trees provided by Alpino are structured differently from those provided by the Stanford parser (which is used as the basis for implementing this algorithm). Minor adaptations were required to process Alpino's constituent trees. The structure of the output is similar to English, illustrated in Appendix A.12 .
- **Input**: Lemma, morphosyntactic information, named-entities, constituents
- **Input representation**: NAF entities, constituency and term layers.
- **Output**: coreferences
- **Output representation**: NAF coreference layer

- **Required modules**: Tokenizer, Alpino POS tagger, lemmatization and parsing and NERC modules
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: none
- **Dependencies**: pyKAF, pycorpus, networkx, pyYALM
- **Flexible in- and output**: The module only takes KAF and NAF input and would not easily be converted to deal with different input. It can produce KAF, NAF and CoNNL output.
- **github address**: `https://bitbucket.org/Josu/corefgraph`

4.9 Semantic Role Labeling

- **Module**: vua-srl
- **Description of the module**: This module is based on the Semantic Role Labelling system as described in [Clercq *et al.*, 2012]. The original system takes one-file-per-sentence xml files as generated by the Alpino parser and generates comma separated feature vectors that provide the input to the machine learner (TiMBL [Daelemans and van den Bosch, 2005]). This machine learning algorithm tries to predict the role label between the predicate and the dependency that are represented in the feature vector.

  The training data used was annotated in the context of the SoNaR project [Oostdijk *et al.*, 2008]. For this module, we settled on using only the texts that pertain to newswire or magazines.[47] The trained model is provided with the module, the raw SoNaR data can be obtained through the TsT centrale.[48]

  For optimal integration with the NewsReader pipeline and to make the module robust against changes in any of the preceding processing steps (e.g. updates of the parser), we chose to reimplement their feature generator so that it takes a NAF file containing the term, constituents and dependencies layers as input. We have also added identifiers to the start of the feature vectors to make insert the predicted roles with the NAF file easier. The output is structures like the output of the IXA-srl module for English shown in Appendix A.9.
- **Input**: Lemmatized and POS tagged text and syntactic dependencies
- **Input representation**: NAF terms, constituents and dependencies layers

[47] A list of files used can be found in the module's GitHub archive

[48] `http://tst-centrale.org/nl/producten/corpora/sonar-corpus/6-85`

- **Output**: Semantic roles
- **Output representation**: NAF SRL layer
- **Required modules**: Tokenizer, POS tagger and Alpino parsing module (version described above)
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: SoNaR SRL training data
- **Dependencies**: Python, TiMBL 6.4.5
- **Flexible in- and output**: The module provides a NAF wrapper around a machine learning module on .csv values. The wrapper only uses KAF/NAF in- and output, but the overall architecture is flexible enough to support other wrappers.
- **github address**: `https://github.com/newsreader/vua-srl-nl`

4.10 FrameNet labelling

- **Module**: vua-framenet-classifier
- **Description of the module**: This module assigns FrameNet frames and elements to predicates and roles in the SRL layer. It reads the predicates in the SRL layer, matches these with the terms and their wordnet references to find the possible frames and elements as they are given in the PredicateMatrix. Next, it selects the highest ranked frame from all the possible frames and assign it to the predicate. The frame elements linked to that frame are then used to find a proper match with the PropBank roles assigned by the Semantic Role Labeling. If there is a direct match, it is added to the role as an external reference, if not it assigns the frame element to the PropBank role that has the strongest association. The latest version can assign both the predicates to terms as explained above and add FrameNet roles to the SRL layer. An example of the output is provided in Appendix A.10.
- **Input**: semantic roles and synsets
- **Input representation**: NAF terms, SRL layers
- **Output**: External references to FrameNet added to the SRL layer
- **Output representation**: NAF SRL layer
- **Required modules**: WSD and SRL
- **Level of operation**: sentence level

- **Language dependent**: yes
- **Resources**: predicate matrix and model for frame element = propbank role associations
- **Dependencies**: Java, Maven, NAF/KAF Saxparser
- **Flexible in- and output**: input stream NAF, output stream NAF (but the Predicate Matrix in itself can easily be used with other formats).
- **github address**: `https://github.com/cltl/OntoTagger`

4.11 Opinions

- **Module**: opinion-miner[49]
- **Description of the module**: This is a module that detects and extracts fine-grained opinions, where one single opinion contains three elements: the opinion expression (the subjective statement itself), the opinion target (what the opinion is about) and the opinion holder (who is expressing the opinion). This module has been developed as part of the OpeNER project,[50] where it has been trained on different domains (hotel reviews, political news...), and for different languages (Dutch, English, Spanish, Italian, French and German) using corpora manually annotated during the project. The extraction and tagging of opinions is divided into two steps. First, the detection of opinion entities (holder, target and expression) using Conditional Random Fields is carried out. This step uses the tokens, lemmas and POS on the context as features, as well as syntactic features and features extracted from lexicons. The second step is the opinion entity linking (expression−target and expression−holder) using binary Support Vector Machines. In this step all the single opinion entities detected are grouped into triples <expression,target,holder> according to the output of the SVM classifiers. In this case, besides the local context features, dependency features and features capturing the relative location of the opinion elements are included. The models have been trained with a rich set of features, but the opinion tagger can be used with a reduced subset of this features considering that the performance will be affected. The output representation for English is illustrated in Appendix A.19. The output of the Dutch module looks the same.
- **Input**: NAF text processed trough the pipeline, this module will use the information provided by all the rest
- **Input representation**: token, term, entity, dependency and constituency NAF layers

[49] The opinion miner module supports all languages in the project and is therefore used in several pipelines. The description largely corresponds to the description of the English module.

[50] `http://www.opener-project.eu/`

- **Output**: fine-grained opinion triples <expression, target, holder>
- **Output representation**: NAF opinion layer
- **Required modules**: tokenizer, lemmatizer, POS tagger, polarity tagger, named entity tagger, constituency parser and dependency parser
- **Level of operation**: document
- **Language dependent**: yes
- **Resources**: models trained during the OpeNER project
- **Dependencies**: CRF library,[51] SVM-Light library,[52] KafNafParser[53] and VUA_pylib library[54]
- **Flexible in- and output**: it takes NAF or KAF input text, and generates NAF or KAF output text enriched with the extracted opinions
- **github address**: `https://github.com/cltl/opinion_miner_deluxe`

4.12 Event coreference

- **Module**: vua-eventcoreference
- **Description of the module**: This module takes the predicates of the SRL layer as input and matches the predicates semantically. If the predicates are sufficiently similar, a coreference set is created with references to the predicates as coreferring expressions. If there is no match, predicates form a singleton set in the coreference layer. The output follows the same principles as the English output, for which an example can be found in Appendix A.12.
- **Input**: SRL
- **Input representation**: SRL predicates
- **Output**: Coreference sets for events
- **Output representation**: NAF coref layer
- **Required modules**: vua-srl
- **Level of operation**: sentence level

[51] `http://www.chokkan.org/software/crfsuite/`

[52] `http://www.chokkan.org/software/crfsuite/`

[53] `https://github.com/cltl/KafNafParserPy`

[54] `https://github.com/cltl/VUA_pylib`

- **Language dependent**: yes
- **Resources**: wordnet-lmf
- **Dependencies**: Java, Maven, KAF/NAF Saxparser, WordnetTools
- **Flexible in- and output**: the module only works on KAF and NAF
- **github address**: `https://github.com/cltl/EventCoreference`

4.13 Dutch Pipeline

This section presents the Dutch pipeline. More specifically, the dependencies among each task (module) are presented in Figure 3.

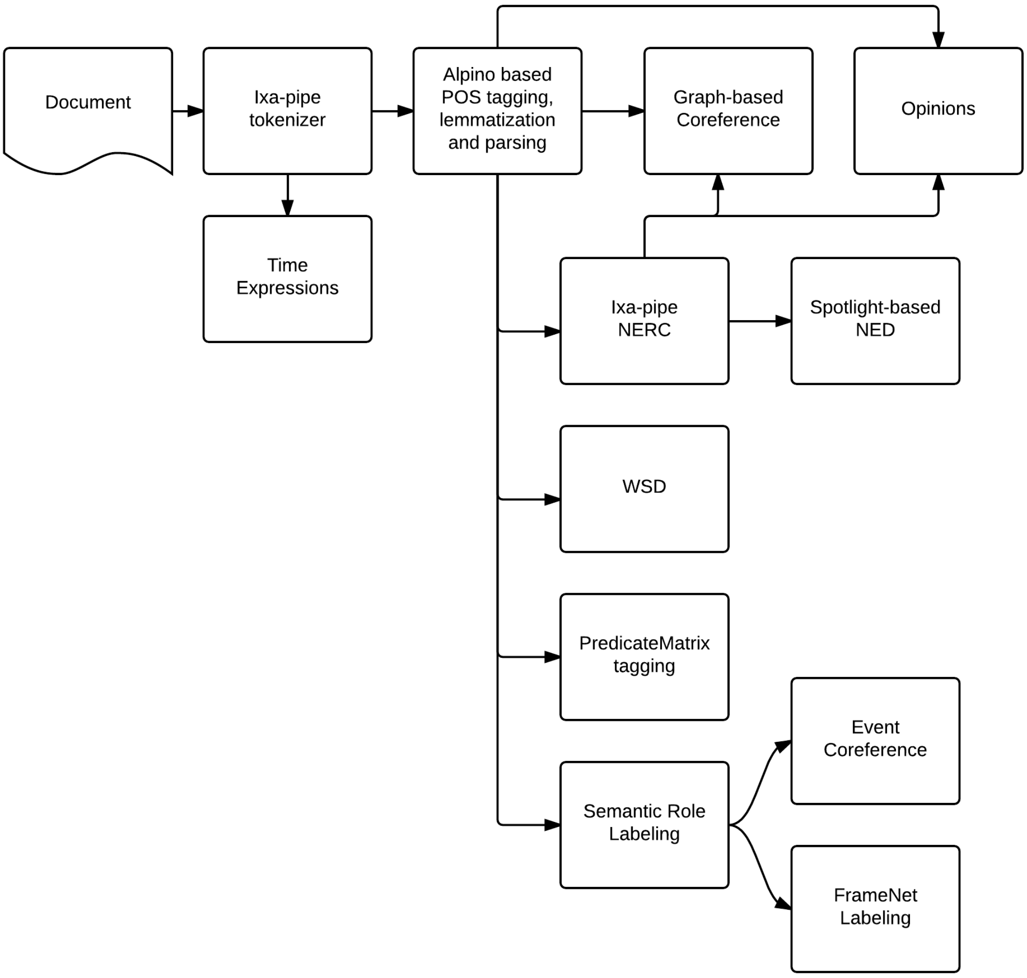

Figure 3: NewsReader Dutch pipeline. Dependencies among modules

5 Italian NLP Processing

The NLP processing suite for Italian is based on TextPro ([Pianta *et al.*, 2008]). The suite has been designed so that it can integrate and reuse existing state of the art NLP components developed by researchers at HLT-FBK group as much as possible. A wrapper program allows us to specify what kind of analyses are requested, and takes into account possible interdependencies between tasks. Internally, most module uses a column format (similar to CoNLL tabular format) of the input/output as interchange format. The modules for time expression detection and nomalization, event recognition, temporal relation extraction and factuality can also use NAF directly. For the other modules, the NAF format is used to get the input text and to provide the final output which is produced by a wrapper written in Java. In the rest of this section, we describe each module of the Italian pipeline. All TextPro modules are available for research purpose on a Github private repository. A form has to be fill at the address `http://textpro.fbk.eu/download-textpro.html` in order to follow it or to download the last version of TextPro.

