Event Detection, version 3
Deliverable D4.2.3
Version FINAL

Authors: Rodrigo Agerri¹, Itziar Aldabe¹, Zuhaitz Beloki¹, Egoitz Laparra¹, German Rigau¹, Aitor Soro¹, Marieke van Erp², Antske Fokkens², Filip Ilievski², Ruben Izquierdo², Roser Morante², Chantal van Son², Piek Vossen², Anne-Lyse Minard³

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

Building structured event indexes of large volumes of financial and economic data for decision making
ICT 316404
<table>
<thead>
<tr>
<th><strong>Grant Agreement No.</strong></th>
<th>316404</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Acronym</strong></td>
<td>NEWSREADER</td>
</tr>
<tr>
<td><strong>Project Full Title</strong></td>
<td>Building structured event indexes of large volumes of financial and economic data for decision making.</td>
</tr>
<tr>
<td><strong>Funding Scheme</strong></td>
<td>FP7-ICT-2011-8</td>
</tr>
<tr>
<td><strong>Project Website</strong></td>
<td><a href="http://www.newsreader-project.eu/">http://www.newsreader-project.eu/</a></td>
</tr>
</tbody>
</table>
| **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 D4.2.3 |
| **Status & Version**    | FINAL |
| **Contractual Date of Delivery** | June 2015 |
| **Actual Date of Delivery** | February 2016 |
| **Type**                | Report |
| **Security (distribution level)** | Public |
| **Number of Pages**     | 158 |
| **WP Contributing to the Deliverable** | WP4 |
| **WP Responsible**      | EHU |
| **Authors:** Rodrigo Agerri, Itziar Aldabe, Zuhaitz Beloki, Egoitz Laparra, German Rigau, Aitor Soroa, Marieke van Erp, Antske Fokkens, Filip Ilievski, Ruben Izquierdo, Roser Morante, Chantal van Son, Piek Vossen, Anne-Lyse Minard |
| **Keywords:** Event detection, EN pipelines, NL pipeline, ES pipeline, IT pipeline, Evaluation, Scaling of text processing |

**Abstract:** This deliverable describes the third prototype for event detection. It includes pipelines for event detection in 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. The evaluation results on the MEANTIME dataset are also presented.
## Table of Revisions

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description and reason</th>
<th>By</th>
<th>Affected sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>30 June 2015</td>
<td>Structure of the deliverable set</td>
<td>EHU</td>
<td>All</td>
</tr>
<tr>
<td>0.2</td>
<td>30 July 2015</td>
<td>Adapted description Dutch Heideltime, included new English Factivity and NAF representations</td>
<td>Antske Fokkens</td>
<td>3-4</td>
</tr>
<tr>
<td>0.2</td>
<td>30 July 2015</td>
<td>Updated Dutch module dependencies figure, included description srl-dutch-nominal-events</td>
<td>Marieke van Erp</td>
<td>4</td>
</tr>
<tr>
<td>0.2</td>
<td>31 July 2015</td>
<td>Factuality evaluation</td>
<td>Antske Fokkens</td>
<td>7</td>
</tr>
<tr>
<td>0.2</td>
<td>31 July 2015</td>
<td>Time evaluation, Italian evaluation</td>
<td>Anne-Lyse Minard</td>
<td>7</td>
</tr>
<tr>
<td>0.2</td>
<td>31 July 2015</td>
<td>Internal revision</td>
<td>Itziar Aldabe</td>
<td>All</td>
</tr>
<tr>
<td>0.2</td>
<td>25 September 2015</td>
<td>Added POCUS</td>
<td>Filip Ilievski</td>
<td>3</td>
</tr>
<tr>
<td>0.3</td>
<td>19 November 2015</td>
<td>Post-processing revisions</td>
<td>Itziar Aldabe</td>
<td>3-5, 7, 9</td>
</tr>
<tr>
<td>1.0</td>
<td>7 January 2016</td>
<td>Dutch nominal event detection</td>
<td>Piek Vossen</td>
<td>4</td>
</tr>
<tr>
<td>1.0</td>
<td>11 January 2016</td>
<td>Event detection revisions</td>
<td>Itziar Aldabe</td>
<td>2</td>
</tr>
<tr>
<td>1.0</td>
<td>13 January 2016</td>
<td>Updates IXA tokenizer, pos-tagger and parser</td>
<td>Itziar Aldabe</td>
<td>3-6, 4</td>
</tr>
<tr>
<td>1.0</td>
<td>12 January 2016</td>
<td>New WSD and opinion mining modules (English, Dutch)</td>
<td>Ruben Izquierdo</td>
<td>3, 4</td>
</tr>
<tr>
<td>1.0</td>
<td>13 January 2016</td>
<td>Updates IXA tokenizer, pos-tagger and parser</td>
<td>Itziar Aldabe</td>
<td>3, 6, 4</td>
</tr>
<tr>
<td>1.0</td>
<td>14 January 2016</td>
<td>Event detection and Dutch Predicate Matrix revisions</td>
<td>Piek Vossen</td>
<td>2</td>
</tr>
<tr>
<td>1.0</td>
<td>14 January 2016</td>
<td>Updates English and Spanish evaluation</td>
<td>Egoitz Laparra</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>14 January 2016</td>
<td>Added Dutch evaluation</td>
<td>Antske Fokkens</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>15 January 2016</td>
<td>Updates FBK tools and Italian evaluation</td>
<td>Anne-Lyse Minard</td>
<td>3, 5, 7, 9, A</td>
</tr>
<tr>
<td>1.0</td>
<td>15 January 2016</td>
<td>Update coregraph module</td>
<td>Rodrigo Agerri</td>
<td>3</td>
</tr>
<tr>
<td>1.0</td>
<td>18 January 2016</td>
<td>Update evaluations, revision Dutch evaluation</td>
<td>Itziar Aldabe</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>19 January 2016</td>
<td>Revision deliverable</td>
<td>Antske Fokkens</td>
<td>All</td>
</tr>
<tr>
<td>1.0</td>
<td>27 January 2016</td>
<td>Update deliverable based on revision</td>
<td>Itziar Aldabe, Rodrigo Agerri, Egoitz Laparra, Anne-Lyse Minard, Antske Fokkens</td>
<td>All</td>
</tr>
<tr>
<td>1.0</td>
<td>1 February 2016</td>
<td>Check by coordinator</td>
<td>VUA</td>
<td>-</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable describes the third 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 deliverable follows the same structure as the previous deliverables on event detection. This means, in particular, that each module specifies exactly what input and dependencies it needs and what output it produces. This structure ensures that anyone interested in using our software can build their own pipeline. The main new contribution of this version are the evaluation results of the four pipelines of the project.

The third 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 and normalization, named entity recognition, word sense disambiguation, named entity disambiguation, coreference resolution, semantic role labeling, temporal and causal relation extraction, topic identification, event classification, and opinion mining. We have integrated new functionalities and modules so that the pipeline incorporates re-ranking of named entity disambiguation and wikification and designed a new module for factuality classification. Just like in the previous versions of the pipeline, each task is executed by one module, which allows us to custom different pipeline topologies for text processing.

We have also worked on the improvement of the Dutch, Italian and Spanish pipelines. All the pipelines include the necessary modules to perform event detection. In addition, the multilingual interoperable semantic interpretation of the information is guaranteed in all the languages of the project by projecting entities, event mentions and time expressions to language independent knowledge representations.

In the last cycle of the project we have carried out a complete evaluation of the four pipelines in standard datasets as well as on the MEANTIME corpus. We present state-of-the-art pipelines for event extraction in all four languages. For some modules, we report results far beyond current state-of-the-art. As expected, English results are usually better since usually there are more and more complete resources for this language. Results for the other languages, however, come very close to those obtained for English. Surprisingly, on some tasks, Italian and Spanish results are even higher than those obtained for English in our MEANTIME evaluation. As far as we know, these are no most complete NLP pipelines for event extraction in all four languages.

Finally, we have continued collecting and processing different datasets. In the last year of the project, we have processed over 3.5 Million articles in English and 627K articles in Dutch. We have also processed the Dutch, English, Italian and Spanish wikinews articles.

Table 1 summarizes the main updates on this deliverable with respect to the previous versions on event detection.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description of the update</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Event detection</td>
<td>Brief explanation of the methods to construct the Predicate Matrix and the strategy followed to construct the Dutch part of it. Description of the process to perform time normalization.</td>
</tr>
<tr>
<td>3 English NLP processing</td>
<td>New modules included: it_makes_sense-WSD (3.7), POCUS (3.10), ixa-pipe-wikify (3.11), vua-factuality (3.17)</td>
</tr>
<tr>
<td>4 Dutch NLP processing</td>
<td>New modules included: vua-srl-dutch-nominal-events (4.12)</td>
</tr>
<tr>
<td>5 Italian NLP processing</td>
<td>New modules included: fbk-srl (5.13), fbk-eventcoref (5.15), fbk-timeanchor (5.16), ixa-pipe-ned (5.12), ixa-pipe-wikify (5.14)</td>
</tr>
<tr>
<td>6 Spanish NLP processing</td>
<td>New modules included: ixa-pipe-wikify (6.9)</td>
</tr>
<tr>
<td>7 Evaluation</td>
<td>This is a complete new section where the evaluation results of the four pipelines of the project are described</td>
</tr>
</tbody>
</table>

Table 1: Main updates on deliverable with respect to the previous versions on event detection
# Contents

Table of Revisions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>11</td>
</tr>
<tr>
<td>2 Event Detection</td>
<td>12</td>
</tr>
<tr>
<td>2.1 Cross-lingual Interoperability</td>
<td>14</td>
</tr>
<tr>
<td>2.1.1 Cross-lingual Predicate Models</td>
<td>14</td>
</tr>
<tr>
<td>2.1.2 Dutch Predicate Matrix</td>
<td>16</td>
</tr>
<tr>
<td>2.1.3 Interoperability of Entities</td>
<td>18</td>
</tr>
<tr>
<td>2.1.4 Normalization of time expressions</td>
<td>21</td>
</tr>
<tr>
<td>3 English NLP Processing</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Tokenizer</td>
<td>22</td>
</tr>
<tr>
<td>3.2 POS tagging</td>
<td>23</td>
</tr>
<tr>
<td>3.3 Constituency Parser</td>
<td>24</td>
</tr>
<tr>
<td>3.4 Dependency Parser</td>
<td>25</td>
</tr>
<tr>
<td>3.5 Time expression detection and normalization</td>
<td>25</td>
</tr>
<tr>
<td>3.6 Named Entity Recognition and Classification</td>
<td>26</td>
</tr>
<tr>
<td>3.7 Word Sense Disambigation I</td>
<td>27</td>
</tr>
<tr>
<td>3.8 Word Sense Disambigation II</td>
<td>28</td>
</tr>
<tr>
<td>3.9 Named Entity Disambiguation I</td>
<td>29</td>
</tr>
<tr>
<td>3.10 Named Entity Disambiguation II</td>
<td>30</td>
</tr>
<tr>
<td>3.11 Wikification</td>
<td>31</td>
</tr>
<tr>
<td>3.12 Coreference Resolution</td>
<td>32</td>
</tr>
<tr>
<td>3.13 Semantic Role Labeling</td>
<td>33</td>
</tr>
<tr>
<td>3.14 Event coreference</td>
<td>34</td>
</tr>
<tr>
<td>3.15 Temporal relation extraction</td>
<td>35</td>
</tr>
<tr>
<td>3.16 Causal relation extraction</td>
<td>36</td>
</tr>
<tr>
<td>3.17 Factuality</td>
<td>37</td>
</tr>
<tr>
<td>3.18 Opinions</td>
<td>38</td>
</tr>
<tr>
<td>3.19 Text Classification</td>
<td>39</td>
</tr>
<tr>
<td>3.20 English Pipeline</td>
<td>40</td>
</tr>
<tr>
<td>4 Dutch NLP Processing</td>
<td>42</td>
</tr>
<tr>
<td>4.1 Tokenizer</td>
<td>42</td>
</tr>
<tr>
<td>4.2 POS tagging, lemmatization and parsing</td>
<td>43</td>
</tr>
<tr>
<td>4.3 Named Entity Recognition and Classification</td>
<td>44</td>
</tr>
<tr>
<td>4.4 Word Sense Disambiguation</td>
<td>45</td>
</tr>
<tr>
<td>4.5 Named Entity Disambiguation</td>
<td>46</td>
</tr>
<tr>
<td>4.6 DBpedia NER</td>
<td>46</td>
</tr>
<tr>
<td>4.7 Time Expressions</td>
<td>47</td>
</tr>
</tbody>
</table>
4.8 PredicateMatrix tagging .................................................. 49
4.9 Semantic Role Labeling .................................................... 49
4.10 Event coreference ......................................................... 50
4.11 FrameNet labelling ....................................................... 51
4.12 Nominal events .......................................................... 52
4.13 Nominal Predicate SRL .................................................. 53
4.14 Opinions ................................................................. 54
4.15 Dutch Pipeline ........................................................... 55

5 Italian NLP Processing ...................................................... 56
5.1 Tokenizer ................................................................. 56
5.2 Morphological analysis .................................................. 57
5.3 POS tagging ............................................................ 57
5.4 Lemmatization .......................................................... 58
5.5 Named Entity Recognition and Classification ...................... 59
5.6 Chunking ............................................................... 60
5.7 Dependency Parser ...................................................... 61
5.8 Time expression detection and normalization ....................... 61
5.9 Event recognizer ........................................................ 62
5.10 Temporal relations extraction ........................................ 63
5.11 Factuality detection ..................................................... 63
5.12 Named Entity Disambiguation ......................................... 63
5.13 Semantic Role Labelling ............................................... 66
5.14 Wikification .............................................................. 67
5.15 Event coreference ....................................................... 67
5.16 Predicate Time Anchor ................................................ 68
5.17 Italian Pipeline ........................................................ 69