5.1 Tokenizer

- **Module**: fbk-tokenpro
- **Description of the module**: TokenPro provides tokenization and sentence segmentation given an Italian raw text. It is a rule based splitter tool and it can be fully customizable from an XML configuration file (putting specific splitting word rules or handling UTF-8 symbols, such as the uncommon apostrophe, quote, dash,...). The sentence splitting is performed when a sentence-ending character (like *.*, *!*, or *?*) is found and it doesn't belong to an abbreviation. TokenPro is not distributed separately but is included in the TextPro distribution. The module produces output in column representations similar to CoNNL (Appendix A.20), this output is converted later to a NAF representation as also produced for English (Appendix A.1).
- **Input**: Raw text
- **Input representation**: NAF raw layer
- **Output**: Tokens and end of sentences
- **Output representation**: Token, tokenid, tokenstart, tokenend, tokentype and tokennorm columns
- **Required modules**: None
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: list of Italian abbreviations

- **Dependencies**: Java
- **Flexible in- and output**: It takes plain text or raw text in NAF. It produces tokenized and segmented text in NAF, running text and CoNLL formats.
- **github address**: `https://github.com/hltfbk/TextPro`

5.2 Morphological analysis

- **Module**: fbk-morphopro
- **Description of the module**: MorphoPro is a morphological analyzer (i.e. it finds all possible morphological analyses of a word in a text). It is based on word-form lists: for Italian it uses 1,878,285 analyses for 149,372 lemmas. An example of the column based output is provided in Appendix A.21.
- **Input**: Tokens
- **Input representation**: Tokennorm column
- **Output**: All possible morphological analyses for each token
- **Output representation**: full_morpho column
- **Required modules**: Tokenizer
- **Level of operation**: word level
- **Language dependent**: yes
- **Resources**: word-form list for Italian as finite state automata
- **Dependencies**: C++ library
- **Flexible in- and output**: The module uses tokennorm columns as input and produces columns with morphological information as output.
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/MorphoPro`

5.3 POS tagging

- **Module**: fbk-tagpro
- **Description of the module**: TagPro is a module for POS-tagging based on Support Vector Machine. It comes with two language models, Italian and English. The Italian model is trained on a corpus using a subset of the ELRA tagset ([Pianta and Zanoli, 2007]). It was the best system at EVALITA 2007 (98.04% F1). The output of this module is illustrated in Appendix A.22.

- **Input**: Token, morphological analysis
- **Input representation**: Token and full_morpho column
- **Output**: POS
- **Output representation**: POS column
- **Required modules**: Tokenizer, morphological analizer
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: language model
- **Dependencies**: YamCha,[55] TinySVM[56]
- **Flexible in- and output**: The module only works on the column format (as in- and output).
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/TagPro`

5.4 Lemmatization

- **Module**: fbk-lemmapro
- **Description of the module**: Given the morphological analyses (produced by the morphological analyzer) and the POS label (assigned by the POS tagger), the module selects compatible lemma(s). The module also selects the morphological analysis that is compatible (or analyses that are compatible) with the POS tag and the selected lemma(s). Appendix A.23 illustrates the output of the module. This output, together with the output of the POS tagger, is later converted to a NAF term layer like the English layer illustrated in Appendix A.2.
- **Input**: Token, morphological analysis, POS
- **Input representation**: Token, full_morpho and POS columns
- **Output**: Lemma and morphological analysis
- **Output representation**: Lemma and comp_morpho columns
- **Required modules**: Tokenizer, morphological analizer, POS-tagger

[55] `http://chasen.org/~taku/software/yamcha/`

[56] `http://chasen.org/~taku/software/TinySVM/`

- **Level of operation**: word level
- **Language dependent**: yes
- **Resources**: mapping among morphological features and POS tagset
- **Dependencies**: Java
- **Flexible in- and output**: The module only uses the column format as in- and output.
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/LemmaPro`

5.5 Named Entity Recognition and Classification

- **Module**: fbk-entitypro
- **Description of the module**: EntityPro performs named entity recognition based on machine learning. It exploits a rich set of linguistic features, as well as the occurrence in proper nouns gazetteers. The Italian model is trained on ICAB ([Magnini *et al.*, 2006]). The data contains entities of four types: Person (PER), Organization (ORG), Location (LOC) and Geo-Political entity (GPE). It was the best performing at EVALITA 2008 (82.1% F1). The module for system training is included in the distribution. Customization through white/black lists is possible. The output is illustrated in Appendix A.24, later converted to a NAF representation like the one illsutrated in A.6.
- **Input**: Token, lemma, POS
- **Input representation**: Token, tokennorm, POS and lemma columns
- **Output**: Named entities
- **Output representation**: Entity column
- **Required modules**: Tokenizer, morphological analyzer, lemmatizer, POS-tagger
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: language model, Proper nouns gazetteers
- **Dependencies**: YamCha,[57] TinySVM,[58]

[57] `http://chasen.org/~taku/software/yamcha/`

[58] `http://chasen.org/~taku/software/TinySVM/`

- **Flexible in- and output**: The module only uses column format as in- and output.
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/EntityPro`

5.6 Chunking

- **Module**: fbk-chunkpro
- **Description of the module**: This module groups words into flat constituents for a syntactic analysis. Chunking for Italian annotates with 2 categories: B-NP (for nominal predicate), B-VX (for verbal predicate). An example of the module's output is given in Appendix A.25.
- **Input**: Token, POS
- **Input representation**: Token and POS columns
- **Output**: Chunks
- **Output representation**: Chunk column
- **Required modules**: Tokenizer, POS-tagger
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: language model
- **Dependencies**: YamCha,[59] TinySVM[60]
- **Flexible in- and output**: The module only uses the column format as in- and output.
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/ChunkPro`

5.7 Dependency Parser

- **Module**: fbk-depparserpro

[59] `http://chasen.org/~taku/software/yamcha/`

[60] `http://chasen.org/~taku/software/TinySVM/`

- **Description of the module**: This module implements an Italian Dependency Parser. It is based on the Malt Parser ([Lavelli *et al.*, 2013]), a language-independent system for data-driven dependency parsing written in Java (open source). It was evaluated at Evalita 2011, using the Turin University Treebank for training (88.62% LAS, 92.85% UAS). The module outputs column representations of dependencies and labels as illustrated in Appendix A.26. This output is later converted to a NAF representation like the English representation provided in Appendix A.9.
- **Input**: Token, POS, lemma
- **Input representation**: Token, POS and lemma columns
- **Output**: Dependencies
- **Output representation**: Parserid, feats, headid and deprel columns
- **Required modules**: Tokenizer, POS-tagger, lemmatizer
- **Level of operation**: sentence level
- **Language dependent**: language model
- **Resources**: dependency parsing model trained on TUT (Turin University Treebank)[61]
- **Dependencies**: MaltParser `http://www.maltparser.org/`
- **Flexible in- and output**: The module only uses the column format as in- and output.
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/DepParserPro`

5.8 Time expression detection and normalization

- **Module**: fbk-timepro
- **Description of the module**: This module recognizes and normalizes temporal expressions in Italian. It processes the same way as TimePro for English. The core library timenorm ([Bethard, 2013]) has been adaptated for Italian. The Italian model is trained on EVENTI-EVALITA 2014 training data. At EVALITA 2014 it was the best performing on time expression recognition (82.7% F1) and class detection (80% F1) This module has been integrated into TextPro pipeline but it also accepts a NAF input file with tokenization, POS-tagging and chunking information and provides a NAF file as output. The output is structured as the English output illustrated in Appendix A.13.

[61] `http://www.di.unito.it/~tutreeb/`

- **Input**: Token layer, lemma, POS, entity, chunk
- **Input representation**: NAF terms, entities and chunks layers
- **Output**: timex3
- **Output representation**: NAF time expression layer
- **Required modules**: Tokenizer, POS-tagger, named entity recognizer, chunker
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language model, language dependent rules, Italian grammar for timenorm
- **Dependencies**: timenorm ([Bethard, 2013]), YamCha,[62] TinySVM[63]
- **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/TimePro`

5.9 Event recognizer

- **Module**: fbk-eventpro
- **Description of the module**: EventPro detects event extents and classifies them in one of the 7 classes defined in TimeML. The module is based on machine learning and it uses the Support Vector Machine (SVM) implementation provided by yamcha. The Italian model is trained on EVENTI-EVALITA 2014 data. It was evaluated at EVALITA 2014, and obtained the result of 86.7% F1 for the task of event recognition and 67.1% F1 for event classification. The module outputs a NAF event layer as also produced by modules for other languages (see Appendix A.15).
- **Input**: Token, lemma, POS, entity, chunk, time expression, morpho
- **Input representation**: NAF terms, entities, chunks and time expression layers
- **Output**: Event
- **Output representation**: NAF events layer
- **Required modules**: Tokenizer, POS-tagger, morphological analizer, chunker, named entity recognizer, temporal expressions recognizer and normalizer

[62] `http://chasen.org/~taku/software/yamcha/`

[63] `http://chasen.org/~taku/software/TinySVM/`

- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language model, language dependent rules
- **Dependencies**: Snowball Italian stemmer (`http://snowball.tartarus.org/algorithms/italian/stemmer.html`), MultiWordNet domains (`http://multiwordnet.fbk.eu/english/home.php`), derIvaTario lexicon (`http://derivatario.sns.it/`), YamCha (`http://chasen.org/~taku/software/yamcha/`), TinySVM[64]
- **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/EventPro`

5.10 Temporal relations extraction

- **Module**: fbk-temprel
- **Description of the module**: TempRel extracts temporal relations between events and time expressions as defined in TimeML. It processes the same way as TempRel for English. The Italian model is trained on EVENTI-EVALITA 2014 data. At EVALITA 2014 on the task of relation classification (identification of the relation type given the relations), it obtained 73.6% F1. The module produces the same output structure as the English module illustrated in Appendix A.14.
- **Input**: Token, lemma, POS, entity, chunk, morpho, dependency, event, time expression
- **Input representation**: NAF terms, entities, chunks, deps, time expression and events layers
- **Output**: tlinks
- **Output representation**: NAF temporal relation layer
- **Required modules**: Tokenizer, POS-tagger, morphological analizer, chunker, named entity recognizer, temporal expressions recognizer and normalizer, dependency parser, event recognizer
- **Level of operation**: document level
- **Language dependent**: yes

[64] `http://chasen.org/~taku/software/TinySVM/`

- **Resources**: language model, language dependent rules
- **Dependencies**: YamCha (`http://chasen.org/~taku/software/yamcha/`), TinySVM[65]
- **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/TempRelPro/`

5.11 Factuality detection

- **Module**: fbk-factpro
- **Description of the module**: FactPro carries out three steps: (1) detection of the polarity of an event (machine learning based), (2) identification of the certainty of an event (machine learning based) and (3) identification of the semantic time (rules based). The Italian models are trained on "Training data EVENTI task - Part 2"[66] annotated with factuality on top of TimeML annotation. The output of this module is illustrated in Appendix A.17. It serves as an example for future development for other languages.
- **Input**: Token layer, lemma, POS, entity, chunk, morpho, event
- **Input representation**: NAF terms, entities, chunks and events layers
- **Output**: Factuality values: polarity, certainty, semantic time
- **Output representation**: NAF factuality layer
- **Required modules**: Tokenizer, POS-tagger, morphological analizer, chunker, named entity recognizer, event recognizer
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language model, language dependent lexicons
- **Dependencies**: derIvaTario lexicon (`http://derivatario.sns.it/`), YamCha (`http://chasen.org/~taku/software/yamcha/`)
- **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).
- **github address**: `https://github.com/hltfbk/TextPro/tree/master/modules/FactPro`

[65] `http://chasen.org/~taku/software/TinySVM/`

[66] `https://sites.google.com/site/eventievalita2014/data-tools`

5.12 Italian Pipeline

This section presents the Italian pipeline. More specifically, the dependencies among each task (module) are presented in Figure 4.