6 Spanish NLP Processing .................................................... 70
6.1 Tokenizer ................................................................. 70
6.2 POS tagging ............................................................ 71
6.3 Constituency Parser ..................................................... 72
6.4 Dependency Parser ..................................................... 73
6.5 Time Expressions ....................................................... 73
6.6 Named Entity Recognition and Classification ..................... 73
6.7 Word Sense Disambiguation .......................................... 75
6.8 Named Entity Disambiguation ........................................ 76
6.9 Wikification .............................................................. 77
6.10 Coreference Resolution ................................................ 78
6.11 Semantic Role Labeling ............................................... 78
6.12 Event coreference ..................................................... 79
6.13 Text Classification ..................................................... 80
6.14 Spanish Pipeline ....................................................... 81
7 Evaluation

7.1 English

7.1.1 English Named Entity Recognition and Classification

7.1.2 English Named Entity Disambiguation

7.1.3 English Event detection and coreference

7.1.4 English Nominal coreference

7.1.5 English Semantic Role Labelling

7.1.6 English Temporal processing

7.1.7 English Factuality

7.2 Dutch

7.2.1 Dutch NERC

7.2.2 Dutch Named Entity Disambiguation

7.2.3 Dutch Intra-document Event coreference

7.2.4 Dutch Semantic Role Labelling

7.2.5 Dutch Temporal Processing

7.3 Italian

7.3.1 Italian Named Entity Recognition and Classification

7.3.2 Italian Named Entity Disambiguation

7.3.3 Italian Intra-document Event Coreference

7.3.4 Italian Semantic Role Labelling

7.3.5 Italian Temporal Processing

7.3.6 Italian Factuality

7.4 Spanish

7.4.1 Spanish Named Entity Recognition and Classification

7.4.2 Spanish Named Entity Disambiguation

7.4.3 Spanish Intra-document Event Coreference

7.4.4 Spanish Nominal Coreference

7.4.5 Spanish Semantic Role Labelling

7.4.6 Spanish Temporal Processing

7.5 Multilingual comparison

8 Scaling of Text Processing

9 Conclusions

A Output representation

A.1 ixa-pipe-tok

A.2 ixa-pipe-pos

A.3 vua-wsd

A.4 wsd-ukb and ims-wsd

A.5 vua-ontotagger

A.6 ixa-pipe-nerc

A.7 ixa-pipe-ned and POCUS
<table>
<thead>
<tr>
<th>A.8</th>
<th>dbpedia-ner</th>
<th>134</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.9</td>
<td>ixa-pipe-wikify</td>
<td>135</td>
</tr>
<tr>
<td>A.10</td>
<td>ixa-pipe-parse</td>
<td>135</td>
</tr>
<tr>
<td>A.11</td>
<td>ixa-pipe-srl, vua-srl and nominal-events</td>
<td>137</td>
</tr>
<tr>
<td>A.12</td>
<td>vua-framenet-classifier</td>
<td>138</td>
</tr>
<tr>
<td>A.13</td>
<td>VUA-alpino</td>
<td>139</td>
</tr>
<tr>
<td>A.14</td>
<td>Coregraph and vua-event coreference</td>
<td>141</td>
</tr>
<tr>
<td>A.15</td>
<td>fbk-timepro</td>
<td>142</td>
</tr>
<tr>
<td>A.16</td>
<td>fbk-temprel</td>
<td>142</td>
</tr>
<tr>
<td>A.17</td>
<td>fbk-eventpro</td>
<td>143</td>
</tr>
<tr>
<td>A.18</td>
<td>fbk-causalrel</td>
<td>143</td>
</tr>
<tr>
<td>A.19</td>
<td>fbk-factpro</td>
<td>143</td>
</tr>
<tr>
<td>A.20</td>
<td>fbk-srl</td>
<td>144</td>
</tr>
<tr>
<td>A.21</td>
<td>fbk-eventcoref</td>
<td>144</td>
</tr>
<tr>
<td>A.22</td>
<td>vua-perspective-factuality</td>
<td>145</td>
</tr>
<tr>
<td>A.23</td>
<td>opinion-miner</td>
<td>145</td>
</tr>
<tr>
<td>A.24</td>
<td>ixa-pipe-topic</td>
<td>146</td>
</tr>
<tr>
<td>A.25</td>
<td>fbk-tokenpro</td>
<td>146</td>
</tr>
<tr>
<td>A.26</td>
<td>fbk-morphopro</td>
<td>146</td>
</tr>
<tr>
<td>A.27</td>
<td>fbk-tagpro</td>
<td>147</td>
</tr>
<tr>
<td>A.28</td>
<td>fbk-lemmapro</td>
<td>147</td>
</tr>
<tr>
<td>A.29</td>
<td>fbk-entitypro</td>
<td>147</td>
</tr>
<tr>
<td>A.30</td>
<td>fbk-chunkpro</td>
<td>148</td>
</tr>
<tr>
<td>A.31</td>
<td>fbk-depparserpro</td>
<td>148</td>
</tr>
</tbody>
</table>
1 Introduction

This deliverable describes the third 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 (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 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.

Although we already use NAF to harmonize the different results from the different linguistic modules, cross-lingual event detection additionally requires to perform all these tasks in a semantically compatible way. Therefore, the NewsReader project has developed a cross-lingual pipeline for interpreting events and event components in text in a common language independent semantic representation. In order to achieve cross-lingual semantic interoperability, entities, event mentions and time expressions are projected to language independent knowledge representations. Thus, named entities are linked to English DBpedia (Lehmann et al., 2015) entity identifiers thanks to DBpedia cross-lingual links while nominal and verbal event mentions are aligned to abstract event representations through the Predicate Matrix (L´opez de Lacalle et al., 2014). Additionally, concepts for open class words are represented using the EuroWordNet Inter-lingual index (Vossen, 1998). Finally, time expressions are normalized following the ISO 24617-1 standard (Pustejovsky et al., 2010). Several demonstrators exhibit the capability of the NewsReader cross-lingual event extraction pipeline, which is, to the best of our knowledge, the first of its kind.

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 are also presented.

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-
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 provides the evaluation results and Section 8 an overview of scaling efforts and data
processing throughout the project. 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 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, lemmatiza-
tion, 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, Free-
base, etc.) and entity identifiers. Furthermore, event detection involves the identification
of event mentions, event participants, the temporal constraints and, if relevant, the loca-
tion. It also implies the detection of expressions of factuality of event mentions and the
authority of the source of each event mention. Moreover, NewsReader has developed:

• 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 third cycle of the NewsReader project (Event Detection, Version 3), we
focused on improving the English, Dutch, Italian and Spanish pipelines. All the pipelines
have been evaluated by the gold-standards generated in WP03. The evaluation results are
presented in Section 7.

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 uses FrameNet for semantic role labelling (SRL), while Mate-tools uses PropBank for the same task. As a backup solution, we process the text with Word Sense Disambiguation modules to match WordNet 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 aligned 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 a first version of the Predicate Matrix for verbs and nouns into the English pipeline.
- including a first version of the Predicate Matrix for verbs into the Spanish and Dutch pipelines.

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

- including a second version of the Predicate Matrix for verbs 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 the third cycle of the project we have worked on this problem by:

- including a new version of the Predicate Matrix that covers more mappings with the Event and Situation Ontology (ESO) and some manual mapping between FrameNet and WordNet.
- including the wikification module based on DBpedia Spotlight which identifies not only named entities but other types of terms. It also returns English identifiers into the Spanish, Dutch and Italian pipelines.
- including a time normalization module which follows the ISO 24617-1 standard.

In that way, the English, Spanish, Dutch and Italian NLP processing chains provide harmonized cross-lingual identifiers for predicates, entities and time expressions.
2.1 Cross-lingual Interoperability

Cross-lingual semantic interoperability is achieved because entities, event mentions and time expressions are projected to language independent knowledge representations.

2.1.1 Cross-lingual Predicate Models

The event representation provided by a SRL system depends on the semantic resource used for training that system. Each knowledge source of predicate information contains different descriptions of the roles for each predicate. For this reason, the output of systems based on different resources is not interoperable. Thus, in order to obtain interoperable annotations we need to come to the compatible event representations. Our pipelines guarantee this interoperability level using the Predicate Matrix ([De Lacalle et al., 2014b]) on the output of the SRL modules. The Predicate Matrix gathers multilingual knowledge bases that contain predicate and semantic role information. These resources are connected automatically through a wide set of mappings. This way, it is possible to know which lexical-semantic units refer to the same events or roles. The knowledge sources integrated into the Predicate Matrix for English are WordNet ([Leacock and Chodorow, 1998]), VerbNet ([Kipper, 2005]), FrameNet ([Baker et al., 1997]), PropBank ([Palmer et al., 2005]), NomBank ([Meyers et al., 2004]), ESO ([Segers et al., 2015]) and some additional lexical knowledge coming from the Multilingual Central Repository ([Gonzalez-Agirre et al., 2012]) like SUMO ([Niles and Pease, 2001]), Top Ontology ([Alvez et al., 2008]) or WordNet domain ([Bentivogli et al., 2004]).

As presented in [De Lacalle et al., 2014a], the first version of the Predicate Matrix arises from the union of SemLink ([Palmer, 2009]) and the set of mappings obtained by our automatic methods. The integration of predicate information is performed at two levels: lexical and role levels.

All the mappings obtained at the lexical level are based on graph-based Word Sense Disambiguation (WSD) algorithms. More specifically, the lexical mappings from WordNet to FrameNet and also to VerbNet are obtained by applying WSD algorithms to semantically coherent groupings of verbal entries whereas the lexical mappings from WordNet to PropBank are obtained by applying the WSD to a corpus annotated with PropBank predicates. We have not created new mappings between PropBank and VerbNet because PropBank already offers this information and its coverage is quite complete.

---

5 http://wordnet.princeton.edu/
6 http://verbs.colorado.edu/~mpalmer/projects/verbnet.html
7 http://framenet.icsi.berkeley.edu/
8 http://verbs.colorado.edu/~mpalmer/projects/ace.html
9 http://nlp.cs.nyu.edu/meyers/NomBank.html
10 http://www.newsreader-project.eu/results/event-and-situation-ontology/
11 http://adimen.siemensa.es/web/MCR
12 http://www.ontologyportal.com/
13 http://adimen.siemensa.es/web/WordNet2TO
14 http://wndomains.fbk.eu/
Just like for lexical mappings, PropBank offers quite complete role mappings between PropBank and VerbNet. We therefore concentrate our efforts on finding new role mappings between FrameNet and VerbNet and between FrameNet and PropBank. The mappings between FrameNet frame-elements and VerbNet thematic roles are obtained following the three-steps methodology described in detail in De Lacalle et al., 2014b and outlined in NewsReader Deliverable 4.2.2 (Agerri et al., 2015). An alternative corpus-based approach is used to automatically create new role mappings between FrameNet and PropBank. This method obtains mappings between predicates and roles at the same time.

Tables 2 and 3 show the differences between SemLink and the Predicate Matrix in terms of mappings between lexicons (Table 2) and roles (Table 3). Thanks to the three-step methodology combined with the corpus-based method for FrameNet and PropBank that we used for creating automatic mappings between lexical entries and roles, the resource obtained is much larger than SemLink.

However, the methods presented above only cover English predicates in their verbal forms. We can easily extend the Predicate Matrix taking advance of some resources that are already linked to any of the resources included in the Predicate Matrix. In the second part of this section, we describe the methodology we have followed to obtain a multilingual Predicate Matrix that includes nominal predicates. For this purpose, we have made use of the mappings existing between the following resources:

- English nominal predicates:
  PropBank(PB)-NomBank(NB)

- Spanish verbal predicates:
  PropBank(PB)-SpanishAnCoraVerb(SAV)

- Spanish nominal predicates:
  SpanishAnCoraVerb(SAV)-SpanishAnCoraNom(SAN)

By integrating these resources we have built a new version of the Predicate Matrix, that includes nominalizations and multilingual predicates. As a result of including NB, SAV and
SAN into the *Predicate Matrix*, we have obtained new mappings between these resource and VerbNet (VN), FrameNet (FN) and WordNet (WN). In Tables 4 and 5, we show the number of mappings we obtain after applying the strategy described in this section.

<table>
<thead>
<tr>
<th>Resource</th>
<th>PB</th>
<th>VN</th>
<th>FN</th>
<th>WN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB (English)</td>
<td>2,963</td>
<td>3,923</td>
<td>3,911</td>
<td>7,430</td>
</tr>
<tr>
<td>SAV (Spanish)</td>
<td>6,745</td>
<td>9,092</td>
<td>8,777</td>
<td>15,310</td>
</tr>
<tr>
<td>SAN (Spanish)</td>
<td>4,469</td>
<td>6,190</td>
<td>6,157</td>
<td>10,747</td>
</tr>
</tbody>
</table>

Table 4: Number of lexicon Mappings in PMv1.3. WN: WordNet; FN: FrameNet; VN: VerbNet; PB: PropBank.

<table>
<thead>
<tr>
<th>Resource</th>
<th>PB</th>
<th>VN</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB (English)</td>
<td>7,699</td>
<td>9,699</td>
<td>10,351</td>
</tr>
<tr>
<td>SAV (Spanish)</td>
<td>17,152</td>
<td>19,173</td>
<td>20,296</td>
</tr>
<tr>
<td>SAN (Spanish)</td>
<td>11,752</td>
<td>13,177</td>
<td>14,439</td>
</tr>
</tbody>
</table>

Table 5: Number of role Mappings in PMv1.3. WN: WordNet; FN: FrameNet; VN: VerbNet; PB: PropBank.

The resulting resource, *Predicate Matrix* v1.3, 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 can be seen in the examples included in Table 6.

Using the Predicate Matrix, we can provide correspondences to the rest of the included resources for each predicate and role detected by a SRL module. Thus, the semantic interoperability offered by the Predicate Matrix allows us to translate the output of a SRL analysis to a representation based on any resource connected to the Predicate Matrix like PropBank, FrameNet, VerbNet, SUMO or ESO.

### 2.1.2 Dutch Predicate Matrix

For Dutch, we followed a different strategy to obtain a *Predicate Matrix*. On the basis of the equivalence relations in the Open Dutch Wordnet ([Vossen et al., 2013](#)), we directly translated the English WordNet synsets to the Dutch synsets and from Dutch synsets to each each of the lexical units or synonyms in the synsets. We used different equivalence relations:

1. eq.synonym: there is a mapping (5,247 synsets)
2. there is no direct mapping but a hypernym of the synset is mapped to a bEnglish synsets (21,768 synsets)
Table 6: Some lines for the arguments of the English predicate “sell.01” and the Spanish predicate “vender.1.default” in the Predicate Matrix.

We applied the relations above in the listed order so that synsets are mapped through the best relation available only. If synsets have no mapping, we used the hierarchy in the Dutch WordNet to find the most specific hypernym with a mapping. We used the hypernym relations within one part-of-speech but also cross-part-of-speech hypernym relations and near synonym relations to get more coverage. In Table 7, we show an example for each type of mapping.
Table 7: Examples of Dutch Predicate Matrix derived using different relations.

The Dutch *Predicate Matrix* is used to obtain a Semantic Role output for Dutch compatible to the output for English and Spanish.

2.1.3 Interoperability of Entities

In the second and third year of the project, we did not restrict our focus to cross-lingual interoperability of predicate information. We now also provide cross-lingual interoperability of 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[^15] (Darber et al., 2013a) 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[^16]. 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[^17].

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. 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.

We therefore updated the NED module in the second cycle of the project (see Sections 3.9 and 6.8) so that the module suggests 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:

[^15]: https://github.com/dbpedia-spotlight
[^17]: http://wiki.dbpedia.org/Datasets

---

NewsReader: ICT-316404

February 1, 2016
The NED module decides that Ford refers most likely to the 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 to link the crosslingual realizations of entities in different languages when the entity detection or spotting is performed by the NERC module.

In the last cycle of the project, we detected other concepts that are also relevant and they are not named entities. In order to detect this type of concepts, as a first approach, we have exploited DBpedia Spotlight to obtain more complete event descriptions. Thus, we have implemented a wikification module (see Section 3.11) which uses DBpedia Spotlight for spotting (recognition of phrases) and disambiguation. Given the sentence “I think we’re responding emotionally to the 20th anniversary of the October 1987 stock market crash.”, the wikification module links the token group “October 1987 stock market crash” to the DBpedia entity [http://dbpedia.org/resource/Black_Monday_(1987)] as follows:

```xml
<!−−October 1987 stock market crash−−>
<mark id="m67" lemma="October 1987 stock market crash" source="DBpedia">
  <span>
    <target id="w350" />
    <target id="w351" />
    <target id="w352" />
    <target id="w353" />
    <target id="w354" />
  </span>
  <externalReferences>
    <externalRef resource="spotlight" reference="http://dbpedia.org/resource/Black_Monday_(1987)" confidence="1.0" />
  </externalReferences>
</mark>
```

Just like we did for languages other than English in the NED module, the wikification module obtains DBpedia entries in the localized and English versions.

### 2.1.4 Normalization of time expressions

Our framework and system obtains interoperable representations of the interpretation of events, the entities that play a role within these events as well as the time expressions associated to the events. Thus, our pipelines also include some NLP modules to perform time expression recognition and normalization. In particular, we normalize time expressions following the ISO 24617-1 standard ([Pustejovsky et al., 2010](#)). For example, if temporal expressions such as next Monday, tomorrow, and yesterday in English or ayer and el próximo lunes in Spanish are referring to the same exact date (let’s say November 16th, 2015), all these temporal expressions are normalized to the same TIMEX3 value corresponding to 2015-11-16.
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 3.20. 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, Galician, Italian and Spanish. Ixa-pipe-tok outputs tokenized and segmented texts in NAF, Oneline and CoNLL formats. It also provides normalization functions to comply with annotation in corpora such as Penn Treebank for English and Ancora Corpus for Spanish, among others. The module is part of the IXA pipes ([Agerri et al., 2014](http://ixa2.si.ehu.es/ixa-pipes/)) a modular set of Natural Language Processing tools (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

---

[^18]: [http://ixa2.si.ehu.es/ixa-pipes/]
• **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](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, 2002a]. 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[^19] under their distribution licenses;  

b) **Morfologik-stemming**[^20] 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;  

c) **WordNet Lookup**: We also provide lemmatization based on lookup in WordNet-3.0 ([Fellbaum, 1998]) via the JWNL API[^21]. 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

[^20]: [https://github.com/morfologik/morfologik-stemming](https://github.com/morfologik/morfologik-stemming)
- **Resources**: POS model; Lemmatizer dictionaries: plain text dictionary and morphologik-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](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 a method of headword finders based on Collins’s head rules as defined in his PhD thesis ([Collins, 1999]). It is 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.10).

- **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 can also output the parse trees into Penn Treebank bracketing style.

- **github address**: [https://github.com/newsreader/ixa-pipes](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.11.

- **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

- **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](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,

---

22The module use additional resources to perform the semantic role labeling.
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 the TempEval3 test corpus) compared to other systems such as HeidelTime ([Strötgen et al., 2013]). TimePro annotates the begin point and end point of a duration when deductible from the text. It also creates non text-consuming time expression to describe a duration when the begin point and end point are expressed in the text. 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.15).

- **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
- **Dependencies:** Java, NAF Java library, scala Java library, timenorm [Bethard, 2013], YamCha TinySVM
- **Flexible in- and output:** can provide NAF and TextPro format (i.e. column format)
- **github address:** https://bitbucket.org/qwaider/textpro-en

### 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 and 2003 tasks which focused on

---

24 [http://chasen.org/~taku/software/TinySVM/](http://chasen.org/~taku/software/TinySVM/)
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. The *ixa-pipe-nerc* system learns supervised models via the Perceptron algorithm as described by [Collins, 2002b]. To avoid duplication of efforts, *ixa-pipe-nerc* uses the Apache OpenNLP project implementation of the Perceptron algorithm\(^\text{27}\) customized with its own features. Specifically, *ixa-pipe-nerc* implements basic non-linguistic local features and on top of those a combination of word class representation features partially inspired by [Turian *et al.*, 2010]. The word representation features use large amounts of unlabeled data. The result is a quite simple but competitive system. 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
- **Resources:** models
- **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](https://github.com/newsreader/ixa-pipes)

### 3.7 Word Sense Disambiguation I

- **Module:** it\_makes\_sense\_WSD

- **Description of the module:** The it\_makes\_sense\_WSD is a wrapper around the state-of-the-art WSD system It Makes Sense ([Zhong and Ng, 2010], IMS). Our wrapper allows the use of the KAF or NAF format as input and output. The IMS is currently one of the systems reaching a higher performance. It is based on Supervised Machine Learning, where a Support Vector Machines is applied to induce models from

sense annotated corpora (SemCor and a parallel corpus). The features selected include basic features (tokens, lemmas, POS tags) and syntactic features. IMS implements a flexible framework that allows to integrate different preprocessing tools, additional features and different classifiers. An example of the modules output can be found in Appendix A.4.

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF terms layer
- **Output**: Synsets or Lexical Leys
- **Output representation**: NAF terms layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes
- **Resources**: English WordNet
- **Dependencies**: IMS system
- **Flexible in- and output**: The wrapper is explicitly designed to deal with NAF and KAF. The IMS-module itself has a modular structure and allows users to integrate different preprocessing tools.

**github address**: [https://github.com/rubenIzquierdo/it_makes_sense_WSD](https://github.com/rubenIzquierdo/it_makes_sense_WSD)

### 3.8 Word Sense Disambiguation II

- **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 ([Navigli et al., 2007], S07CG). 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:** This module only works with NAF and KAF.

• **github address:** [https://github.com/ixa-ehu/ukb](https://github.com/ixa-ehu/ukb)

### 3.9 Named Entity Disambiguation I

• **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

• **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**: This wrapper is specifically designed for NAF and KAF. The original implementation offers various kinds of output (e.g. xml, json)

• **github address**: https://github.com/newsreader/ned-spotlight

### 3.10 Named Entity Disambiguation II

• **Module**: POCUS

• **Description of the module**: This module performs domain adaptation of named entities, mainly focusing on disambiguation. It uses information from within the article, background knowledge and from the other linguistic modules (POS). The first two are especially important for domain adaptation, in terms of topic and genre. It relies on entity-to-lemmas list from previous processing of the automotive domain and accesses DBpedia. To ensure our development and evaluation sets are representative, we retrieved 38 previously processed articles. On the development set, our domain model achieves improvement of something around 2% on the NED task in terms of both precision and recall. On the evaluation set, there is also a minimal improvement on the entity disambiguation. On MEANTIME, we get also an improvement of around 2% per corpus, on 3 out of 4 corpora. For the sake of the disambiguation approach, sometimes we extend the entity mention and make an attempt to link the maximum length entity. The output of this module when applied after ixa-pipe-ned is illustrated in Appendix A.7.

• **Input**: Entities layer (processed by ixa-pipe-ned), pos tags and sentences

• **Input representation**: NAF entities layer

• **Output**: Disambiguated named entities

---

29 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.
• **Output representation**: NAF entities layer

• **Required modules**: NED module

• **Level of operation**: sentence level

• **Language dependent**: yes

• **Resources**: DBpedia server (local or remote)

• **Dependencies**: dbpediaEnquirer, KafNafParserPy

• **Flexible in- and output**: This module is designed for the NWR pipeline and only works with NAF and KAF

• **github address**: [https://github.com/filipdbrsk/NWRDomainModel](https://github.com/filipdbrsk/NWRDomainModel)

### 3.11 Wikification

• **Module**: ixa-pipe-wikify

• **Description of the module**: This module performs wikification 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 wf and term elements) and performs wikification (spotting and disambiguation of relevant terms). The module accepts a NAF document containing wf and term elements as input, performs Wikification for your language of choice, and outputs outputs a NAF document with references to DBpedia on markables element. The module offers the “disambiguate” and “candidates” service endpoints. The disambiguate service takes the text input and it returns one disambiguation for each term. The output of this module is illustrated in Appendix A.9.

• **Input**: terms

• **Input representation**: NAF terms and wfs layers.

• **Output**: markables

• **Output representation**: NAF markables layer

• **Required modules**: ixa-pipe-pos 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**: The module itself only works with NAF/KAF. The DBpedia spotlight component offers various kinds of in- and output.

• **github address**: [https://github.com/ixa-ehu/ixa-pipe-wikify](https://github.com/ixa-ehu/ixa-pipe-wikify)

### 3.12 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](http://example.com)). 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](http://example.com); [Lee et al., 2011](http://example.com)), 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](http://example.com)), 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](http://example.com)). 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.14 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

### 3.13 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. Originally 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 correspondences 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.11.

- **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](https://github.com/newsreader/ixa-pipe-srl)

3.14 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 module also selects the top-ranked senses (so-called dominant-senses) of all the mentions of the predicates according to the WSD scores. If the coreference set is based on a similarity match, it additionally stores the lowest-common-subsumer for the match. Source events and grammatical events are excluded from coreference. Whether or not an event is a source event or grammatical event is determined by a list FrameNet frames that can be given as a parameter for source events and grammatical events. Once a coreference sets is determined, the hypernym synsets of the dominant-senses are also added as external references. The external references can thus contain the dominant-senses with the cumulated WSD score, the lowest-comon-subsumers if there was a similarity match with the similarity score and the hypernyms of the dominant-senses. An example of the exact output can be found in Appendix A.14.

• **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](https://github.com/cltl/EventCoreference)
3.15 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 the TempEval3 test corpus is: 58.8% precision, 58.2% recall and 58.5% F1-measure.

The module also annotates relations between predicates and time anchors. It uses the temporal relations, the predicate arguments of type TMP, the dependencies and some manually defined patterns. If a predicate refers to a punctual event then the time when it occurred is considered as its time anchor. If it refers to a durative event then the time when it starts and ends are expressed through begin point and end point attributes.

This module is part of TextPro pipeline, a multilingual NLP pipeline developed at FBK. Appendix [A.16](#) provides an example of the module’s output.

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

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

- **Output**: tlinks, predicateAnchor relations

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

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

- **Level of operation**: document level

- **Language dependent**: yes

- **Resources**: language model, language dependent rules
• **Dependencies**: Java, Python, NAF Java library, Explicit Discourse Connectives Tagger ([Pitler and Nenkova, 2009](http://hlt-services2.fbk.eu/textpro/?p=91)), MorphoPro[^30], YamCha[^31], TinySVM[^32]

• **Flexible in- and output**: this module works with NAF/KAF as well as the TextPro format (i.e. column format)

• **github address**: [https://github.com/paramitamirza/TempCauseRelPro](https://github.com/paramitamirza/TempCauseRelPro)

### 3.16 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](http://chasen.org/~taku/software/yamcha/)). 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.18.

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

• **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-timepro, fbk-temprel

• **Level of operation**: document level

• **Language dependent**: yes

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

[^32]: [http://chasen.org/~taku/software/TinySVM/](http://chasen.org/~taku/software/TinySVM/)
• **Dependencies**: Java, Python, NAF Java library, Explicit Discourse Connectives Tagger ([Pitler and Nenkova, 2009](http://hlt-services2.fbk.eu/textpro/?p=91)), MorphoPro[33] YamCha[34] TinySVM[35]

• **Flexible in- and output**: This module can work with KAF/NAF as well as the TextPro format (i.e. column format).