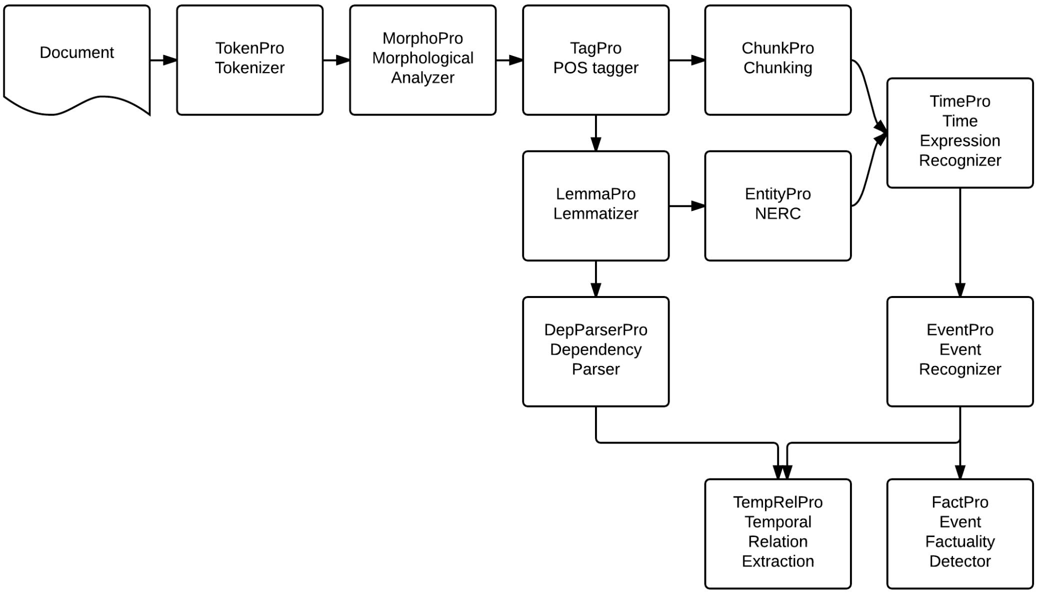
Document
TokenPro Tokenizer
MorphoPro Morphological Analyzer
TagPro POS tagger
ChunkPro Chunking
TimePro Time Expression Recognizer
LemmaPro Lemmatizer
EntityPro NERC
DepParserPro Dependency Parser
EventPro Event Recognizer
TempRelPro Temporal Relation Extraction
FactPro Event Factuality Detector

Figure 4: NewsReader Italian pipeline. Dependencies among modules

6 Spanish NLP Processing

The NLP processing for Spanish is similar to the English pipeline as they both share various modules to perform the processing. In this section, we follow the same structure as in Section 3 to present the modules. The descriptions of the modules are provided along with some technical information: input, output, required modules, level of operation, language dependency, resources, dependencies and github address.

6.1 Tokenizer

- **Module**: ixa-pipe-tok
- **Description of the module**: This module provides Sentence Segmentation and Tokenization for English and Spanish and other languages such as Dutch, German, English, French, Italian and Spanish. It implements a rule-based segmenter and tokenizer originally inspired by the Stanford English Penn Treebank tokenizer[67] but with several modifications and improvements. These include tokenization for other languages such as Italian, normalization according the Spanish Ancora Corpus [Taulé *et al.*, 2008], paragraph treatment, and more comprehensive gazetteers of non breaking prefixes. The tokenizer depends on a JFlex[68] specification file which compiles in seconds and performs at a very reasonable speed (around 250K word/second, and much quicker with Java multithreading). JFlex is a lexical analyzer generator. The module is part of the IXA pipes [Agerri *et al.*, 2014],[69] a modular set of Natural Language Processing tools (or pipes) which provide easy access to NLP technology for English and Spanish. The output for these languages has the same structure and is illustrated for this module in Appendix A.1.
- **Input**: Raw text
- **Input representation**: NAF raw layer
- **Output**: Tokens and sentences.
- **Output representation**: NAF text layer
- **Required modules**: None
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: None

[67] `http://www-nlp.stanford.edu/software/tokenizer.shtml`

[68] `http://jflex.de/`

[69] `http://ixa2.si.ehu.es/ixa-pipes/`

- **Dependencies**: Java, Maven, NAF Java library, JFlex
- **Flexible in- and output**: It takes plain text or raw text in NAF. It produces tokenized and segmented text in NAF, running text and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

6.2 POS tagging

- **Module**: ixa-pipe-pos
- **Description of the module**: This module provides POS tagging and lemmatization for English and Spanish. The module is part of the IXA pipes. We have obtained the best results so far with *Maximum Entropy* models and the same featureset as in [Collins, 2002]. The models ave been trained and evaluated for Spanish using the Ancora corpus; it was randomly divided in 90% for training and 10% for testing. This corresponds to 440K words used for training and 70K words for testing. We obtain a performance of 98.88% (the corpus partitions are available for reproducibility). [Giménez and Màrquez, 2004] report 98.86%, although they train and test on a different subset of the Ancora corpus.

  For Spanish, lemmatization is currently performed via 2 different dictionary lookup methods: a) *Simple Lemmatizer*: It is based on HashMap lookups on a plain text dictionary. Currently we use dictionaries from the LanguageTool project[70] under their distribution licenses; b) Morfologik-stemming:[71] The Morfologik library provides routines to produce binary dictionaries, from dictionaries such as the one used by the Simple Lemmatizer above, as finite state automata. This method is convenient whenever lookups on very large dictionaries are required because it reduces the memory footprint to 10% of the memory required for the equivalent plain text dictionary. By default, the module accepts tokenized text in NAF format as standard input and outputs NAF (see Appendix A.2).
- **Input**: Tokens
- **Input representation**: NAF text layer
- **Output**: Lemmas and POS-tags
- **Output representation**: NAF terms layer
- **Required modules**: Tokenizer module
- **Level of operation**: sentence level

[70] `http://languagetool.org/`

[71] `https://github.com/morfologik/morfologik-stemming`

- **Language dependent**: yes
- **Resources**: POS model; Lemmatizer dictionaries: plain text dictionary and morfologik-stemming dictionary.
- **Dependencies**: Java, Maven, NAF Java library, JWNL API, Apache OpenNLP.
- **Flexible in- and output**: It accepts tokenized text in NAF. It outputs NAF or CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

6.3 Constituency Parser

- **Module**: ixa-pipe-parse
- **Description of the module**: This module provides statistical constituent parsing for English and Spanish. The module is part of the IXA pipes. Maximum Entropy models are trained to build shift reduce bottom up parsers [Ratnaparkhi, 1999] as provided by the Apache OpenNLP Machine Learning API. Parsing models for Spanish have been trained using the Ancora corpus [Taulé *et al.*, 2008]. Furthermore, ixa-pipe-parse provides two methods of headword finders: one based on Collins' head rules as defined in his PhD thesis [Collins, 1999], and another one based on Stanford's parser Semantic Head Rules.[72] The latter are a modification of Collins' head rules according to lexical and semantic criteria. We obtain a F1 88.40%. As far as we know, and although previous approaches exist [Cowan and Collins, 2005], ixa-pipe-parse provides the first publicly available statistical parser for Spanish. The module accepts lemmatized and POS tagged text in NAF format as standard input and outputs NAF as illustrated in Appendix A.8.
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Constituents; Syntactic tree of sentences.
- **Output representation**: NAF constituency layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: Parsing model

[72] `http://www-nlp.stanford.edu/software/lex-parser.shtml`

- **Dependencies**: Java, Maven, NAF Java library, Apache OpenNLP
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. In addition to NAF output, ixa-pipe-parse can also output the parse trees into Penn Treebank bracketing style.
- **github address**: `https://github.com/newsreader/ixa-pipes`

6.4 Dependency Parser

- **Module**: ixa-pipe-srl
- **Description of the module**: This module is based on the MATE-tools [Björkelund *et al.*, 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. As the input of the module is a NAF file that includes lemmatization and pos-tagging, the module only implements the dependency parser [Bohnet, 2010]. The module is ready to work with Spanish and English. An example of the NAF output of this module for English is given in Appendix A.9.
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Dependencies
- **Output representation**: NAF deps layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: mate-tools package, dependency parsing model[73]
- **Dependencies**: Java, Maven, NAF Java library, mate-tools
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. The modules allows to output dependencies trees in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipe-srl`

[73] The module use additional resources to perform the semantic role labeling.

6.5 Time Expressions

- **Module**: ixa-heideltime
- **Description of the module**: This module identifies time expressions in text and normalizes them. The core component of the module is HeidelTime [Strötgen and Gertz, 2013], a temporal tagger supporting English, German, Dutch, Vietnamese, Arabic, Spanish, Italian, French, Chinese and Russian.[74] HeidelTime identifies temporal expressions based on language specific patterns. Identified temporal expressions are normalized and represented according to TIMEX annotations [Sundheim, 1996]. This module is currently under development. The output will be modeled after the output for the other languages as illustrated in Appendix A.13.
- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Normalized time expressions in timex
- **Output representation**: NAF time expression layer
- **Required modules**: Tokenizer module
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: language specific rules
- **Dependencies**: Python
- **Flexible in- and output**: HeidelTime itself runs on raw text and produces inline annotations. It can also be integrated in UIMA pipelines. Our implementation takes NAF as input and produces this as output as well.
- **github address**: `https://github.com/ixa-ehu`

6.6 Named Entity Recognition and Classification

- **Module**: ixa-pipe-nerc
- **Description of the module**: This module is a multilingual Named Entity Recognition and Classification tagger. ixa-pipe-nerc is part of IXA pipes. The named entity types are based on: a) the CONLL 2002[75] and 2003[76] tasks which were focused on

[74] HeidelTime source code is available here: `https://code.google.com/p/heideltime/`

[75] `http://www.clips.ua.ac.be/conll2002/ner/`

[76] `http://www.clips.ua.ac.be/conll2003/ner/`

language-independent supervised named entity recognition for four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. We currently provide two very fast language independent models using a rather simple baseline feature set. It is based on the features presented by [Zhang and Johnson, 2003] with several differences: We do not use POS tags, chunking or gazetteers in our baseline models but we do use bigrams as a feature.

For Spanish we currently obtain best results training Maximum Entropy models on the CoNLL 2002 dataset. Our best model obtains 80.16 F1 vs 81.39 F1 of [Carreras *et al.*, 2002], the best result so far on this dataset. Their result uses external knowledge and without it, they system obtains 79.28 F1. The module can format its output in CoNLL style tabulated BIO format as specified in the CoNLL 2003 shared evaluation task in addition to the NAF representation as shown in Appendix A.6.

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Named entities
- **Output representation**: NAF entities layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: CoNLL 2003 models, properties file
- **Dependencies**: Java, Maven, NAF Java library, Apache OpenNLP
- **Flexible in- and output**: It accepts lemmatized and POS tagged text in NAF format. The modules allows to output dependencies trees in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipes`

6.7 Word Sense Disambiguation

- **Module**: wsd-ukb
- **Description of the module**: UKB is a collection of programs for performing graph-based Word Sense Disambiguation. UKB applies the so-called Personalized PageRank on a Lexical Knowledge Base (LKB) to rank the vertices of the LKB and thus perform disambiguation. UKB has been developed by the IXA group. The Spanish WSD module was evaluated on SemEval-2007 Task 09 dataset [Màrquez *et al.*, 2007].

The dataset contains examples of the 150 most frequent nouns in the CESS-ECE corpus, manually annotated with Spanish WordNet synsets. We ran the experiment over the test part of the dataset (792 instances) and obtained F1 79.3. The module accepts lemmatized and POS tagged text in NAF format as standard input and outputs NAF (see Appendix A.4.

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Synsets
- **Output representation**: NAF terms layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: Spanish WordNet
- **Dependencies**: C++, boost libraries
- **Flexible in- and output**: The module uses KAF/NAF as input and output.
- **github address**: `https://github.com/ixa-ehu/ukb`

6.8 Named Entity Disambiguation

- **Module**: ixa-pipe-ned
- **Description of the module**: This module performs the Named Entity Disambiguation task based on DBpedia Spotlight. Assuming that a DBpedia Spotlight Rest server for a given language is locally running, the module will take NAF as input (containing elements) and perform Named Entity Disambiguation. The module offers the "disambiguate" and "candidates" service endpoints. The former takes the spotted text input and it returns the identifier for each entity. The later is similar to disambiguate, but returns a ranked list of candidates. For Spanish, given a disambiguate entity, it is also possible to return the corresponding English dbpedia-entry. This feature allows the interoperability of Named Entities. The Spanish ixa-pipe-ned module has been evaluated on the TAC 2012 Spanish dataset. We obtained a performance of 78.15 in precision and 55.80 in recall. The module accepts text with named entities in NAF format as standard input, it disambiguates them and outputs them in NAF. An example of this output for English is given in Appendix A.7.
- **Input**: Named entities and sentences

- **Input representation**: NAF entities layer
- **Output**: Disambiguated named entities
- **Output representation**: NAF entities layer
- **Required modules**: NERC module
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: DBpedia spotlight server
- **Dependencies**: Java, Maven, MapDB, NAF Java library, DBpedia spotlight
- **Flexible in- and output**: The module only uses NAF/KAF as input and output.
- **github address**: `https://github.com/newsreader/ned-spotlight`

6.9 Coreference Resolution

- **Module**: corefgraph
- **Description of the module**: The module of coreference resolution included in the IXA pipeline is loosely based on the Stanford Multi Sieve Pass system [Lee *et al.*, 2013]. The system consists of a number of rule-based sieves. Each sieve pass is applied in a deterministic manner, reusing the information generated by the previous sieve and the mention processing. The order in which the sieves are applied favours a highest precision approach and aims at improving the recall with the subsequent application of each of the sieve passes. This is illustrated by the evaluation results of the CoNLL 2011 Coreference Evaluation task [Lee *et al.*, 2013; Lee *et al.*, 2011], in which the Stanford's system obtained the best results. The results show a pattern which has also been shown in other results reported with other evaluation sets [Raghunathan *et al.*, 2010], namely, the fact that a large part of the performance of the multi pass sieve system is based on a set of significant sieves. Thus, this module focuses for the time being, on a subset of sieves only, namely, Speaker Match, Exact Match, Precise Constructs, Strict Head Match and Pronoun Match [Lee *et al.*, 2013]. For Spanish, the module has been evaluated on the publicly available datasets distributed by the SemEval 2010 task on Multilingual Coreference resolution and we obtained different F1 metrics: 70.67 CEAFm F1, 43.58 MUC F1, 75.94 B3 F1 and 61.42 BLANC F1. Appendix A.12.
- **Input**: lemma, POS, named-entities, constituents
- **Input representation**: NAF entities, constituency layers

- **Output**: coreferences
- **Output representation**: NAF coreferences layer
- **Required modules**: Tokenizer, POS-tagger and NERC modules
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: none
- **Dependencies**: pyKAF, pycorpus, networkx, pyYALM
- **Flexible in- and output**: It accepts lemmatized and POS tagged text, entities and constituents in NAF format. The modules allows to output coreference clusters in NAF and CoNLL formats.
- **github address**: `https://bitbucket.org/Josu/corefgraph`

6.10 Semantic Role Labeling

- **Module**: ixa-pipe-srl
- **Description of the module**: This module is based on the MATE-tools [Björkelund *et al.*, 2010], a pipeline of linguistic processors that performs lemmatization, part-of-speech tagging, dependency parsing, and semantic role labeling of a sentence. They report on the CoNLL 2009 Shared Task a labelled semantic F1 of 85.63 for English and 79.91 for Spanish. As the input of the module is a NAF file that includes lemmatization, POS-tagging and dependency parsing, the module only implements the semantic role labeler [Björkelund *et al.*, 2009]. The module is ready to work with Spanish and English. By default, the module accepts parsed text in NAF format as standard input and outputs the enriched text in NAF. Original the output annotations are based on PropBank/NomBank or AnCora, but the module makes use of the *Predicate Matrix* as an external resource to enrich the semantic information of the annotation, including both for predicates and arguments their correspondances in FrameNet, VerbNet and, in case of spanish or nominal predicates, their sources in PropBank. The structure of the modules output is illustrated in Appendix A.9.
- **Input**: Lemmatized and POS tagged text and syntactic dependencies
- **Input representation**: NAF terms, deps layers
- **Output**: Semantic roles
- **Output representation**: NAF srl layer

- **Required modules**: Tokenizer, POS tagger and Dependency parsing modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: mate-tools package, PredicateMatrix
- **Dependencies**: Java, Maven, NAF Java library, mate-tools
- **Flexible in- and output**: It accepts lemmatized and POS tagged text and syntactic dependencies in NAF format. The modules allows to output semantic roles in NAF and CoNLL formats.
- **github address**: `https://github.com/newsreader/ixa-pipe-srl`

6.11 Event coreference

- **Module**: vua-eventcoreference
- **Description of the module**: This module takes the predicates of the SRL layer as input and matches the predicates semantically. If the predicates are sufficiently similar, a coreference set is created with references to the predicates as coreferring expressions. If there is no match, predicates form a singleton set in the coreference layer. Appendix A.12 provides an example of the output of the event coreference module for English. The output for Spanish follows the same structure.
- **Input**: SRL
- **Input representation**: NAF srl layer
- **Output**: Coreference sets for events
- **Output representation**: NAF coref layer
- **Required modules**: ixa-pipe-srl
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: wordnet-lmf
- **Dependencies**: Java, Maven, KAF/NAF Saxparser, WordnetTools
- **Flexible in- and output**: no
- **github address**: `https://github.com/cltl/EventCoreference`

6.12 Spanish Pipeline

This section presents the Spanish pipeline. More specifically, the dependencies among each task (module) are presented in Figure 5.

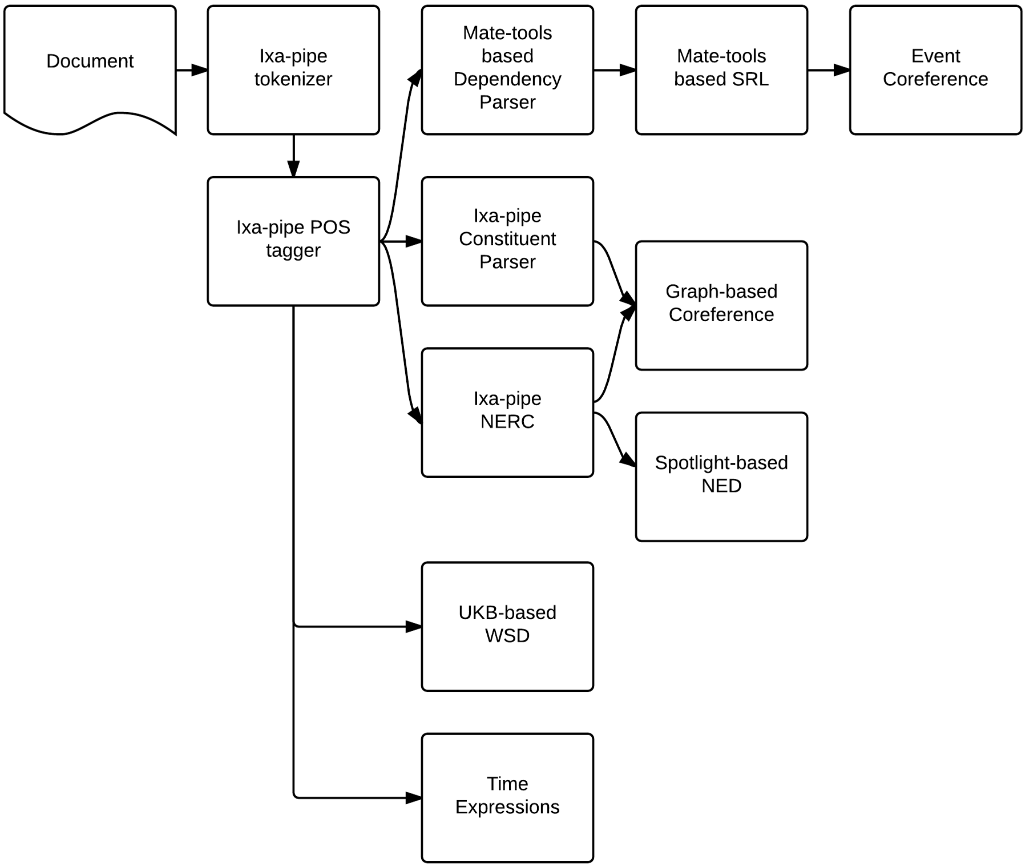

Figure 5: NewsReader Spanish pipeline. Dependencies among modules

7 Text Classification

In this second cycle of the project, we have decided to automatically set the topic of each document. This task is part of WP4 and it is applicable in all the languages of the project. We have implemented one module which works for the four languages of the project and it runs per set of documents. Given a set of documents, the module can be seen either as a pre-processing or post-processing step.

Document descriptors are useful in NewsReader to perform event coreference. The topic determines the domain of the document and this information, among other features, is used for event coreference resolution [Cybulska and Vossen, 2013].

The module is based on the Multilingual Eurovoc thesaurus descriptors and it makes use of the JRC Eurovoc Indexer JEX [Steinberger *et al.*, 2012]. As the rest of the available modules for text processing, this module reads from the standard input a NAF file and it writes the new version with the topic information in NAF. The topic information is represented in the <topic> layer in the NAF document and it corresponds to the whole document.

The technical information regarding this module can be summarized as follows:

- **Input**: text
- **Input representation**: NAF terms layer
- **Output**: topic
- **Output representation**: NAF topic layer
- **Required modules**: None
- **Level of operation**: document level
- **Language dependent**: yes
- **Resources**: None
- **Dependencies**: Java, Maven, NAF Java library, JEX
- **Flexible in- and output**: no
- **github address**: `https://github.com/newsreader/`

We are currently studing an alternative version of the module that processes one document per time. If the processing time is reasonable, we will integrate it into the pipelines.