• **github address**: [https://github.com/paramitamirza/TempCauseRelPro](https://github.com/paramitamirza/TempCauseRelPro)

### 3.17 Factuality

• **Module**: VUA-perspective-factuality

• **Description of the module**: The perspective-factuality module aims to identify whether an event is certain, probable or possible, whether it is confirmed or denied, or whether it takes place in the future or not. The core of the module is trained on the most embedded layer of factuality values from FactBank v1.0[36]. The current implementation uses TiMBL ([Daelemans and van den Bosch, 2005](http://chasen.org/~taku/software/TiMBL/))’s K-nearest neighbor’s algorithm for classification. Features include lemma, token, POS, morphological properties, dependency relation with head, and lexical information on the target word, as well as the lemma, word, POS, morphological properties and dependency relation with head of words in a three-word context label. In addition, the target word’s head in the dependency tree, and information of direct dependent and dependency chains are taken into consideration. FactBank provides values that can be directly mapped to the certainty and polarity values used in NewsReader. A basic rule based approach determines whether the event is situated in the future or not based on verbal morphology. The representation of factuality in NAF was corrected to be conform to the general requirements for NAF. An example of the factuality layer can be found in Appendix [A.22](http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T23). The first version of the module was released in July 2015. Experiments with more elaborate features and other Machine Learning are currently under development.

• **Input**: Tokens, POS-tags, morphological analysis, dependencies, semantic roles and WSD, coreference with events

• **Input representation**: NAF token-layer, term-layer, dependency-layer, srl-layer, coreference layer

• **Output**: Factuality values indicating certainty, polarity and tense for a span of terms

• **Output representation**: NAF factuality layer
• **Required modules**: tokenizer, POS-tagger, dependency-parser, SRL labeller, ontotagger, WSD

• **Level of operation**: sentence level

• **Language dependent**: yes

• **Resources**: FactBank

• **Dependencies**: Perl, Python, TiMBL, KafNafParserPy

• **Flexible in- and output**: The module consist of a CoNLL based core for the FactBank values and a NAF wrapper.

• **github address**: https://github.com/newsreader/vua_factuality

### 3.18 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 was developed originally during the OpeNER project and improved during NewsReader, 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 OpeNER 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.23.

• **Input**: NAF text processed through the pipeline, this module will use the information provided by all the rest

---

[^37]: http://www.opener-project.eu/
• **Input representation:** token, term, entity, dependency and constituency NAF layers

• **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 on the OpeNER project

• **Dependencies:** CRF++ library, SVM-Light library and KafNafParser

• **Flexible in- and output:** This module only works with NAF and KAF.

• **github address:** [https://github.com/cltl/opinion_miner_deluxe](https://github.com/cltl/opinion_miner_deluxe)

### 3.19 Text Classification

• **Module:** ixa-pipe-topic

• **Description of the module:** The module is based on the Multilingual Eurovoc thesaurus descriptors and it makes use of the JRC Eurovoc Indexer JEX ([Steinberger et al., 2012](https://taku910.github.io/crfpp)). 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 module works for the four languages of the project. The output representation is illustrated in Appendix A.24.

• **Input:** text

• **Input representation:** NAF raw 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**: This module only works with KAF/NAF.
• **github address**: https://github.com/newsreader/

### 3.20 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 order of the linguistic processors is displayed in Figure 1. Note that the current modules inherit the dependencies of previous modules. Then, the dependencies among each task (module) are presented in Figure 2.

![Figure 1: NewsReader pipeline. Linguistic processors’ order](image)

---

**NewsReader: ICT-316404**

**February 1, 2016**
Figure 2: NewsReader pipeline. Dependencies among modules
4 Dutch NLP Processing

The description of modules for the Dutch pipeline are structured similarly 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 4.15.

4.1 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, Galician, Italian and Spanish. Ixa-pipe-tok outputs tokenized and segmented texts in NAF, Oneline and CoNLL formats. It also provides normalization functions to comply with annotation in corpora such as Penn Treebank for English and Ancora Corpus for Spanish, among others. The module is part of the IXA pipes (Agerri et al., 2014) a modular set of Natural Language Processing tools (or pipes) which provide easy access to NLP technology for various languages. 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

\[http://ixa2.si.ehu.es/ixa-pipes/\]
• **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](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 analysis of Dutch text. It is based on the Alpino parser [42] 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 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.13.

• **Input**: Tokens

• **Input representation**: NAF text layer

• **Output**: Lemmas and POS-tags, constituent 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/](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](https://github.com/cltl/morphosyntactic_parser_nl)
- Alpino (git address): [git://urd.let.rug.nl/Alpino.git](git://urd.let.rug.nl/Alpino.git)

### 4.3 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\(^\text{43}\) and 2003\(^\text{44}\) 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. The *ixa-pipe-nerc* system learns supervised models via the Perceptron algorithm as described by [Collins, 2002b]. To avoid duplication of efforts, *ixa-pipe-nerc* uses the Apache OpenNLP project implementation of the Perceptron algorithm\(^\text{45}\) customized with its own features. Specifically, *ixa-pipe-nerc* implements basic non-linguistic local features and on top of those a combination of word class representation features partially inspired by [Turian et al., 2010]. The word representation features use large amounts of unlabeled data. The result is a quite simple but competitive system. 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.

---

\(^{43}\)http://www.clips.ua.ac.be/conll2002/ner/

\(^{44}\)http://www.clips.ua.ac.be/conll2003/ner/

\(^{45}\)http://opennlp.apache.org/
4.4 Word Sense Disambiguation

- **Module**: vua-wsd

- **Description of the module**: VUA-WSD is a machine learning system that performs Word Sense Disambiguation for Dutch text. It is based on a supervised machine learning approach: Support Vector Machines. The library selected for the low level machine learning engine is lib-svm.\(^{46}\) A bag-of-words feature model is used for representing the training instances, where tokens and lemmas are considered to build the context for each target word. The training material used has been the corpus manually annotated on the DutchSemCor project.\(^{47}\) One classifier is build for each target lemma, following the one-vs-all approach to deal with the multilabel classification of the 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. The type of references created can be specified: Cornetto lexical unit identifiers, OpenDutchWordNet lexical unit ids or OpenDutchWordNet synset ids.

- **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](https://github.com/cltl/svm_wsd)

\(^{46}\) [https://github.com/cjlin1/libsvm](https://github.com/cjlin1/libsvm)

\(^{47}\) [http://www2.let.vu.nl/oz/cltl/dutchsemcor](http://www2.let.vu.nl/oz/cltl/dutchsemcor)
4.5 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

- **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

- **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](https://github.com/newsreader/ned-spotlight)

4.6 DBpedia NER

- **Module**: dbpedia_ner

- **Description of the module**: Named Entity Recognizer (NER) and entity linker to DBpedia entries based on Dbpedia spotlight for KAF/NAF files. This tool is a wrapper around the original DBpedia spotlight tool ([Daiber et al., 2013b](#)) that allows the use of KAF/NAF as input and output. The original DBpedia spotlight Rest service can be used online or via local installation, which provides a more reliable service.

---

48 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.
Our wrapper takes a KAF/NAF input file containing the token and term layer (the term layer is only required because it is used to establish the internal links), calls to the DBpedia spotlight service and creates the whole entity layer, with links to DBpedia and DBpedia ontology types. This is the main difference with the ixa-pipe-ned module. This module takes an already existing entity layer, and links these entities with DBpedia resources, using DBpedia spotlight. If a certain entity is not detected by the NER recognizer, it will not be linked to DBpedia even if there is an existing entry. For certain tasks, mainly in specific domains (such as biomedicine or maths), there are many domain terminology that are not properly Named Entities, but linking them to DBpedia would provide a very useful and rich background information. This was the main reason that motivated the implementation of this module. This module can either be used in addition to the ixa-pipe-nerc and ixa-pipe-ned modules or it can be used instead of them. It achieves high recall at cost of precision and is currently not standardly included in the pipeline. An illustration of the output is provided in Appendix A.8.

- **Input**: Lemmatized and POS tagged text
- **Input representation**: NAF token and term layers
- **Output**: Entity layer with new entities and external references to DBpedia
- **Output representation**: NAF entity layer
- **Required modules**: Tokenizer and POS tagger modules
- **Level of operation**: sentence level
- **Language dependent**: yes (there are prebuilt models for many languages)
- **Resources**: DBpedia spotlight server
- **Dependencies**: KafNafParserPy
- **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/rubenIzquierdo/dbpedia_ner

### 4.7 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 ([Strötzgen and Gertz, 2013](#)), a temporal tagger supporting English, German, Dutch, Vietnamese,
Arabic, Spanish, Italian, French, Chinese and Russian. 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. Heideltime for Dutch uses the NAF wrapper built around the UIMA architecture that is also used for Spanish, which was recently developed. Two adaptations were necessary to make sure Heideltime can work correctly with the NAF output currently provided by the tokenizer and Alpino in the Dutch pipeline. First, the POS-tags and morphological features provided by Alpino need to be mapped to those provided by TreeTagger. In principle, the implementation could also handle output from another morphological analyzer, if the correct set of mappings is provided. Second, the tokenizer analyzes expressions such as 2001-2005 as one token, where Heideltime identifies two years. Additional rules were added to handle these cases.

The output of time expressions for Dutch is modeled on the English representations (see Appendix A.15).

- **Input**: Tokens, terms with POS-tags and morphofeat from Alpino
- **Input representation**: NAF token layer and NAF term layer
- **Output**: Normalized time expressions in timex
- **Output representation**: NAF time expression layer
- **Required modules**: Tokenizer producing NAF or KAF, Alpino with NAF wrapper
- **Level of operation**: Document level
- **Language dependent**: yes (language specific rules and patterns)
- **Resources**: language specific rules
- **Dependencies**: Java, Maven, NAF Java Library
- **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/ixa-ehu/ixa-pipe-time

---

HeidelTime source code is available here: https://code.google.com/p/heideltimem
4.8 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

- **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](https://git@github.com:cltl/OntoTagger)

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 [De 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. The trained model is provided with the module, the raw SoNaR data can be obtained through the TsT centrale.

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.11.

- **Input**: Lemmatized and POS tagged text and syntactic dependencies
- **Input representation**: NAF terms, constituents and dependencies layers
- **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](https://github.com/newsreader/vua-srl-nl)

4.10 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.14.

---

50 A list of files used can be found in the module’s GitHub archive
51 [http://tst-centrale.org/nl/producten/corpora/sonar-corpus/6-85](http://tst-centrale.org/nl/producten/corpora/sonar-corpus/6-85)
• **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

• **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](https://github.com/cltl/EventCoreference)

### 4.11 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 with the sense scores 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.12.

• **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

### 4.12 Nominal events

• **Module**: vua-nominal-event-detection

• **Description of the module**: This module, is a function with the OntoTagger package (eu.kyotoproject.main.NominalEventCoreference), that uses a FrameNet lexical unit to frame data file. It reads the term layer and checks every noun against this lexical unit index to obtain the Frames. If it finds any Frames, these are added to the term as external References and the term is selected for creating a predicate element in the SRL. If there is no match in the FrameNet lexical unit file, it checks of the term has been tagged with PredicateMatrix references. If it was mapped to an ESO class or FrameNet class it is added to the selection. Else if there it is tagged as Dynamic it is added as well. Compound terms are processed to obtain the head component which is checked in the same way as above. All terms with a event matched are used to create predicate elements in the SRL with all the relevant information. The nominal predicates are used by the Nominal Predicate SRL to obtain the roles linked to the predicated, see the vua-srl-dutch-nominal-events module. We translated the English FrameNet file with lexical unit to FrameNet mappings to Dutch as a resource for this program. The output is integrated in the SRL layer presented in Appendix A.11.

• **Input**: Lemmatized and POS tagged text, preferably extended with external references through the PredicateMatrix tagger

• **Input representation**: NAF token and term layers

• **Output**: Predicates are added to the SRL layer.

• **Output representation**: NAF SRL layer with predicate but without the roles

• **Required modules**: Tokenizer and POS tagger, WSD module, PredicateMatrix tagger

• **Level of operation**: term layer
- **Language dependent**: no.
- **Resources**: FrameNet lexical unit to frame file
- **Dependencies**: KafSaxParser
- **Flexible in- and output**: Only handles NAF input/output streams
- **github address**: [https://github.com/cltl/OntoTagger.git](https://github.com/cltl/OntoTagger.git)

### 4.13 Nominal Predicate SRL
- **Module**: vua-srl-dutch-nominal-events
- **Description of the module**: This is a basic program that checks whether nominal predicates have PP modifiers. For each PP modifier, we verify whether the modifier is headed by the preposition *van*, in which case we create an *Arg1* role. All other prepositions receive the role *ArgM* for an underspecified modifier. Version 2 improves on version 1 in that it can detect multiple PPs for the same nominal events, performs a minor interpretation check and output propBank relations which can be picked up by NAF to SEM software. The output is integrated in the SRL layer presented in Appendix [A.11](#).
- **Input**: Lemmatized and POS tagged text, syntactic dependencies and predicates from the srl layer
- **Input representation**: NAF terms and dependencies and srl layers
- **Output**: Roles are added to nominal predicates in the srl layer
- **Output representation**: Predicate roles in the srl layer
- **Required modules**: Tokenizer, POS tagger, Alpino parsing module, Nominal events
- **Level of operation**: sentence level
- **Language dependent**: partially (checking for a specific Dutch preposition)
- **Resources**: None
- **Dependencies**: Python, KafNafParserPy
- **Flexible in- and output**: only works on NAF and KAF
- **github address**: [https://github.com/newsreader/vua-srl-dutch-nominal-events/tree/master](https://github.com/newsreader/vua-srl-dutch-nominal-events/tree/master)
4.14 Opinions

- **Module**: opinion-miner

- **Description of the module**: This module 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). It has been developed as part of the OpeNER project[^52] 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 for Dutch is the same as for English (Appendix A.23).