8 Scaling of Text Processing

In the first cycle of event detection, we performed some initial experiments with the goal of analyzing the scaling capabilities of our English NLP processing pipeline. As a result of the analysis made in WP2, we chose the distributed framework called STORM to integrate the pipeline into it. For that we use virtual machines which are described in [Soroa and Fernández, 2014]. The results of these experiments are described in the deliverable 4.2.1 “Event detection, version 1”.

In the second cycle of the project, we have continued collecting and processing different data. So far, we have stored the following data: LN car company news (EN) : 63K articles (6M); TechCrunch (EN) : 43K articles; WikiNews (EN, ES) : 19K English and 7K Spanish; ECB+ (EN) : 984 articles; FIFA World Cup (Hackathon Londres): LexisNexis, BBC, The Guardian (200K news).

The analyzed documents from the FIFA 2014 World Cup domain were the basis from which to create the RDF triplets as used in the First NewsReader Hackaton held in London on the 10th June, 2014. The documents were processed within a 15 day time-frame using 30 copies of the virtual machines. FBK has set up and run the NLP processing pipeline on the FBK cluster. Once again, we tested the pipeline as regards the scaling of text processing. Due to time constraints, the 200K news articles were processed using the processing pipeline set up at the end of Y1.

9 Conclusions

This deliverable describes the second version of the **Event Detection** framework developed in NewsReader to process large and continuous streams of English, Dutch, Spanish and Italian news articles.

During the second cycle of the NewsReader project (Event Detection, version 2) we focused on providing generic pipelines for English, Dutch, Spanish and Italian. We also improved the modules for English event detection. For instance, several modules of the English NLP pipeline have been improved, including those for event detection, named entity recognition, coreference resolution and semantic role labeling. Some of these modules have been also adpated to the financial domain. We have been also deploying new generic pipelines and modules for Dutch, Spanish and Italian. Table 15 presents the current state of all the available modules.

Name	2nd year	Version	1st year	Language(s)	Third-party software	Resources
ixa-pipe-tok	No update	1.5.0	✓	All		
ixa-pipe-pos	Update	1.3.0	✓	EN, ES	API Apache OpenNLP project	WSJ treebank, Ancora

ixa-pipe-parse	Update	1.0.2	✓	EN, ES	API Apache OpenNLP project	Penn Treebank, Ancora
ixa-pipe-srl	Update	1.1.0	✓	EN, ES	mate-tools	PropBank, NomBank, Ancora, Predicate Matrix
fbk-timepro	Update	1.2	✓	EN, IT	timenorm, YamCha	TempEval3 data, EVENTI-EVALITA 2014 data
ixa-pipe-nerc	Update	1.3.3	✓	All	API Apache OpenNLP project	CoNLL 2002, 2004, OntoNotes, Evalita07, Evalita09
wsd-ukb	Update	2.0.22	✓	EN, ES		WordNet
ixa-pipe-ned	Update	1.1.1	✓	All	DBpedia spotlight	DBpedia spotlight server, Wikipedia
corefgraph	Update	0.8	✓	EN, ES, NL		
vua-eventcoreference	Update	2.1	✓	EN, ES, NL		WordNet-lmf
fbk-temprel	New	2.0.0		EN, IT	Explicit Discourse Connectives Tagger, YamCha	TempEval3 data, EVENTI-EVALITA 2014 data
fbk-causalrel	New	1.0.0		EN	Explicit Discourse Connectives Tagger, YamCha	TimeBank
vua-factuality	No update	1.0	✓	EN	mallet	FactBank
opinion-miner	No update	2.0	✓	EN, NL		OpeNER models

vua-alpino	Update	1.0		NL	Alpino parser	
vua-heideltime	New	1.0		NL	HeidelTime, TreeTager	
vua-wsd	Update	1.2		NL		DutchSemCor
vua-ontotagging	Update	1.0		NL		Predicate Matrix
vua-srl	New	1.0		NL	TimBL	SoNaR SRL data
vua-framenet-classifier	New	1.0		NL		Predicate Matrix
ixa-heideltime	✓	1.0.0		ES	HeidelTime	
fbk-tokenpro	no update	2.1	✓	EN, IT		
fbk-morphopro	no update	1.3.2-1	✓	EN, IT		word form list
fbk-tagpro	no update	1.5.0	✓	EN, IT	Yamcha, TinySVM	ELRA tagset
fbk-lemmapro	no update	2.0	✓	EN, IT		
fbk-entitypro	no update	1.4.3	✓	EN, IT	Yamcha, TinySVM	ICAB corpus
fbk-chunkpro	no update	2.0	✓	EN, IT	Yamcha, TinySVM	
fbk-depparsepro	no update	1.0	✓	IT	MaltParser	Turin University Treebank
fbk-timepro	update	1.2	✓	EN, IT	timenorm, YamCha, TinySVM	TempEval3 data, EVENTI-Evalita data
fbk-temprel	new	2.0.0		EN, IT	Explicit Discourse Connectives Tagger, fbk-morphopro, YamCha, TinySVM	TempEval3 data, EVENTI-Evalita data

fbk-eventpro	new	1.0		IT	Snowball Italian stemmer, MultiWordNet domains, derIvaTario lexicon, Yamcha, TinySVM	EVENTI-Evalita 2014 data
fbk-factpro	new	1.0		IT	derIvaTario lexicon, Yamcha, TinySVM	Fact-Ita Bank
text-classification	New	1.0.0		All	JEX	
Predicate Matrix	Update	1.1.0	✓	EN, ES		PropBank, NomBank, Ancora, VerbNet, FrameNet, SemLink, WordNet

Table 15: EN, IT, NL and SP modules.

In order to achieve cross-lingual semantic interoperability, entity and event mentions have been projected to language independent knowledge representations. Thus, named entities are linked as much as possible to external sources such as DBpedia entity identifiers while event mentions are aligned to abstract event representations thanks to the Predicate Matrix [López de Lacalle *et al.*, 2014]. We are also developing new techniques and resources to achieve interoperable semantic interpretation for English, Dutch, Spanish and Italian thanks to DBpedia cross-lingual links and multilingual semantic projections through local wordnets of the Predicate Matrix.

The benchmarks for the four languages will be available in the last trimester of the second year of the project. We will use this datasets to evaluate our four pipelines. The evaluation results will be presented in deliverable D3.3.2 “Annotated data.”

Five datasets covering different topics have been processed by the NLP pipeline. Together these sets consists of approximately 326K articles in English and 7K articles in Spanish.

We will continue researching on the improvement of the event detection systems by analysing the interacting tasks such as NERC, NED and coreference. We are also planning to define an experiment to start exploiting the KnowledgeStore provided in WP06 to try to improve the event detection system.

A Output representation

This section presents the input and output of each module presented in sections 3, 4, 5 and 6 in the NAF representation format. Each module produces linguistic information at one layer defined in NAF. The complete NAF representation is available at `http://wordpress.let.vupr.nl/naf/` and `https://github.com/newsreader/NAF`.

A.1 ixa-pipe-tok

The ixa-pipe-tok provides sentence segmentation and tokenization. For that, it takes the information contained in the <raw> element and it annotates word forms within the <text> element. Each form is enclosed by a <wf> element and it has the following attributes: word id, sentence id, paragraph id, the offset and length of the word form. Example:

```
<wf id="w1" length="9" offset="21" para="1" sent="1">President</wf>
```

A.2 ixa-pipe-pos

The ixa-pipe-pos provides POS tagging and lemmatization. The ixa-pipe-pos reads <wf> elements and it annotates terms within the <terms> element. Each term is enclosed by a <term> element and it has the following attributes: term id, type, lemma, pos, morphofeat and the span sub-element which is used to identify the tokens that the term spans. Example:

```
<term id="t1" lemma="President" morphofeat="NNP" pos="R" type="close">
  <span>
    <!--President-->
    <target id="w1"/>
  </span>
</term>
```

A.3 wsd-vua

The wsd-vua module provides word sense disambiguation. It associate terms to WordNet resource. It consists of several <externalRef> elements, one per association. The module returns the <externalRef> element with resource, reference and confidence attributes. Example:

```
<term id="t9" lemma="man" morphofeat="NN" pos="N" type="open">
  <span>
    <!--man-->
    <target id="w9"/>
  </span>
```

```
      <externalReferences>
        <externalRef confidence="0.020081263" reference="r_n-23112" resource="Cornetto"/↩
            >
        <externalRef confidence="0.048200976" reference="r_n-23113" resource="Cornetto"/↩
            >
        <externalRef confidence="0.16325809" reference="r_n-23111" resource="Cornetto"/>
      </externalReferences>
    </term>
```

A.4 wsd-ukb

The wsd-ukb module provides word sense disambiguation. It associate terms to WordNet resource. It consists of several <externalRef> elements, one per association. The module returns the <externalRef> element with resource, reference and confidence attributes. Example:

```
    <term id="t5" type="open" lemma="presidente" pos="N" morphofeat="NCMS000">
      <span>
        <target id="w5" />
      </span>
      <externalReferences>
        <externalRef resource="wn30sp.bin64" reference="eng-30-10467179-n" confidence="↩
            0.266266" />
        <externalRef resource="wn30sp.bin64" reference="eng-30-10468559-n" confidence="↩
            0.253603" />
        <externalRef resource="wn30sp.bin64" reference="eng-30-10467395-n" confidence="↩
            0.245211" />
        <externalRef resource="wn30sp.bin64" reference="eng-30-10468962-n" confidence="↩
            0.23492" />
      </externalReferences>
    </term>
```

A.5 Predicate Matrix (term layer)

The predicate matrix adds references to FrameNet frames and PropBank roles to terms (based on the mappings between WordNet synsets and these resources). It currently provides a complete overview of the frame/predicate with its associated roles. It is illustrated in the image below.

```
<term id="t_1" lemma="treden" morphofeat="WW(pv,verl,ev)" pos="verb" type="open">
      <span>
        <!--trad-->
        <target id="w2"/>
      </span>
      <externalReferences>
        <externalRef confidence="0.546046" reference="d_v-5898-v" resource="cdb2.0-nld-↩
            all.infv.0.0.no-allwords">
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Board_vehicle" resource="fn"/>
            <externalRef reference="fn-role:Source" resource="fn-role"/>
            <externalRef reference="fn-role:Time" resource="fn-role"/>
            <externalRef reference="fn-role:Vehicle" resource="fn-role"/>
            <externalRef reference="fn-role:Path" resource="fn-role"/>
          </externalRef>
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Getting" resource="fn"/>
            <externalRef reference="pb:get.03" resource="pb"/>
            <externalRef reference="fn-pb-role:Theme#0" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Recipient#2" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Source#1" resource="fn-pb-role"/>
            <externalRef reference="fn-role:Source" resource="fn-role"/>
            <externalRef reference="pb:get.04" resource="pb"/>
            <externalRef reference="pb:get.05" resource="pb"/>
            <externalRef reference="pb:get.15" resource="pb"/>
          </externalRef>
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Becoming" resource="fn"/>
            <externalRef reference="fn-role:Entity" resource="fn-role"/>
            <externalRef reference="fn-role:Final_category" resource="fn-role"/>
            <externalRef reference="pb:get.03" resource="pb"/>
            <externalRef reference="fn-pb-role:Cause#0" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Place#2" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Entity#1" resource="fn-pb-role"/>
            <externalRef reference="pb:get.04" resource="pb"/>
            <externalRef reference="pb:get.05" resource="pb"/>
            <externalRef reference="pb:get.15" resource="pb"/>
          </externalRef>
        </externalRef>
        <externalRef confidence="0.312131" reference="d_v-4023-v" resource="cdb2.0-nld-↩
            all.infv.0.0.no-allwords">
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Self_motion" resource="fn"/>
            <externalRef reference="pb:walk.01" resource="pb"/>
            <externalRef reference="fn-role:Source" resource="fn-role"/>
            <externalRef reference="fn-pb-role:Area#1" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Self_mover#0" resource="fn-pb-role"/>
          </externalRef>
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Cotheme" resource="fn"/>
            <externalRef reference="pb:walk.01" resource="pb"/>
            <externalRef reference="fn-role:Theme" resource="fn-role"/>
            <externalRef reference="fn-pb-role:Area#1" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Theme#0" resource="fn-pb-role"/>
          </externalRef>
        </externalRef>
        <externalRef confidence="0.141823" reference="d_v-7257-v" resource="cdb2.0-nld-↩
            all.infv.0.0.no-allwords">
          <externalRef resource="predicate-matrix1.1">
            <externalRef reference="fn:Cause_harm" resource="fn"/>
            <externalRef reference="fn:kick.v" resource="fn"/>
            <externalRef reference="pb:kick.01" resource="pb"/>
            <externalRef reference="fn-pb-role:Agent#0" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Cause#0" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Instrument#2" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Body_part#1" resource="fn-pb-role"/>
            <externalRef reference="fn-pb-role:Victim#1" resource="fn-pb-role"/>
            <externalRef reference="fn-role:Body_part" resource="fn-role"/>
            <externalRef reference="fn-role:Instrument" resource="fn-role"/>
            <externalRef reference="fn-role:Place" resource="fn-role"/>
          </externalRef>
        </externalRef>
      </externalReferences>
    </term>
```

A.6 ixa-pipe-nerc

The ixa-pipe-nerc module provides named entities. For that, it takes the <wf> elements and it uses the <entity> to represent a named entity in the document.[77] The <entity> element has the id and type attributes. It also has the <references> sub-element which contains one or more <span> element, each one spaning terms. Example:

```
<entities>
  <entity id="e1" type="PERSON">
    <references>
      <span>
        <!--Mariano Rajoy-->
        <target id="t1"/>
        <target id="t2"/>
      </span>
    </references>
  </entity>
```

A.7 ixa-pipe-ned

The ixa-pipe-ned module provides external references to named entities. It reads <wf>, <term> and <entity> elements and it links named entities to an external resources using the <externalRef> element. The <externalRef> has the resource, reference and confidence attributes. Example:

```
<entities>
  <entity id="e1" type="PERSON">
  ...
    <externalReferences>
      <externalRef reference="http://dbpedia.org/resource/Mariano_Rajoy" resource="↩
          spotlight_v1" confidence="0.7"/>
    </externalReferences>
  </entity>
```

For Spanish, if the corresponding English dbpedia-entry is provided, the <externalRef> does not have the confidence attribute.

```
  <entity id="e1" type="PERSON">
    <references>
      <!--Mariano Rajoy-->
      <span>
        <target id="t1" />
        <target id="t2" />
      </span>
    </references>
    <externalReferences>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/↩
          Mariano_Rajoy" confidence="0.9999986" />
      <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/resource↩
          /Mariano_Rajoy" />
    </externalReferences>
```

[77] The <term> elements are used to write the <span> elements in the output NAF document.

```
  </entity>
```

A.8 ixa-pipe-parse

The ixa-pipe-parse provides constituent parsing. It reads <wf> elements and it annotates constituents within the <constituency> element, and each sentence (parse tree) is represented by a <tree> element.[78] Inside each <tree>, there are three types of elements: a) <nt> elements representing non-terminal nodes; b) <t> elements representing terminal nodes; and c) <edge> elements representing in-tree edges. The <nt> element has the id and label attributes. The <t> element has the id attribute and the <span> element pointing to the term layer. Finally, the <edge> element has the id, from, to and head attributes. Example:

```
<constituency>
  <tree>
    <t id="ter1">
      <!--Mariano-->
      <span>
        <target id="t1"/>
      </span>
    </t>
    <t id="ter2">
      <!--Rajoy-->
      <span>
        <target id="t2"/>
      </span>
    </t>
    <t id="ter3">
      <!--is-->
      <span>
        <target id="t3"/>
      </span>
    </t>
    <t id="ter4">
      <!--the-->
      <span>
        <target id="t4"/>
      </span>
    </t>
    <t id="ter5">
      <!--President-->
      <span>
        <target id="t5"/>
      </span>
    </t>
    <t id="ter6">
      <!--of-->
      <span>
        <target id="t6"/>
      </span>
    </t>
    <t id="ter7">
      <!--Spain-->
      <span>
```

[78] The <term> elements are used to write the <span> elements in the output NAF document.

```
        <target id="t7"/>
      </span>
    </t>
    <t id="ter8">
      <!--.-->
      <span>
        <target id="t8"/>
      </span>
    </t>
    <nt id="nter1" label="TOP"/>
    <nt id="nter2" label="S"/>
    <nt id="nter3" label="NP"/>
    <nt id="nter4" label="NNP"/>
    <nt id="nter5" label="NNP"/>
    <nt id="nter6" label="VP"/>
    <nt id="nter7" label="VBZ"/>
    <nt id="nter8" label="NP"/>
    <nt id="nter9" label="NP"/>
    <nt id="nter10" label="DT"/>
    <nt id="nter11" label="NNP"/>
    <nt id="nter12" label="PP"/>
    <nt id="nter13" label="IN"/>
    <nt id="nter14" label="NP"/>
    <nt id="nter15" label="NNP"/>
    <nt id="nter16" label="."/>
    <edge from="nter2" head="yes" id="tre2" to="nter1"/>
    <edge from="nter3" id="tre3" to="nter2"/>
    <edge from="nter4" id="tre4" to="nter3"/>
    <edge from="ter1" id="tre5" to="nter4"/>
    <edge from="nter5" head="yes" id="tre6" to="nter3"/>
    <edge from="ter2" id="tre7" to="nter5"/>
    <edge from="nter6" head="yes" id="tre8" to="nter2"/>
    <edge from="nter7" id="tre9" to="nter6"/>
    <edge from="ter3" id="tre10" to="nter7"/>
    <edge from="nter8" head="yes" id="tre11" to="nter6"/>
    <edge from="nter9" head="yes" id="tre12" to="nter8"/>
    <edge from="nter10" id="tre13" to="nter9"/>
    <edge from="ter4" id="tre14" to="nter10"/>
    <edge from="nter11" head="yes" id="tre15" to="nter9"/>
    <edge from="ter5" id="tre16" to="nter11"/>
    <edge from="nter12" id="tre17" to="nter8"/>
    <edge from="nter13" head="yes" id="tre18" to="nter12"/>
    <edge from="ter6" id="tre19" to="nter13"/>
    <edge from="nter14" id="tre20" to="nter12"/>
    <edge from="nter15" head="yes" id="tre21" to="nter14"/>
    <edge from="ter7" id="tre22" to="nter15"/>
    <edge from="nter16" id="tre23" to="nter2"/>
    <edge from="ter8" id="tre24" to="nter16"/>
  </tree>
</constituency>
```

A.9 ixa-pipe-srl

The ixa-pipe-srl provides dependency parsing. It annotates dependency relations among terms. For that, it reads the "lemma" and "pos" attributes of the <term> elements. For Spanish, it also works with the "morphofeat" attribute of the <term> elements. Each dependency is represented by an empty <dep> element and span previous terms. The <dep> element has the from, to and rfunc attributes. Example:

```
  <deps>
    <!--NAME(Rajoy,Mariano)-->
    <dep from="t2" rfunc="NAME" to="t1"/>
    <!--SBJ(is,Rajoy)-->
    <dep from="t3" rfunc="SBJ" to="t2"/>
    <!--NAME(Spain,the)-->
    <dep from="t7" rfunc="NAME" to="t4"/>
    <!--NAME(Spain,President)-->
    <dep from="t7" rfunc="NAME" to="t5"/>
    <!--NAME(Spain,of)-->
    <dep from="t7" rfunc="NAME" to="t6"/>
    <!--PRD(is,Spain)-->
    <dep from="t3" rfunc="PRD" to="t7"/>
    <!--P(is,.)-->
    <dep from="t3" rfunc="P" to="t8"/>
  </deps>
```

The ixa-pipe-srl also provides semantic roles. For that, it reads the “lemma” and “pos” attributes of the <term> elements and the <dep> elements. For Spanish, it also works with the “morphofeat” attribute of the <term> elements. Each annotate predicate is represented by an <predicate> element. The <predicate> element has the id attribute. It also has the <externalReferences>, <span> and <role> elements. The <span> element contains one or more <target> elements with the id attribute. The <role> element represents filler of a particular argument of the predicate and it has the id attribute and the <externalReferences> and <span> sub-elements.

```
    <predicate id="pr1">
      <!--elected-->
      <externalReferences>
        <externalRef reference="elect.01" resource="PropBank"/>
        <externalRef reference="appoint-29.1" resource="VerbNet"/>
        <externalRef reference="Change_of_leadership" resource="FrameNet"/>
        <externalRef reference="elect.01" resource="PropBank"/>
        <externalRef reference="contextual" resource="EventType"/>
      </externalReferences>
      <span>
        <target id="t4"/>
      </span>
      <role id="rl1" semRole="A1">
        <!--Mariano Rajoy-->
        <externalReferences>
          <externalRef reference="appoint-29.1#Theme" resource="VerbNet"/>
          <externalRef reference="Change_of_leadership-New_leader" resource="FrameNet"/>
          <externalRef reference="elect.01#1" resource="PropBank"/>
        </externalReferences>
        <span>
          <target id="t1"/>
          <target head="yes" id="t2"/>
        </span>
      </role>
      <role id="rl2" semRole="AM-TMP">
        <!--on 20 November 2011-->
        <span>
          <target head="yes" id="t5"/>
          <target id="t6"/>
          <target id="t7"/>
          <target id="t8"/>
        </span>
      </role>
    </predicate>
```

A.10 vua-framenet-classifier

The vua-framenet-classifier adds FrameNet roles to the semantic role layer. In this case, it maps PropBank arguments (provided by the SRL module) to FrameNet roles. Sample output for Dutch is illustrated below. Note that the PropBank roles are mapped to English FrameNet frames (since there currently is no FrameNet for Dutch).

```
<srl>
    <predicate id="pr1"> <!-- given by the propbank srl -->
      <!--varen-->
      <externalReferences>
        <externalRef reference="Operate_vehicle" resource="FrameNet"/> <!-- to be added ↩
            by Chantal -->
      </externalReferences>
      <span>
        <target id="t1.49"/>
      </span>
      <role id="rl1" semRole="A0"> <!-- given by the propbank srl -->
        <!--passanten-->
        <externalReferences>
          <externalRef reference="Operate_vehicle#Driver" resource="FrameNet"/> <!-- to ↩
              be added by Chantal -->
        </externalReferences>
        <span>
          <target head="yes" id="t1.21"/>
        </span>
      </role>
      <role id="rl2" semRole="AM-LOC"> <!-- given by the propbank srl -->
        <!--langs het meer -->
        <externalReferences>
          <externalRef reference="Operate_vehicle#Area" resource="FrameNet"/> <!-- to be↩
               added by Chantal -->
        </externalReferences>
        <span>
          <target head="yes" id="t1.45"/>
          <target id="t1.46"/>
          <target id="t1.47"/>
        </span>
      </role>
    </predicate>
  </srl>
```

A.11 VUA-alpino

The VUA-alpino is a morphosyntactic analyzer that performs POS-tagging, constituent parsing and depencency parsing at the same time. It reads the text layer annotated with tokens as input, and generate the term, constituency and dependency layers. The output of Alpino looks slightly different from the output provided by the MATE tools that are used for other languages.