- **Input**: NAF text processed through 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

- **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

[^52]: [http://www.opener-project.eu/](http://www.opener-project.eu/)
• **Dependencies**: CRF library\(^{53}\) SVM-Light library, KafNafParser\(^{54}\) and VUA\(_{pylib}\) library\(^{55}\)

• **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](https://github.com/cltl/opinion_miner_deluxe)

### 4.15 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

---

\(^{53}\)[http://www.chokkan.org/software/crfsuite/]

\(^{54}\)[https://github.com/cltl/KafNafParserPy]

\(^{55}\)[https://github.com/cltl/VUA\(_{pylib}\)]
5 Italian NLP Processing

The NLP processing suite for Italian is based on TextPro ([Pianta et al., 2008](http://example.com)). The suite has been designed so that it can integrate and reuse existing state of the art NLP components developed by researchers at FBK (HLT 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 modules use a column format (similar to CoNLL tabular format) of the input/output as interchange format. The modules for time expression detection and normalization, 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. The Italian modules of TextPro are available on [https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0) under a GNU GPL v3 license.

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 does not 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.25), 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 sentence splitting

- **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://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0)

### 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.26.

• **Input**: Tokens

• **Input representation**: Tokennorm column

• **Output**: All possible morphological analyses for each token

• **Output representation**: full_morpho column

• **Required modules**: fbk-tokenpro

• **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://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/MorphoPro](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/MorphoPro)

### 5.3 POS tagging

• **Module**: fbk-tagpro
• **Description of the module**: TagPro is a module for POS-tagging based on Conditional Random Field. The Italian model is trained on a corpus composed by Wikipedia, Wikinews and Wikisource articles, automatically annotated with the previous version of TagPro (Pianta and Zanoli, 2007). This update has been done in order to be able to freely distribute the module. The first version of TagPro was the best system at EVALITA 2007 (98.04% F1). The output of this module is illustrated in Appendix A.27.

• **Input**: Token, morphological analysis

• **Input representation**: Token and full_morpho column

• **Output**: POS

• **Output representation**: POS column

• **Required modules**: fbk-tokenpro, fbk-morphopro

• **Level of operation**: sentence level

• **Language dependent**: yes

• **Resources**: language model

• ** Dependencies**: Java, CRFsuite

• **Flexible in- and output**: The module only works on the column format (as input and output).

• github address: [https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/TagPro](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/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.28 illustrates the output of the module. This output, together with the output of the POS tagger, is later converted to the NAF terms 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**: fbk-tokenpro, fbk-morphopro, fbk-tagpro

• **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://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/LemmaPro](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/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](https://www.aclweb.org/anthology/P06-1014)). 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.29](https://example.com/appendix29) later converted to a NAF representation like the one illustrated in [A.6](https://example.com/appendix6).

• **Input**: Token, lemma, POS

• **Input representation**: Token, tokennorm, POS and lemma columns

• **Output**: Named entities

• **Output representation**: Entity column

• **Required modules**: fbk-tokenpro, fbk-morphopro, fbk-lemmapro, fbk-tagpro

• **Level of operation**: sentence level
• **Language dependent**: yes

• **Resources**: language model, proper noun gazetteers

• **Dependencies**: YamCha, TinySVM

• **Flexible in- and output**: The module only uses column format as in- and output.

• **github address**: https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/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 two categories: B-NP (for nominal predicate), B-VX (for verbal predicate). An example of the module’s output is given in Appendix A.30.

• **Input**: Token, POS

• **Input representation**: Token and POS columns

• **Output**: Chunks

• **Output representation**: Chunk column

• **Required modules**: fbk-tokenpro, fbk-tagpro

• **Level of operation**: sentence level

• **Language dependent**: yes

• **Resources**: language model

• **Dependencies**: YamCha, TinySVM

• **Flexible in- and output**: The module only uses the column format as in- and output.

• **github address**: https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/ChunkPro
5.7 Dependency Parser

- **Module**: fbk-depparserpro

- **Description of the module**: This module implements an Italian Dependency Parser. It is based on the Malt Parser ([Lavelli et al., 2013](http://www.maltparser.org/)), 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.31. This output is later converted to a NAF representation like the English representation provided in Appendix A.11.

- **Input**: Token, POS, lemma

- **Input representation**: Token, POS and lemma columns

- **Output**: Dependencies

- **Output representation**: Parserid, feats, headid and deprel columns

- **Required modules**: fbk-tokenpro, fbk-tagpro, fbk-lemmapro

- **Level of operation**: sentence level

- **Language dependent**: language model

- **Resources**: dependency parsing model trained on TUT (Turin University Treebank)

- **Dependencies**: MaltParser [http://www.maltparser.org/](http://www.maltparser.org/)

- **Flexible in- and output**: The module only uses the column format as in- and output.

- **github address**: [https://bitbucket.org/qwaider/textpro-ita/src/master/\TextPro2.0/modules/MaltParser](https://bitbucket.org/qwaider/textpro-ita/src/master/\TextPro2.0/modules/MaltParser)

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](https://sites.google.com/site/eventievalita2014/)) has been adapted 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.15.

- **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**: Java, NAF Java library, scala Java library, timenorm ([Bethard, 2013]), YamCha, TinySVM
- **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).
- **github address**: https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/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.17).

- **Input**: Token, lemma, POS, entity, chunk, time expression, morpho

- **Input representation**: NAF terms, entities, chunks and time expression layers

---

59 [https://sites.google.com/site/eventievalita2014/](https://sites.google.com/site/eventievalita2014/)
• **Output**: Predicates

• **Output representation**: NAF SRL layer

• **Required modules**: Tokenizer, POS-tagger, morphological analyzer, chunker, named entity recognizer, temporal expressions recognizer and normalizer

• **Level of operation**: document level

• **Language dependent**: yes

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

• **Dependencies**: Java, NAF Java library, Snowball Italian stemmer[^60], MultiWordNet domains[^61], derIvaTario lexicon[^62], YamCha, TinySVM

• **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).

• **github address**: [https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/modules/EventPro](https://bitbucket.org/qwaider/textpro-ita/src/master/TextPro2.0/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.16.

• **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

[^60]: [http://snowball.tartarus.org/algorithms/italian/stemmer.html](http://snowball.tartarus.org/algorithms/italian/stemmer.html)
[^62]: [http://derivatario.sns.it/](http://derivatario.sns.it/)
• **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

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

• **Dependencies**: Java, Python, NAF Java library, YamCha, TinySVM

• **Flexible in- and output**: the module can use both NAF and the TextPro format (i.e. column format).

• **github address**: [https://bitbucket.org/qwaider/textpro-ita/src/master\TextPro2.0/modules/TempRelPro](https://bitbucket.org/qwaider/textpro-ita/src/master\TextPro2.0/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”\(^{63}\) annotated with factuality on top of TimeML annotation. The output of this module is illustrated in Appendix A.19. 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**: Python, derIvaTario lexicon\(^{64}\) YamCha, TinySVM


\(^{64}\)[http://derivatario.sns.it/](http://derivatario.sns.it/)
5.12 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 Italian, given a disambiguate entity, it is also possible to return the corresponding English dbpedia-entry. This feature allows the interoperability of Named Entities. 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
5.13 Semantic Role Labelling

- **Module**: fbk-srl

- **Description of the module**: The SRL system for Italian is based on dependency relations (output of the dependency parser module), events (output of the event recognition module) and PropBank-like frames (built automatically using the Multi-SemCor English-Italian aligned corpus ([Bentivogli and Pianta, 2005](#)). In order to disambiguate predicate senses we use MultiWordNet ([Pianta et al., 2002](#)) through the resource provided by [Bond and Paik, 2012](#). Thanks to this resource, predicates have an external reference through an interlingual index (ili). The match is created based on the lemma and morphological features, as well as comparing the roles extracted and those represented in the PropBank-like frames. The output of this module is illustrated in Appendix A.20.

- **Input**: terms, chunks, dependency trees, named entities, time expressions, predicates

- **Input representation**: NAF terms, chunks, entities, timeExpressions, deps, srl layers

- **Output**: Predicate arguments and external references

- **Output representation**: NAF srl layer

- **Required modules**: fbk-lemmapro, fbk-chunkpro, fbk-entitypro, fbk-timepro, fbk-eventpro, fbk-depparserpro

- **Level of operation**: sentence level

- **Language dependent**: yes

- **Resources**: MultiWordnet ([Pianta et al., 2002](#))

- **Dependencies**: Java, NAF Java library

- **Flexible in- and output**: The module only uses NAF as input and output.

- **github address**: [https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-SRL](https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-SRL)

---

[65] Bond and Paik, 2012 have extracted and normalized all the data from wordnets in a variety of languages, and linked them to Princeton WordNet 3.0 ([Fellbaum, 1998](#)).
5.14 Wikification

- **Module**: ixa-pipe-wikify

- **Description of the module**: This module performs wikification 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 wf and term elements) and performs wikification (spotting and disambiguation of relevant terms). The module accepts a NAF document containing wf and term elements as input, performs Wikification for your language of choice, and outputs a NAF document with references to DBpedia on markables element. The module offers the “disambiguate” and “candidates” service endpoints. The disambiguate service takes the text input and it returns one disambiguation for each term. The output of this module is illustrated in Appendix A.9.

- **Input**: terms

- **Input representation**: NAF terms and wfs layers.

- **Output**: markables

- **Output representation**: NAF markables layer

- **Required modules**: fbk-tagpro 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**: The module only uses NAF as input and output.

- **github address**: https://github.com/ixa-ehu/ixa-pipe-wikify

5.15 Event coreference

- **Module**: fbk-eventcoref

- **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 of this module is illustrated in Appendix A.21.
• **Input**: lemma, POS, morphological analysis, SRL

• **Input representation**: NAF terms, SRL layers

• **Output**: Coreference sets for events

• **Output representation**: NAF coref layer

• **Required modules**: fbk-lemmapro, fbk-morphopro, fbk-srl

• **Level of operation**: sentence level

• **Language dependent**: yes

• **Resources**: none

• **Dependencies**: Java, NAF Java library

• **Flexible in- and output**: The module only uses NAF as input and output.

• **github address**: https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-eventCoref

### 5.16 Predicate Time Anchor

• **Module**: fbk-timeanchor

• **Description of the module**: The module annotates relations between predicates and time anchors. It uses the temporal relations, the predicate arguments of type TMP, the dependencies and some manually defined patterns. If a predicate refers to a punctual event then the time when it occurred is considered as its time anchor. If it refers to a durative event then the time when it starts and ends are expressed through begin point and end point attributes. The module produces the same output structure as the temprel English module illustrated in Appendix A.16.

• **Input**: token layer, term layer with lemma, POS, dependency, SRL, event coreference, time expression and temporal relation

• **Input representation**: NAF terms, entities, deps, SRL, coref, timeExpression and temporalRelations layers

• **Output**: predicate time anchor

• **Output representation**: NAF temporalRelations layer

• **Required modules**: fbk-tagpro, fbk-entitypro, fbk-depparserpro, fbk-srl, fbk-timepro, fbk-eventcoreference, fbk-temprel
• **Level of operation**: document level
• **Language dependent**: yes
• **Resources**: language model, language dependent rules
• **Dependencies**: Java, NAF Java library
• **Flexible in- and output**: The module only uses NAF as input and output.

• github address: [https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-timeAnchor](https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-timeAnchor)

### 5.17 Italian Pipeline

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

![Italian pipeline diagram](https://bitbucket.org/qwaider/textpro-ita/src/master/FBK-timeAnchor)

**Figure 4**: NewsReader Italian pipeline. Dependencies among modules
6 Spanish NLP Processing

The NLP pipeline 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 other languages such as Dutch, German, French, Galician, Italian and Spanish. Ixa-pipe-tok outputs tokenized and segmented texts in NAF, Oneline and CoNLL formats. It also provides normalization functions to comply with annotation in corpora such as Penn Treebank for English and Ancora Corpus for Spanish, among others. The module is part of the IXA pipes ([Agerri et al., 2014](http://ixa2.sii.ehu.es/ixa-pipes)) 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

- **Dependencies**: Java, Maven, NAF Java library

- **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](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, 2002a](http://www.cs.cmu.edu/~tml/collins.shtml). The models have 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 Márquez, 2004](http://www.dbis.de/grammars/2004/faculty/minerva.html) 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[67](http://languagetool.org/) under their distribution licenses; b) Morfologik-stemming[68](https://github.com/morfologik/morfologik-stemming). 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

- **Language dependent**: yes

- **Resources**: POS model; Lemmatizer dictionaries: plain text dictionary and morfologik-stemming dictionary.

- **Dependencies**: Java, Maven, NAF Java library, JWNLI API, Apache OpenNLP.


[68]https://github.com/morfologik/morfologik-stemming
• **Flexible in- and output:** It accepts tokenized text in NAF. It outputs NAF or CoNLL formats.

• **github address:** [https://github.com/newsreader/ixa-pipes](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 English have been trained using the Penn treebank. Furthermore, ixa-pipe-parse provides a method of headword finders based on Collins’s head rules as defined in his PhD thesis ([Collins, 1999](#)). It is a modification of Collins’s 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.10.

- **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 can also output the parse trees into Penn Treebank bracketing style.

• **github address:** [https://github.com/newsreader/ixa-pipes](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.11](#).

- **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 mode\(^{69}\)

- **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](#)

6.5 Time Expressions

- **Module**: ixa-heideltimetime

- **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.\(^{70}\) HeidelTime identifies temporal expressions based on language specific patterns. Identified temporal expressions are normalized and represented according to TIMEX annotations ([Sundheim, 1996](#)).