The <term> has the same structure and attributes as for other languages consisting of mainly the lemma of each token, its part-of-speech tag (POS) and the morphofeat attribute, but it uses a different set of values. The pos attribute assigns Alpino's basic POS-tag and a more complex pos-tag assigned by Alpino is stored in the morphofeat attribute. A complete overview of the values that Alpino outputs can be found in [van Noord *et al.*, 2010] (in Dutch). The term layer that Alpino outputs is illustrated below.

```
<terms>
  <term id="t_0" lemma="Mariano" pos="name" morphofeat="SPEC(deeleigen)" type="open">
    <span>
      <target id="w1"/>
    </span>
  </term>
  <term id="t_1" lemma="Rajoy" pos="name" morphofeat="SPEC(deeleigen)" type="open">
    <span>
      <target id="w2"/>
    </span>
  </term>
  <term id="t_2" lemma="," pos="punct" morphofeat="LET()" type="open">
    <span>
      <target id="w3"/>
    </span>
  </term>
  <term id="t_3" lemma="die" pos="pron" morphofeat="VNW(betr,pron,stan,vol,persoon,↩
    getal)" type="close">
    <span>
      <target id="w4"/>
    </span>
  </term>
  ...
  </terms>
```

The overall structure of constituent trees and dependencies coming out of Alpino also respect the basic structure of the parsers for other languages. Here, the difference lies in the treatment of punctuation. It is important to take this difference into account, because Alpino trees have more daughter nodes under the head node than trees produced for other languages. Largely language independent modules that make use of syntactic information may thus have to be adapted for Dutch (e.g. the corefgraph module). We will illustrate and describe both layers below.

In the constituency layer one <tree> element is found for each sentence. Inside each <tree>, there are three types of elements: a) <nt> elements representing non-terminal nodes; b) <t> elements representing terminal nodes; and c) <edge> elements representing in-tree edges. The <nt> element has the id and label attributes. The <t> element has the id attribute and the <span> element pointing to the term layer. Finally, the <edge> element has the id, from, to and head attributes.

```
<constituency>
  <tree>
  <t id="ter2">
    <span>
      <target id="t_0"/>
    </span>
  </t>
  <t id="ter3">
    <span>
      <target id="t_1"/>
    </span>
  </t>
  ...
  <edge id="tre0" from="nter1" to="nter0">
    <!--  tre0  top <- punct -->
  </edge>
  <edge id="tre1" from="ter0" to="nter1">
    <!--  tre1  punct <- , -->
  </edge>
    <edge id="tre8" from="ter2" to="nter6">
    <!--  tre8  name <- Mariano -->
   </edge>
    <edge id="tre9" from="nter7" to="nter5">
    <!--  tre9  mwu <- name -->
  </edge>
  <edge id="tre10" from="ter3" to="nter7">
    <!--  tre10  name <- Rajoy -->
  </edge>
</constituency>
```

Finally, in the dependency layer the dependencies found between the terms are represent by means of the elements <dep>. The elements are “to” and “from” representing the two words related by means of a dependency function, encoded in the attribute “rfunc”. Below we can see an example:

```
  <deps>
    <dep from="t_2" to="t_8" rfunc="-- / --">
    <!-- - - / - -(punct:,,punct:,) -->
  </dep>
  <dep from="t_2" to="t_9" rfunc="-- / --">
    <!-- - - / - -(punct:,,verb:ben) -->
  </dep>
   <dep from="t_9" to="t_0" rfunc="hd/su">
    <!-- hd/su(verb:ben,name:Mariano) -->
  </dep>
    <dep from="t_9" to="t_10" rfunc="hd/predc">
    <!-- hd/predc(verb:ben,noun:president) -->
  </dep>
  </deps>
```

We can see how “Mariano Rajoy” is grouped as a multiword unit (mwu) in the constituency layer, and two dependencies stating that “Mariano” is the subject of the main verb of the sentence and the predicate of the verb is “president”. As mentioned above, note that Alpino links the punctuation symbols to the root element of the sentence in both the constituency and the dependency tree.

A.12 Corefgraph and vua-event coreference

The corefgraph module provides clusters of terms which share the same referent. For that, it reads the “morphofeat” attribute of the <term> elements, the <entity> and <constituent> elements. It outputs the <coref> element that has the id attribute and it also contains <span> elements. Each <span> contains one or more <target> element, each one with the id attribute. Example:

```
  <coref id="co1">
     <span>
       <!--the Prime Minister of Spain-->
       <target id="t4"/>
       <target id="t5"/>
       <target id="t6"/>
       <target id="t7"/>
       <target id="t8"/>
     </span>
     <span>
       <!--leader of the People s Party-->
       <target id="t19"/>
       <target id="t20"/>
       <target id="t21"/>
       <target id="t22"/>
       <target id="t23"/>
       <target id="t24"/>
     </span>
     <span>
       <!--Mariano Rajoy-->
       <target id="t1"/>
       <target id="t2"/>
     </span>
     <span>
       <!--He-->
       <target id="t16"/>
     </span>
   </coref>
```

The event coreference module extends this layer further, applying the exact same structure for events:

```
  <span>
       <!--playing-->
       <target id="t848"/>
     </span>
     <span>
       <!--mentored-->
       <target id="t1006"/>
     </span>
     <span>
       <!--contributions-->
       <target id="t1026"/>
     </span>
```

A.13 fbk-timepro

The TimePro module provides temporal expressions and their normalization. The <timex3> element is used to represent a temporal expression in the document. The <timex3> element has 5 attributes: id, type and value. This element is either empty (for example to represent the date of creation of the document) or contains a <span> element spaning words. Example:

```
<timeExpressions>
  <timex3 id="tmx0" type="DATE" value="2013-03-22" functionInDocument="CREATION_TIME" />
  <timex3 id="tmx1" type="DURATION" value="P1M">
<!-- a month -->
    <span>
      <target id="w26" />
      <target id="w27" />
    </span>
  </timex3>
</timeExpressions>
```

A.14 fbk-temprel

The TempRel module provides temporal relations between events and time expressions. The <tlink> element is used to represent a temporal relation in the document. The <tlink> is an empty element which has 6 attributes: id, from, to, fromType, toType and relType. Example:

```
<temporalRelations>
<tlink from="pred10" to="pred11" relType="BEFORE" fromType="event" toType="event" id="↩
    tlink0"/>
    <tlink from="pred1" to="tmx0" relType="AFTER" fromType="event" toType="timex" id="↩
        tlink1"/>
</temporalRelations>
```

A.15 fbk-eventpro

The EventPro module provides events. The <event> element is used to represent a event in the document. The <event> element has the id and class attributes. It also has the <references> sub-element which contains one or more <span> element, each one spanning terms. Example:

```
  <events>
    <event id="ev1" class="REPORTING">
      <references>
        <span>
          <!-- announcement -->
          <target id="t10"/>
        </span>
      </references>
    </event>
```

A.16 fbk-causalrel

The CausalRel module provides causal relations between events. The <clink> element is used to represent a causal relation in the document. The <clink> is an empty element which has 4 attributes: id, from, to and relType. Example:

```
<causalRelations>
```

```
  <clink from="pred10" to="pred11" relType="CAUSE" id="clink0"/>
</causalRelations>
```

A.17 fbk-factpro

The fbk-factpro provides attribution values for each event. For that, it reads the “event” elements. Each event attribution value is represented by an empty <factvalue> element which has the “eventid”, “certainty”, “polarity”, “time” and “specialCases” attributes. Example:

```
  <factualitylayer>
    <factvalue eventid="e1" certainty="CERTAIN" polarity="POS" time="NON-FUTURE" ↩
        specialCases="NONE"/>
    <factvalue eventid="e2" certainty="CERTAIN" polarity="POS" time="FUTURE" ↩
        specialCases="COND_MAIN_CLAUSE"/>
```

A.18 vua-factuality

A factuality module that will provide the final output for English (and Dutch) is under development. We currently use a module that produces FactBank based output. The NAF representation is illustrated below:

```
<factualitylayer>
<factvalue id="w179" prediction="CT+" confidence="0.89"/>
<factvalue id="w108" prediction="CT+" confidence="0.93"/>
</factualitylayer>
```

A.19 vua-opinions

The xml representation below illustrates potential output of the opinion miner. The opinion_holder indicates who has the opinion, the opinion_target indicates what the opinion is about and the span of the opinion_expression points to the terms that express the opinion. Typically, only the opinion_expression will be present.

```
<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="1">
      <span>
        <target id="t3"/>
        <target id="t4"/>
      </span>
    </opinion_expression>
  </opinion>
</opinions>
```

A.20 fbk-tokenpro

The fbk-tokenpro provides sentence segmentation and tokenization given a raw text. It produces one line by token containing the following columns: token, token id, token offset start, token offset end, token type (the status of the token as lower or upper case) and token's normlization.

token	*token id*	*token start*	*token end*	*token type*	*token norm*
Steve	1	0	5	UPP	Steve
Jobs	2	6	10	UPP	Jobs
was	3	11	14	LOW	was
the	4	15	18	LOW	the
CEO	5	19	22	CAP	CEO
of	6	23	25	LOW	of
Apple	7	26	31	UPP	Apple
.	8	31	32	PUN	.

A.21 fbk-morphopro

The fbk-morphopro provides all possible morphological analysis of each word of a text. The fbk-morphopro takes in input the output of the fbk-tokenpro and produces a new column containing all possible morphological analysis separated by a space character.

token	*possible morph analyses*
the	the+adv the+art
CEO	ceo+pn

A.22 fbk-tagpro

The fbk-tagpro provides POS tagging. It uses the output of the fbk-tokenpro module and produces a new column containing PoS tags.

token	*PoS*
the	AT0
CEO	NP0

A.23 fbk-lemmapro

The fbk-lemmapro provides lemmatization and morphological analysis. It takes in input the token form, the POS and the possible morphological analysis of each token and adds two columns: all compatible morphological analyses (separated by a space character) and the lemma.

token	*possible morph analyses*	*PoS*	*compatible morph analyses*	*lemma*
the	the+adv the+art	AT0	the+art	the
CEO	ceo+pn	NP0	ceo+pn	ceo

A.24 fbk-entitypro

The fbk-entitypro module provides named entities. For that, it uses token, token's normalization, POS and lemma columns. The output is a new column with named entities labelled following the IOB-2 format.

token	*token norm*	*PoS*	*lemma*	*entity*
Steve	Steve	NP0	Steve	B-PER
Jobs	Jobs	NP0	Jobs	I-PER
was	was	VBD	be	O

A.25 fbk-chunkpro

The fbk-chunkpro module provides chunking. It takes tokens and POS columns as input and produces a new column with the chunk labels (nominal phrase or verbal phrase) using the IOB-2 format. The exact output is illustrated below:

token	*POS*	*chunk*
Steve	NP0	B-NP
Jobs	NP0	I-NP
was	VBD	B-VP

A.26 fbk-depparserpro

The fbk-depparserpro provides dependency parsing. It annotates dependency relations among terms. It uses token, POS and lemma columns and it adds three columns as output: the token id from the parser, the head's id and the dependency relation.

token	*PoS*	*lemma*	*parser id*	*head id*	*dep relation*
Steve	NP0	Steve	1	3	SUBJ
Jobs	NP0	Jobs	2	1	CONTIN-DENOM
was	VBD	be	3	0	ROOT

References

[Agerri *et al.*, 2014] 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, May 26-31 2014.