\(^{69}\)The module use additional resources to perform the semantic role labeling.

\(^{70}\)HeidelTime source code is available here: [https://code.google.com/p/heideltime/](#)
It has been integrated in a Java wrapper so that it can take NAF with token and term layer as input. The output is modeled after the output for the other languages as illustrated in Appendix A.15.

- **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:** Java

- **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](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\(^1\) and 2003\(^2\) 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. The *ixa-pipe-nerc* system learns supervised models via the Perceptron algorithm as described by [Collins, 2002b]. To avoid duplication of efforts, *ixa-pipe-nerc* uses the Apache OpenNLP project implementation of the Perceptron algorithm\(^3\) customized with its own features. Specifically, *ixa-pipe-nerc* implements basic non-linguistic local features and on top of those a combination of word class representation features partially inspired by [Turian *et al.*,].

\(^1\)[http://www.clips.ua.ac.be/conll2002/ner/]
\(^2\)[http://www.clips.ua.ac.be/conll2003/ner/]
\(^3\)[http://opennlp.apache.org/]

NewsReader: ICT-316404  February 1, 2016
The word representation features use large amounts of unlabeled data. The result is a quite simple but competitive system.

For Spanish we currently obtain best results using CoNLL 2002 clusters and dictionaries. Our best model obtains 84.30 F1 vs 81.39 F1 of \cite{Carreras et al., 2002a}. 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 (\cite{MÁRQUEZ 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 Wikification
• Module: ixa-pipe-wikify

• Description of the module:
This module performs wikification 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 wf and term elements) and performs wikification (spotting and disambiguation of relevant terms). The module accepts a NAF document containing wf and term elements as input, performs Wikification for your language of choice, and outputs outputs a NAF document with references to DBpedia on markables element. The module offers the “disambiguate” and “candidates” service endpoints. The disambiguate service takes the text input and it returns one disambiguation for each term. The output of this module is illustrated in Appendix A.9.

• Input: terms
• Input representation: NAF terms and wfs layers.
• Output: markables
• Output representation: NAF markables layer
• Required modules: ixa-pipe-pos 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/ixa-ehu/ixa-pipe-wikify
6.10 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. For Spanish, the module has been adapted from English and evaluated on the publicly available datasets distributed by the SemEval 2010 task on Multilingual Coreference resolution obtaining 64.22 F1 vs. 67.50 F1 obtained by the best system in the task (SUCRE). Appendix A.14.

- **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](https://bitbucket.org/Josu/corefgraph)

6.11 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 correspondences 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.11.

- **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](https://github.com/newsreader/ixa-pipe-srl)

### 6.12 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. Appendix A.14 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](https://github.com/cltl/EventCoreference)

### 6.13 Text Classification

• **Module**: ixa-pipe-topic

• **Description of the module**: The module is based on the Multilingual Eurovoc thesaurus descriptors and it makes use of the JRC Eurovoc Indexer JEX ([Steinberger et al., 2012](https://doi.org/10.2788/47462)). 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 module works for the four language of the project and Appendix A.24 provides an example of the output of the module for English. The output for Spanish follows the same structure.

• **Input**: text

• **Input representation**: NAF raw 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/

### 6.14 Spanish Pipeline

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

![NewsReader Spanish pipeline. Dependencies among modules](image)

Figure 5: NewsReader Spanish pipeline. Dependencies among modules
7 Evaluation

This section presents the evaluation results for the pipelines developed in the NewsReader project. We have evaluated the modules of the pipelines using standard datasets and the NewsReader MEANTIME corpus (see the Technical Report D3-3-2 for more details). The NewsReader MEANTIME (Multilingual Event AND TIME) corpus, produced in the NewsReader project as part of WP3, is a semantically annotated corpus of 480 English, Italian, Spanish, and Dutch news articles. The English section of the corpus comes from Wikinews. The Spanish, Italian and Dutch sections are translations of the English articles aligned at the sentence level. It has been annotated manually at multiple levels, including entities, events, temporal information, semantic roles, and intra-document and cross-document event and entity coreference.

Various evaluation scorers (and format converters) have been implemented in order to evaluate the pipelines. They have been gathered in order to form an evaluation package available on github: [https://github.com/newsreader/evaluation](https://github.com/newsreader/evaluation). It includes the evaluation of Named Entity Recognition and Classification, Named Entity Disambiguation, Event detection and coreference, Nominal coreference, Semantic Role Labeling, Temporal Processing and Event Factuality detection.

We first present the evaluation results for each language on its own and in Section 7.5 we present all the evaluation results together.

7.1 English

7.1.1 English Named Entity Recognition and Classification

In NewsReader we use the iza-pipe-nerc system ([Agerri et al., 2014]) off-the-self to train our NERC models; iza-pipe-nerc learns supervised models via the Perceptron algorithm as described by [Collins, 2002b]. To avoid duplication of efforts, iza-pipe-nerc uses the Apache OpenNLP project implementation of the Perceptron algorithm customized with its own features. Specifically, iza-pipe-nerc implements basic non-linguistic local features and on top of those a combination of word class representation features partially inspired by [Turian et al., 2010]. The word representation features use large amounts of unlabeled data. The result is a quite simple but competitive system which obtains the best results for English on the two datasets we have used for the evaluation, both on the CoNLL 2003 dataset [Tjong Kim Sang and De Meulder, 2003] and on MEANTIME.

The local features implemented are: current token and token shape (digits, lowercase, punctuation, etc.) in a 2 range window, previous prediction, beginning of sentence, 4 characters in prefix and suffix, bigrams and trigrams (token and shape). On top of them we induce three types of word representations:

---

74 http://www.newsreader-project.eu/results/data/wikinews/
75 Wikinews is a collection of multilingual online news articles written collaboratively in a wiki-like manner [http://en.wikinews.org/]
76 https://github.com/ixa-ehu/ixa-pipe-nerc
77 http://opennlp.apache.org/
• Brown (Brown et al., 1992) clusters, taking the 4th, 8th, 12th and 20th node in the path. We induced 1,000 clusters on the Reuters RCV1 corpus using the tool implemented by Liang.

• Clark clusters (Clark, 2003), using the standard configuration to induce 600 clusters on the Reuters RCV1 corpus.

• Word2vec clusters (Mikolov et al., 2013), based on K-means applied over the extracted word vectors using the skip-gram algorithm; 200 classes were induced using the English Gigaword corpus (5th edition).

The implementation of the clustering features looks for the cluster class of the incoming token in one or more of the clustering lexicons induced following the three methods listed above. If found, then the class is added as a feature. The Brown clusters only apply to the token related features, which are duplicated. We chose the best combination of features on the CoNLL 2003 development dataset, which corresponds to the configuration we have just described.

First we evaluate our NERC system on the English CoNLL 2003 official testset. This data is a collection of news wire articles from the Reuters Corpus. The data consists of columns separated by a single space. The first item on each line is a word and the last one the named entity tag. An example is shown in Figure 6. In total, 301,418 annotated tokens for dev/train/test datasets are provided. Due to copyright issues only the annotations were made available at CoNLL 2003. In order to build the complete datasets, it is necessary to access the Reuters Corpus, which can be obtained from NIST for research purposes. They also provide an official evaluation script.

For this evaluation, we added the publicly available gazetteers from the Illinois NER Tagger (Ratinov and Roth, 2009) to our system. The results obtained by ixa-pipe-nerc are the best of any publicly available system up to date (Ratinov and Roth, 2009), and comparable to the best published results on this dataset (Passos et al., 2014), as shown in Table 8.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (ixa-pipe-nerc)</td>
<td>92.20</td>
<td>90.19</td>
<td>91.18</td>
</tr>
<tr>
<td>Stanford NER</td>
<td>89.37</td>
<td>87.95</td>
<td>88.65</td>
</tr>
<tr>
<td>Ratinov et al. (2009)</td>
<td>-</td>
<td>-</td>
<td>90.57</td>
</tr>
<tr>
<td>Passos et al. (2014)</td>
<td>-</td>
<td>-</td>
<td>90.90</td>
</tr>
</tbody>
</table>

Table 8: NERC CoNLL 2003 testb results.

Using the same configuration as the one used on the CoNLL 2003 dataset, we evaluated the model using the MEANTIME testset annotated within the NewsReader project. This
Wolff B-PER
, O
currently O
a O
journalist O
in O
Argentina I-LOC
, O
played O
with O
Del I-PER
Bosque I-PER

Figure 6: Example of CoNLL format for NERC with the NewsReader system output

evaluation constitutes a hard out of domain evaluation because even though the gold
standard is news, the text style is quite different to that of Reuters. Furthermore, and
most importantly, the type of named entities annotated in the MEANTIME corpus differs
from the type of annotation done in the CoNLL 2003 dataset. In other words, the criteria
for a string of text to be considered the extent of a named entity greatly differ, which
makes the NERC evaluation in terms of F1 quite hard. Moreover, MEANTIME contains
annotations for nested entities. We therefore provide the results of the ixa-pipe-nerc best
model on the MEANTIME corpus in terms of phrase- and token-based F1 for both inner
and outer extents in Table 9. The 3 classes which map from the CoNLL 2003 annotations
to the MEANTIME datasets, namely, person, organization and location are included in
the evaluation.

<table>
<thead>
<tr>
<th>System</th>
<th>mention extent</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Inner phrase-based</td>
<td>64.68</td>
<td>78.44</td>
<td>70.90</td>
</tr>
<tr>
<td>Stanford NER (all english distsim)</td>
<td>Inner phrase-based</td>
<td>62.66</td>
<td>71.90</td>
<td>66.96</td>
</tr>
<tr>
<td>Illinois NER (CoNLL 2003)</td>
<td>Inner phrase-based</td>
<td>57.19</td>
<td>69.44</td>
<td>62.72</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Inner token-based</td>
<td>73.10</td>
<td>71.85</td>
<td>77.18</td>
</tr>
<tr>
<td>Stanford NER (all english distsim)</td>
<td>Inner token-based</td>
<td>75.93</td>
<td>76.10</td>
<td>76.01</td>
</tr>
<tr>
<td>Illinois NER (CoNLL 2003)</td>
<td>Inner token-based</td>
<td>69.24</td>
<td>72.93</td>
<td>71.04</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Outer phrase-based</td>
<td>55.46</td>
<td>70.53</td>
<td>62.09</td>
</tr>
<tr>
<td>Stanford NER (all english distsim)</td>
<td>Outer phrase-based</td>
<td>52.32</td>
<td>62.95</td>
<td>57.14</td>
</tr>
<tr>
<td>Illinois NER (CoNLL 2003)</td>
<td>Outer phrase-based</td>
<td>47.53</td>
<td>60.52</td>
<td>53.24</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Outer token-based</td>
<td>73.58</td>
<td>68.56</td>
<td>70.98</td>
</tr>
<tr>
<td>Stanford NER (all english distsim)</td>
<td>Outer token-based</td>
<td>77.16</td>
<td>64.43</td>
<td>70.22</td>
</tr>
<tr>
<td>Illinois NER (CoNLL 2003)</td>
<td>Outer token-based</td>
<td>70.26</td>
<td>61.66</td>
<td>65.68</td>
</tr>
</tbody>
</table>

Table 9: NERC Intra-document Benchmarking with MEANTIME.
Table 9 shows the results of the best *ixa-pipe-nerc* model obtained according to the evaluation on the CoNLL 2003 data. However, it uses both the CoNLL 2003 trainset as well as the MUC7 and Ontonotes 4.0 datasets. We compare these results with the best models distributed by the Illinois NER and the Stanford NER taggers. The Illinois NER model is trained on CoNLL 2003 (as shown in Table 8) whereas the Stanford NER best model on the MEANTIME dataset is trained for the three CoNLL entity types (person, location and organization) on a variety of datasets, including MUC6, MUC7, Ontonotes, Web data and CoNLL 2003.

The results show that *ixa-pipe-nerc* outperforms both the Stanford and Illinois NER systems on every type of evaluation, the differences between them being larger in the standard phrase-based F1 score. The comparatively low scores also confirm the difficulty of adapting supervised models to the MEANTIME dataset, although the results for the token based evaluation are higher. It should also be noted that the somewhat larger differences with the Illinois NER tagger are probably due to the fact that we tested with their best CoNLL 2003 model, which albeit competitive, proved to be less robust than the multi corpora models of the Stanford and *ixa-pipe-nerc* systems.

The results are coherent with the previous assertions on out-of-domain evaluation. The drop in performance is in particular not surprising if we consider that the MEANTIME gold standard was annotated with a completely different guidelines stating as to what a *named entity* is. For example, a frequent source of false positives are the following cases, among others:

- Different criteria to decide whether a Named Entity is marked: in the expression “40 billion US air tanker contract” the MEANTIME gold standard does not mark ‘US’ as location, where as in the CoNLL 2003 guidelines this is systematically annotated.

- ‘the United States’ in MEANTIME gold standard vs. ‘United States’ in CoNLL 2003.

- Longer extents containing common nouns: in MEANTIME corpus there are many entities such as “United States airframer Boeing” which in this case is considered an organization, whereas in CoNLL 2003 this extent will be two entities: ‘United States’ as location and ‘Boeing’ as organization.

- Common nouns modifying the proper name: ‘Spokeswoman Sandy Angers’ is annotated as a Named Entity of type person whereas in CoNLL 2003 the extent would be ‘Sandy Angers’ only.

The system presented here is the same for every language of the NewsReader project. Thus, the discussion about the MEANTIME evaluation also applies to Italian, Spanish and Dutch.

Summarizing, the NERC system integrated in NewsReader obtains the best results in the very competitive CoNLL 2003 evaluation. Furthermore, *ixa-pipe-nerc* clearly outperforms a robust Stanford NER system by around 3-4 F1 scores in the phrase based evaluations and by around 0.5-2 F1 scores in the token based evaluations. These differences are
even larger if the CoNLL-only models are used of the multi corpora ones. The results can be reproduced following the procedure explained in the NERC evaluation package.

7.1.2 English Named Entity Disambiguation

In NewsReader, Named Entity Disambiguation is performed using the DBpedia Spotlight technology. More specifically, we use the DBpedia Spotlight probabilistic models. For the evaluation, we will be using the 2010, 2011 English dataset from the TAC KBP editions and the AIDA corpus.

The AIDA corpus contains assignments of entities to the mentions of named entities annotated for the original CoNLL 2003 NERC task. The entities are identified by their YAGO2 entity name, their Wikipedia URL or Freebase.

The TAC KBP 2009 edition distributed a knowledge base extracted from a 2008 dump of Wikipedia and a test set of 3,904 queries. Each query consisted of an ID that identified a document within a set of Reuters news articles, a mention string that occurred at least once within that document, and a node ID within the knowledge base. Each knowledge base node contained the Wikipedia article title, Wikipedia article text, a predicted entity type (person, organization, location or misc), and a key-value list of information extracted from the articles infobox. If the entity referred to did not occur in the knowledge base, it was labelled NIL. A high percentage of queries in the 2009 test set did not map to any nodes in the knowledge base: the gold standard answer for 2,229 of the 3,904 queries was NIL.

In the 2010 challenge, the same configuration and the same knowledge base were used as in the 2009 challenge. In this edition, however, a training set of 1,500 queries was provided, with a test set of 2,250 queries. In the 2010 training set, only 28.4% of the queries were NIL, compared to 57.1% in the 2009 test data and 54.6% in the 2010 test data. In the KBP 2012 edition, the reference KB is derived from English Wikipedia, while source documents come from a variety of languages, including English, Chinese, and Spanish.

The evaluation consists of running the NED system on the standard datasets described above, assessing their overall performance. This section presents the results of this evaluation. The performance of the system is measured using the standard precision and recall metrics. Precision is the number of correctly assigned instances divided by the total instances as returned by the system. Recall is the number of correctly assigned instances divided by the number of instances in the gold standard dataset. In our particular setting, we seek to maximize precision, that is, we care more about returning correct links to DBpedia entities than trying to link all possible mentions in the input text. 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.

---

82 https://github.com/newsreader/evaluation/tree/master/nerc-evaluation
83 http://www.cnts.ua.ac.be/conll2003/ner/
84 http://www.mpi-inf.mpg.de/yago-naga/yago/
85 http://wiki.freebase.com/wiki/Machine_ID
As the module has several parameters, it was optimized in TAC 2011 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.

We have also checked the performance of the NED module on the MEANTIME gold standard of the NewsReader project. We have evaluated the entities disambiguated in the first six sentences of the 120 documents. The entities are automatically obtained using the best model of the *ixa-pipe-nerc* module. Table 10 presents the evaluation results, the number of entities manually annotated as NAM or PRE.NAM, the number of entities automatically identified by the NERC module and the number of entities disambiguated by the NED module. The precision and recall are obtained comparing the manually disambiguated entities with the information contained in the entities layer of the NAF files obtained with the NewsReader pipeline.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Gold</th>
<th>System-NERC</th>
<th>System-NED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>78.39%</td>
<td>62.50%</td>
<td>69.55%</td>
<td>296</td>
<td>265</td>
<td>236</td>
</tr>
<tr>
<td>Airbus</td>
<td>60.00%</td>
<td>66.67%</td>
<td>63.16%</td>
<td>270</td>
<td>339</td>
<td>300</td>
</tr>
<tr>
<td>GM</td>
<td>66.22%</td>
<td>67.35%</td>
<td>66.78%</td>
<td>291</td>
<td>327</td>
<td>296</td>
</tr>
<tr>
<td>Stock</td>
<td>56.83%</td>
<td>47.88%</td>
<td>51.97%</td>
<td>165</td>
<td>164</td>
<td>139</td>
</tr>
<tr>
<td>Total</td>
<td>66.54%</td>
<td>55.07%</td>
<td>60.26%</td>
<td>1058</td>
<td>1064</td>
<td>971</td>
</tr>
</tbody>
</table>

Table 10: Performance of the NED module on the MEANTIME dataset

Looking at the overall results, the results obtained on the MEANTIME dataset are lower compared to the ones obtained in the TAC and AIDA datasets, but the differences vary depending on the corpus. For instance, the NED module obtains similar results on the Apple corpus as on the TAC 2010 dataset. For the rest of the corpora, the module obtains lower results. Overall, the number of entities automatically detected by the NERC module is similar to the ones manually annotated, but the NED module fails when disambiguating some of the entities (see Table 10).

Finally, we have also evaluated the NewsReader pipeline based on the annotation done for the SemEval-2015 Task 4 ([Minard et al., 2015](#)). In this evaluation, we evaluated the pipeline with a set of target entities. Thus, the manually disambiguated entities are not all the entities appearing in the documents but the ones corresponding to the core entities. The evaluation data consist of 3 sets of documents annotated with a set of target entities and each set contains 30 documents. The precision of the system can not be obtained because not all the entities were manually disambiguated so we have only measured the recall of our pipeline. Table 11 presents the results. If we compare the results to those obtained for the MEANTIME dataset, it seems the NewsReader pipeline is more robust across the corpora when targeting the SemEval entities. The information evaluated corresponds to centroid entities.

This evaluation can be reproduced following the procedure explained in the NED eval-
### 7.1.3 English Event detection and coreference

Event detection and event coreference are important for NewsReader since they capture the core of the news. Event detection and event-coreference is very different from entity detection and linking. Firstly, events are not as tangible as entities and only exists during a very limited time-frame. What constitutes an event is not easy to define and to some degree also very subjective. Secondly, most events are not registered in a resource as entities are in e.g. DBpedia, because events are more fluid and temporary. Likewise, it is not surprising that the way people can refer to an event varies a lot. We can thus conclude that event detection and coreference is by far more challenging than entity detection, linking and coreference. Event coreference is important because it forms the basis for defining event instances within and across documents. In this section, we describe the evaluation of detecting events mentions and instances within a single document on the MEANTIME corpus and the extended Event Coreference Bank (ECB+, Cybulska and Vossen, 2014).

The Semantic Role layer (SRL) is the basic input for the event detection and coreference. We take the predicates that are listed in the SRL as a starting point for creating coreference sets in the coreference layer. Any predicate detected by the SRL is a potential event. The coreference layer thus includes both singleton sets (mentions of predicates that do not corefer with other mentions) and multitude set (two or more predicates referring to the same event). Coreference sets are created using different methods, described below.

Event-coreference is measured in different ways in the literature. We use the method BLANC (Pradhan et al., 2014) for the intra-document coreference because it also measures singleton coreference sets. In most cases, there is no coreference relation and a mention of an event within a document represents a unique event only mentioned once. For example, the MEANTIME corpus annotation shows that only 3% of the annotated events have a coreference relation, whereas in the ECB+ corpus only 10% of the event mentions have a coreference relation. Measures that only consider co-referring event mentions therefore do not consider these non-coreferential event mentions.

For the evaluation of the event-coreference, we used the CorScorer package developed by Luo et al., 2014. The CorScorer expects that coreferences are represented in

---

**Table 11: Performance of the NED module on the MEANTIME dataset**

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Recall</th>
<th>Gold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airbus</td>
<td>34.55%</td>
<td>544</td>
</tr>
<tr>
<td>GM</td>
<td>31.80%</td>
<td>585</td>
</tr>
<tr>
<td>Stock</td>
<td>32.61%</td>
<td>279</td>
</tr>
<tr>
<td>Total</td>
<td>33.02%</td>
<td>1408</td>
</tr>
</tbody>
</table>

---

<sup>86</sup> https://github.com/newsreader/evaluation/tree/master/ned-evaluation

<sup>87</sup> https://code.google.com/p/reference-coreference-scorers/
CoNLL2011/2012 format. We thus developed a package\textsuperscript{88} that converts CAT annotations to this format and NAF representations. An example of the output format shown in Figure \ref{fig:example_output}.

\begin{verbatim}
#begin document (3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners);
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 1 Chinese -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 2 airlines -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 3 agree (9)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 4 purchase (10)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 5 of -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 6 Boeing -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 7 787 -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 8 Dreamliners -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 9 worth -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 10 US$ -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 11 7.2 -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 1 12 bn (11)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 17 Officials (12)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 18 from -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 19 the -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 20 People -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 21 's -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 22 Republic -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 23 of -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 24 China -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 25 have -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 26 agreed (9)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 27 to -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 28 purchase (10)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 29 60 -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 30 Boeing -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 31 787 -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 32 Dreamliner -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 33 aircraft -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 34 in -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 35 a -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 36 deal (13)
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 37 worth -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 38 US$ -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 39 7.2 -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 40 bn -
3835_Chinese_airlines_agree_purchase_of_Boeing_787_Dreamliners 3 41 . -
\end{verbatim}

Figure 7: Example of CoNLL2011/2012 format for coreference with the NewsReader system output

The NewsReader pipeline annotates all sentences of an article. Since only a few sentences from each article have been annotated, we implemented a function that reduces the CoNLL file for the manual annotation to only those sentences that have an event annotated and we implemented another function that reduces the system output (response) to the same sentences of the manual CoNLL file (key). We developed 3 different systems for the intra-document coreference:

\textbf{No-coreference baseline} Every predicate in the SRL represents a unique event and no...
coreference relations are created. Since in most cases there is indeed no co-reference within a document (events are mentioned only once), this constitutes a good baseline for the majority case.

**Lemma baseline** All predicates in the SRL that have the same lemma are coreferential, following the hypothesis *one-lemma-one-instance-per-document*.

**Wordnet-wsd-similarity** The output of the lemma baseline is taken as a starting point. We take the highest ranked senses of all the lemma mentions according to the word-sense-disambiguation (WSD). Next we compare the lemma-based coreference set to see how similar these senses are across the lemma-sets. If sufficiently similar, coreference sets are merged and we store the lowest-common-subsumer of the merged set in addition to the top-ranked senses.

The wordnet-based method uses various parameters that can be set and different resources can be used. We used WordNet3.0-LMF as a resource and the Wordnet Similarity method defined by [Leacock and Chodorow, 1998][89] and implemented in the WordNetTools package[90] to measure the similarity. The advantage of using WordNet Tools is that it can use any wordnet in LMF format and therefore it can easily be applied to other languages. We experimented with the following parameters and resources:

- Extending the semantic relations for measuring similarity
- Threshold for selecting the top-ranked senses
- Different thresholds for WordNet similarity
- Different WSD systems for ranking senses
- The maximum sentence distance between mentions of events to be coreferential

We extended the relations with event relations from the Princeton morphosemantic relation file[91] Through these relations, we can establish cross-part-of-speech similarity in addition to similarity based on solely the hypernym relations in WordNet. Extending the relations had a positive impact on the recall. Next, we added ontological relations to WordNet to obtain more connectivity between concepts. We added FrameNet subclass relations from the PredicateMatrix and the ESO hierarchy to FrameNet frames as described in Deliverable D5.2.2 ([van Erp et al., 2015](http://wordnetcode.princeton.edu/standoff-files/morphosemantic-links.xls)). We obtained the following resources:

**WordNet3.0** 95,322 concepts with 97,666 hyperonym relations

**Wordnet3.0.xpos** Extended with cross-part-of-speech event relations: 95,450 concepts with 100,972 hyperonym relations

---

[89]: This can be replaced by any other similarity measure, e.g. based on distributional semantic models.
[90]: git@github.com:cltl/WordnetTools.git
[91]: http://wordnetcode.princeton.edu/standoff-files/morphosemantic-links.xls
WordNet3.0.xpos.fn.eso  Extended with cross-part-of-speech event relation, FrameNet mappings and ESO ontology relations: 95,805 concepts with 107,244 hyperonym relations

We experimented with different similarity thresholds (ranging from scores of 1 up to 3.5) and with selecting different proportion of the ranked senses of the WSD system. If all senses of a word can be used for similarity, this can result in a chaining effect: if word A is most similar to sense s1 of word B and s2 of word B is most similar to word C. As a result, A, B and C are merged into one set but the connection is the result of different and possibly incompatible senses of B. We observed that these chaining effects lead to unnatural and wrong coreference sets. By limiting the senses of a word to the output of the WSD, such effects can be prevented. Our experiments show that we get the best results if we take the 80% highest scoring senses.

We also experimented with using different WSD systems for English: the UKB and the IMS modules described for the English pipeline. By taking the combination of the two systems, we got the best results.

Finally, we experimented with setting a sentence limit within which coreference relations can survive. Due to the limited amount of sentence annotated (5 sentences in MEANTIME and 1.8 sentences on average in ECB+), we observed only a limited effect for setting a sentence distance between mentions. Only for source-introducing predicates, such as speech-act verbs like “say” or “tell”, we found a positive effect when setting the sentence boundary to mentions with a distance of 4 or less sentences.

We provide the evaluations on two different data sets: the English MEANTIME data.

The BLANC scorer provides separate statistics on the mentions of events in the key data and the system response. Table 12 show the results for the English pipeline Version 3.0.

<table>
<thead>
<tr>
<th></th>
<th># key mentions</th>
<th># response mentions</th>
<th># missed mentions</th>
<th># invented mentions</th>
<th># strictly correct identified mentions</th>
<th># partially correct identified mentions</th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2055</td>
<td>2416</td>
<td>464</td>
<td>825</td>
<td>1591</td>
<td>0</td>
<td>0.77</td>
<td>0.66</td>
<td>0.71</td>
</tr>
</tbody>
</table>

Table 12: Detection of event mentions

We can see that there are 825 response mentions invented (not annotated in the key data) but also 464 mentions missed. The recall is 77% and precision is 66%. Error-analysis showed that especially nominal events are not detected but also filtering of non-events will be beneficially ([Caselli et al., 2015]). The recall of 77% forms the upper bound for the
results of event coreference, since the system cannot detect co-reference for events that are missed.