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

[Bethard, 2013] Steven Bethard. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 821–826, Seattle, Washington, USA, 2013. Association for Computational Linguistics. (Acceptance rate 24

[Bizer *et al.*, 2009] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[Björkelund *et al.*, 2009] Anders Björkelund, Love Hafdell, and Pierre Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Björkelund *et al.*, 2010] 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.

[Bohnet, 2010] Bernd Bohnet. Very high accuracy and fast dependency parsing is not a contradiction. In *The 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China, 2010.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[Carreras *et al.*, 2002] Xavier Carreras, Lluís Màrquez, and Lluís Padró. Named entity extraction using adaboost. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[Clercq *et al.*, 2012] Orphée De Clercq, Veronique Hoste, and Paola Monachesi. Evaluating automatic cross-domain semantic role annotation. In *Proceedings of the 8th International*

Conference on Language Resources and Evaluation Conference (LREC-2012), pages 88–93, Istanbul, Turkey, 2012.

[Collins, 1999] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[Collins, 2002] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[Cowan and Collins, 2005] Brooke Cowan and Michael Collins. Morphology and reranking for the statistical parsing of spanish. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 795–802, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Cybulska and Vossen, 2013] Agata Cybulska and Piek Vossen. Semantic relations between events and their time, locations and participants for event coreference resolution. In G. Angelova, K. Bontcheva, and R. Mitkov, editors, *Proceedings of Recent Advances in Natural Language Processing (RANLP-2013)*, number ISSN 1313-8502, Hissar, Bulgaria, Sept 7-14 2013. INCOMA Ltd.

[Daelemans and van den Bosch, 2005] Walter Daelemans and Antal van den Bosch. *Memory-Based Language Processing*. Cambridge University Press, 2005.

[Daiber *et al.*, 2013] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.

[de Lacalle *et al.*, 2014] Maddalen López de Lacalle, Egoitz Laparra, and German Rigau. First steps towards a predicate matrix. In *Proceedings of the 7th Global WordNet Conference (GWC2014)*, Tartu, Estonia, 2014.

[Erk and Pado, 2004] Katrin Erk and Sebastian Pado. A powerful and versatile xml format for representing role-semantic annotation. In *Proceedings of LREC-2004*, Lisbon, 2004.

[Fellbaum, 1998] C. Fellbaum, editor. *WordNet. An Electronic Lexical Database*. The MIT Press, 1998.

[Ferro *et al.*, 2002] Lisa Ferro, Laurie Gerber, Inderjeet Mani, Beth Sundheim, and George Wilson. Tides. instruction manual for the annotation of temporal expressions. In *Technical Report Interim Draft for Terqas Workshop*. The MITRE Corporation, 2002.

[Fillmore, 1976] Charles J. Fillmore. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, volume 280, pages 20–32, New York, 1976.

[Fokkens *et al.*, 2014] Antske Fokkens, Aitor Soroa, Zuhaitz Beloki, Niels Ockeloen, German Rigau, Willem Robert van Hage, and Piek Vossen. NAF and GAF: Linking linguistic annotations. In *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, page 9, Reykjavik, Iceland, 2014.

[Giménez and Màrquez, 2004] Jesús Giménez and Lluís Màrquez. SVMTool: A general pos tagger generator based on support vector machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46, 2004.

[Giuglea and Moschitti, 2006] Ana-Maria Giuglea and Alessandro Moschitti. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of COLING-ACL 2006*, pages 929–936, Morristown, NJ, USA, 2006. ACL.

[Gonzalez-Agirre *et al.*, 2012] Aitor Gonzalez-Agirre, Egoitz Laparra, and German Rigau. Multilingual central repository version 3.0. In *LREC*, pages 2525–2529, 2012.

[Gurevych *et al.*, 2012] Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M Meyer, and Christian Wirth. Uby: A large-scale unified lexical-semantic resource based on lmf. In *Proceedings of EACL*, pages 580–590, 2012.

[Juan Aparicio and Martí, 2008] Mariona Taulé Juan Aparicio and M.Antònia Martí. Ancora-verb: A lexical resource for the semantic annotation of corpora. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008.

[Kipper, 2005] Karen Kipper. *VerbNet: A broad-coverage, comprehensive verb lexicon.* PhD thesis, University of Pennsylvania, 2005.

[Lacalle *et al.*, 2014] Maddalen López De Lacalle, Egoitz Laparra, and German Rigau. Predicate matrix: extending semlink through wordnet mappings. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, 2014.

[Laparra and Rigau, 2009] Egoitz Laparra and German Rigau. Integrating wordnet and framenet using a knowledge-based word sense disambiguation algorithm. In *Proceedings of RANLP*, Borovets, Bulgaria, 2009.

[Laparra and Rigau, 2013] Egoitz Laparra and German Rigau. Impar: A deterministic algorithm for implicit semantic role labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 33–41, 2013.

[Laparra *et al.*, 2010] Egoitz Laparra, German Rigau, and Montse Cuadros. Exploring the integration of wordnet and framenet. In *Proceedings of the 5th Global WordNet Conference (GWC 2010), Mumbai, India*, 2010.

[Lavelli *et al.*, 2013] Alberto Lavelli, Johan Hall, Jens Nilsson, and Joakim Nivre. Maltparser at the evalita 2009 dependency parsing task. In *Proceedings of the EVALITA 2009 Workshop on Evaluation of NLP Tools for Italian*, 2013.

[Lee *et al.*, 2011] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, 2011.

[Lee *et al.*, 2013] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, pages 1–54, January 2013.

[Levin, 1993] Beth Levin. *English verb classes and alternations: A preliminary investigation*, volume 348. University of Chicago press Chicago, 1993.

[López de Lacalle *et al.*, 2014] Maddalen López de Lacalle, Egoitz Laparra, and German Rigau. Extending semlink through wordnet mappings. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014)*, Reykjavik, Iceland, May 26-31 2014.

[Magnini *et al.*, 2006] Bernardo Magnini, Emanuele Pianta, Christian Girardi, Matteo Negri, Lorenza Romano, Manuela Speranza, Valentina Bartalesi Lenzi, and Rachele Sprugnoli. I-cab: The italian content annotation bank. In *Proceedings of LREC*, pages 963–968, 2006.

[Màrquez *et al.*, 2007] Lluís Màrquez, Luis Villarejo, M. A. Martí, and Mariona Taulé. Semeval-2007 task 09: Multilevel semantic annotation of catalan and spanish. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 42–47, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[McCallum, 2002] Andrew K. McCallum. MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`, 2002.

[Meyers *et al.*, 2004] Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The nombank project: An interim report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*, 2004.

[Mirza and Tonelli, 2014a] Paramita Mirza and Sara Tonelli. An analysis of causality between events and its relation to temporal information. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING2014)*, Dublin, Ireland, August 23-29 2014.

[Mirza and Tonelli, 2014b] Paramita Mirza and Sara Tonelli. Classifying temporal relations with simple features. In *Proceedings of the 14th Conference of the European*

Chapter of the Association for Computational Linguistics, EACL-14, pages 308–317, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

[Navigli and Ponzetto, 2010] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of ACL*, pages 216–225, 2010.

[Navigli *et al.*, 2007] Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 30–35, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[Oostdijk *et al.*, 2008] Nelleke Oostdijk, Martin Reynaert, Paola Monachesi, Gertjan Van Noord, Roeland Ordelman, Ineke Schuurman, and Vincent Vandeghinste. From d-coi to sonar: a reference corpus for dutch. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[Palmer *et al.*, 2005] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, March 2005.

[Palmer, 2009] Martha Palmer. Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the Generative Lexicon Conference*, pages 9–15, 2009.

[Pianta and Zanoli, 2007] Emanuele Pianta and Roberto Zanoli. Tagpro: a system for italian pos tagging based on svm. volume 4, pages 8–9. Associazione Italiana per l'Intelligenza Artificiale, 2007.

[Pianta *et al.*, 2008] Emanuele Pianta, Christian Girardi, and Roberto Zanoli. The textpro tool suite. In *Proceedings of LREC, 6th edition of the Language Resources and Evaluation Conference*, LREC-08, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[Pitler and Nenkova, 2009] Emily Pitler and Ani Nenkova. Using syntax to disambiguate explicit discourse connectives in text. In *In Proceedings of the ACL-IJCNLP 2009*, ACL-09, pages 13–16, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Raghunathan *et al.*, 2010] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501, 2010.

[Ratnaparkhi, 1999] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

[Shi and Mihalcea, 2004] Lei Shi and Rada Mihalcea. An algorithm for open text semantic parsing. In Vincenzo Pallotta and Amalia Todirascu, editors, *COLING 2004 3rd Workshop on Robust Methods in Analysis of Natural Language Data*, pages 59–67, Geneva, Switzerland, August 29th 2004. COLING.

[Soroa and Fernández, 2014] Aitor Soroa and Enrique Fernández. Newsreader virtual machines. Technical Report NWR-2014-4, The University of the Basque Country, 2014.

[Steinberger *et al.*, 2012] Ralf Steinberger, Mohamed Ebrahim, and Marco Turchi. Jrc eurovoc indexer jex - a freely available multi-label categorisation tool. In *LREC'12*, pages 798–805, 2012.

[Strötgen and Gertz, 2013] Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[Strötgen *et al.*, 2013] Jannik Strötgen, Julian Zell, and Michael Gertz. Heideltime: Tuning english and developing spanish resources for tempeval-3. In *Proceedings of the Seventh International Workshop on Semantic Evaluation, SemEval '13*, pages 15—19, Stroudsburg, PA, USA, 2013. Association for Computational Linguistics.

[Suchanek *et al.*, 2007] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *WWW conference*, New York, NY, USA, 2007. ACM Press.

[Sundheim, 1996] Beth M. Sundheim. Overview of results of the muc-6 evaluation. In *Proceedings of a Workshop on Held at Vienna, Virginia: May 6-8, 1996*, TIPSTER '96, pages 423–442, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[Taulé *et al.*, 2008] Mariona Taulé, Maria Antònia Martí, and Marta Recasens. Ancora: Multilevel annotated corpora for catalan and spanish. In *LREC*, 2008.

[Toutanova *et al.*, 2003] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259, 2003.

[van de Camp and Christiansen, 2013] Matje van de Camp and Henning Christiansen. Resolving relative time expressions in dutch text with constraint handling rules. In *Constraint Solving and Language Processing*, pages 166–177. Springer, 2013.

[van Noord *et al.*, 2010] Gertjan van Noord, Ineke Schuurman, and Gosse Bouma. Lassy syntactische annotatie. Technical report, Technical Report 19455, University of Groningen, 2010.

[Zhang and Johnson, 2003] Tong Zhang and David Johnson. A robust risk minimization based named entity recognition system. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 204–207, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.