Table 13 shows the results for the two baseline systems. We present two calculations of the results. First we give the cumulated results over all the files across the 4 subcorpora as are given by CorefScorer for REFERENCE links, COREFERENCE links and NOREFERENCE links. The latter are event mentions that have no coreference relations (true singletons), COREFERENCE links are mention coreference relations and REFERENCE links are the sum of both. From the totals given by BLANC, we calculated recall, precision and F1 measures. The last part gives the macro averaged results according to the output of the CorefScorer. BLANC calculates an average across NOREFERENCE and COREFERENCE which is slightly different from the standard calculation on the totals reported.

<table>
<thead>
<tr>
<th></th>
<th>singleton-baseline</th>
<th>lemma-baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REFERENCE (BLANC)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td># reference links</td>
<td>18341</td>
<td>18341</td>
</tr>
<tr>
<td># response reference links</td>
<td>24949</td>
<td>24949</td>
</tr>
<tr>
<td># correct reference links</td>
<td>10673</td>
<td>10723</td>
</tr>
<tr>
<td>recall</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td>precision</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>F1</td>
<td>0.49</td>
<td>0.50</td>
</tr>
<tr>
<td><strong>COREFERENCE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key coreference links</td>
<td>558</td>
<td>558</td>
</tr>
<tr>
<td># response coreference links</td>
<td>0</td>
<td>507</td>
</tr>
<tr>
<td># correct coreference links</td>
<td>0</td>
<td>201</td>
</tr>
<tr>
<td>recall</td>
<td>0</td>
<td>0.36</td>
</tr>
<tr>
<td>precision</td>
<td>NaN</td>
<td>0.40</td>
</tr>
<tr>
<td>F1</td>
<td>NaN</td>
<td>0.38</td>
</tr>
<tr>
<td><strong>NOREFERENCE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key non-coreference links</td>
<td>17783</td>
<td>17783</td>
</tr>
<tr>
<td># response non-coreference links</td>
<td>24949</td>
<td>24442</td>
</tr>
<tr>
<td># correct non-coreference links</td>
<td>10673</td>
<td>10522</td>
</tr>
<tr>
<td>recall</td>
<td>0.60</td>
<td>0.59</td>
</tr>
<tr>
<td>precision</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>F1</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Macro average coreference Identification of Mentions</td>
<td>77.3935</td>
<td>77.3935</td>
</tr>
<tr>
<td>recall</td>
<td>31.84</td>
<td>49.97</td>
</tr>
<tr>
<td>precision</td>
<td>21.73</td>
<td>42.70</td>
</tr>
<tr>
<td>F1</td>
<td>24.54</td>
<td>41.57</td>
</tr>
</tbody>
</table>

Table 13: Singleton and lemma baseline results for event coreference

Given the mention recall of 60%, the overall scores for both baselines is reasonable when calculating recall (58%), precision (43%) and F1 (49%) from the total results. However, the macro average scores from BLANC are more than 20 points lower.

It is important to point to a few issues with the BLANC score. Typically, the results are skewed towards the majority class which is NOCOREFERENCE. In the MEANTIME only 558 COREFERENCE links are annotated, which is 3% of total number of links according to BLANC (18,341). The results for NOREFERENCE thus almost completely determine the results for REFERENCE. The singleton approach therefore already gives good results.
for REFERENCE and even slightly better results for NOREFERENCE than the lemma approach. Obviously, it gives no results for the true coreference sets. Finally due to the nature of BLANC, differences to the REFERENCE relations will likely have an impact on the NOREFERENCE results unless mentions of events go from one reference set to the other. Usually, the coreference approach is less or more strict, leading to larger sets or smaller sets but not resulting into swapping mentions across sets.

In the next tables, we will use these baselines as a reference for the WordNet similarity approach. We first present the results for the WordNet similarity approach using different thresholds for similarity. We keep the selection of the WSD ranked senses to 80% and applied a maximal sentence range of 4 for source events. We used WordNet3.0 extended with the cross-part-of-speech relations. Table 14 show the results for similarity thresholds from 0.5 up to 3.5. The first row in the table shows the number of lowest-common-subsumers or LCSs assigned to coreference sets. The lower the similarity thresholds the larger the coreference sets and the more different LCS results will be obtained. A similarity of 0.5 results in 1,049 LCSs mappings, whereas for similarity of 3.5, which is almost the same as lemma matching with an occasional synonym depending on the depth in the hierarchy, no LCSs matches were found in this data set. A similarity of 0.5 lumps together many events as is shown in the next example. Four different LCS classes group 13 different predicates together resulting in a very unintuitive coreference set.

When we run the software with sim=2.5 the set above dissolves to lemma-based sets and we get coreference sets based on similarity as the following:

Clearly, the similarity function has potential for finding matches but the process is subtle and the settings make small differences with respect to the lemma matches.
can also be observed in Table 14. The first observation to be made is that differences are very small and not significant. For the REFERENCE statistics, we see that similarity of 3.5 gives the same scores as the lemma match, while the scores for sim=2.5 and sim=3.0 are very close. The lower the threshold the lower the scores for recall, precision and F1. However, to see the effect of the similarity, it is better to look at the separate results for COREFERENCE and NOREFERENCE. Although lemma matches work best for COREFERENCE (4 points higher for recall and F1), we see that sim=1.0 gives the highest recall of all similarity values and sim=3.5 gives the highest precision, whereas sim=2.5 gives the highest F1. We now see decreasing recall when the similarity threshold is increased and an increasing precision, as expected. For NOREFERENCE, the majority case, the singleton has the highest recall (0.60), sim=2.5 has the highest precision (0.43062) and sim=3.0 and sim=3.5 have the highest F1 score (0.49911). Finally, the macro averaging generated by BLANC gives the best results for the lemma-baseline, with the sim=2.5 as the best results among the similarity thresholds. Overall, differences in these results are not significant.

This is partially due to the fact that only 5 sentences are annotated per article. We expect more variation when the whole article is considered and especially, when cross-document coreference is considered.

<table>
<thead>
<tr>
<th>BLANC</th>
<th>sim0.5</th>
<th>sim1.0</th>
<th>sim1.5</th>
<th>sim2.0</th>
<th>sim2.5</th>
<th>sim3.0</th>
<th>sim3.5</th>
<th>singleton</th>
<th>lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td># LCS</td>
<td>1049</td>
<td>942</td>
<td>518</td>
<td>189</td>
<td>44</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># ref. links</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
</tr>
<tr>
<td># response ref. links</td>
<td>24468</td>
<td>24838</td>
<td>24884</td>
<td>24932</td>
<td>24949</td>
<td>24949</td>
<td>24949</td>
<td>24949</td>
<td>24949</td>
</tr>
<tr>
<td># correct re. links</td>
<td>8779</td>
<td>10218</td>
<td>10588</td>
<td>10685</td>
<td>10722</td>
<td>10722</td>
<td>10722</td>
<td>10723</td>
<td>10723</td>
</tr>
<tr>
<td>recall</td>
<td>0.47865</td>
<td>0.55711</td>
<td>0.57729</td>
<td>0.58257</td>
<td>0.58459</td>
<td>0.58459</td>
<td>0.58465</td>
<td>0.58192</td>
<td>0.58465</td>
</tr>
<tr>
<td>precision</td>
<td>0.3588</td>
<td>0.41139</td>
<td>0.42549</td>
<td>0.42857</td>
<td>0.42976</td>
<td>0.42976</td>
<td>0.4298</td>
<td>0.42779</td>
<td>0.4298</td>
</tr>
<tr>
<td>F1</td>
<td>0.41015</td>
<td>0.47329</td>
<td>0.48699</td>
<td>0.49384</td>
<td>0.49536</td>
<td>0.49536</td>
<td>0.4954</td>
<td>0.49309</td>
<td>0.4954</td>
</tr>
<tr>
<td>COREFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key coref. links</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
</tr>
<tr>
<td># response coref. links</td>
<td>3617</td>
<td>1269</td>
<td>624</td>
<td>491</td>
<td>445</td>
<td>421</td>
<td>417</td>
<td>0</td>
<td>507</td>
</tr>
<tr>
<td># correct coref. links</td>
<td>171</td>
<td>179</td>
<td>172</td>
<td>170</td>
<td>163</td>
<td>163</td>
<td>163</td>
<td>0</td>
<td>201</td>
</tr>
<tr>
<td>recall</td>
<td>0.30645</td>
<td>0.32079</td>
<td>0.30824</td>
<td>0.30666</td>
<td>0.29211</td>
<td>0.29211</td>
<td>0.36022</td>
<td>0.36022</td>
<td>0.36022</td>
</tr>
<tr>
<td>precision</td>
<td>0.04728</td>
<td>0.14106</td>
<td>0.27564</td>
<td>0.34623</td>
<td>0.38202</td>
<td>0.38717</td>
<td>0.39098</td>
<td>NaN</td>
<td>0.39645</td>
</tr>
<tr>
<td>F1</td>
<td>0.08192</td>
<td>0.19595</td>
<td>0.29103</td>
<td>0.32412</td>
<td>0.33898</td>
<td>0.33299</td>
<td>0.33436</td>
<td>0.37746</td>
<td>NaN</td>
</tr>
<tr>
<td>NOREFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key noref links</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
</tr>
<tr>
<td># response noref links</td>
<td>20851</td>
<td>23569</td>
<td>24260</td>
<td>24441</td>
<td>24504</td>
<td>24524</td>
<td>24532</td>
<td>24532</td>
<td>24544</td>
</tr>
<tr>
<td># correct noref links</td>
<td>8608</td>
<td>10039</td>
<td>10416</td>
<td>10515</td>
<td>10552</td>
<td>10559</td>
<td>10560</td>
<td>10673</td>
<td>10522</td>
</tr>
<tr>
<td>recall</td>
<td>0.48406</td>
<td>0.56453</td>
<td>0.58573</td>
<td>0.5913</td>
<td>0.59338</td>
<td>0.59377</td>
<td>0.59383</td>
<td>0.60018</td>
<td>0.59169</td>
</tr>
<tr>
<td>precision</td>
<td>0.41283</td>
<td>0.42594</td>
<td>0.42935</td>
<td>0.43022</td>
<td>0.43062</td>
<td>0.43049</td>
<td>0.43046</td>
<td>0.42779</td>
<td>0.43049</td>
</tr>
<tr>
<td>F1</td>
<td>0.44562</td>
<td>0.48554</td>
<td>0.49549</td>
<td>0.49806</td>
<td>0.49907</td>
<td>0.49911</td>
<td>0.49911</td>
<td>0.49953</td>
<td>0.49838</td>
</tr>
</tbody>
</table>

Table 14: Event coreference results using WordNet Similarity thresholds from 0.5 to 3.5. WSD ranking threshold is set to 0.8 and sentence range for source event is set to 4. WordNet3.0 with cross-part-of-speech relations is used as a resource.

NewsReader: ICT-316404  February 1, 2016
Table 15 shows the results for the different proportion of senses used to calculate the similarity. We use a similarity threshold of 2.5 and kept the other settings the same as in the previous experiment. The WSD threshold varies from 0.1, i.e. all senses scoring 10% or higher with respect to the top-scoring sense, to 1.0, i.e. only using the top-scoring senses. The scores for the senses are based on the WSD scores for all mentions in the document for all the WSD systems applied to each mention of a predicate.\footnote{Pipeline v3.0 has the output of two systems: UKB and IMS. We experimented with using either of the two but combining the two systems gave best results. We take the proportional output of each system separately and combine both results with respect to the thresholds of the best score of each system.}

<table>
<thead>
<tr>
<th>REFERENCE (BLANC)</th>
<th>wsd-0.1</th>
<th>wsd-0.5</th>
<th>wsd-0.6</th>
<th>wsd-0.7</th>
<th>wsd-0.8</th>
<th>wsd-0.9</th>
<th>wsd-1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td># reference links</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
</tr>
<tr>
<td># correct reference links</td>
<td>10708</td>
<td>10722</td>
<td>10722</td>
<td>10722</td>
<td>10722</td>
<td>10725</td>
<td>10725</td>
</tr>
<tr>
<td>recall</td>
<td>0.58383</td>
<td>0.58459</td>
<td>0.58459</td>
<td>0.58459</td>
<td>0.58459</td>
<td>0.58476</td>
<td>0.58476</td>
</tr>
<tr>
<td>precision</td>
<td>0.4292</td>
<td>0.42976</td>
<td>0.42976</td>
<td>0.42976</td>
<td>0.42976</td>
<td>0.42988</td>
<td>0.42988</td>
</tr>
<tr>
<td>F1</td>
<td>0.4971</td>
<td>0.49536</td>
<td>0.49536</td>
<td>0.49536</td>
<td>0.49536</td>
<td>0.4955</td>
<td>0.4955</td>
</tr>
<tr>
<td>COREFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key coreference links</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
<td>558</td>
</tr>
<tr>
<td># correct coreference links</td>
<td>179</td>
<td>175</td>
<td>175</td>
<td>171</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>recall</td>
<td>0.32079</td>
<td>0.31362</td>
<td>0.31362</td>
<td>0.30645</td>
<td>0.30466</td>
<td>0.30466</td>
<td>0.30466</td>
</tr>
<tr>
<td>precision</td>
<td>0.37214</td>
<td>0.38462</td>
<td>0.38462</td>
<td>0.38255</td>
<td>0.38202</td>
<td>0.38462</td>
<td>0.38636</td>
</tr>
<tr>
<td>F1</td>
<td>0.34456</td>
<td>0.34551</td>
<td>0.34551</td>
<td>0.34303</td>
<td>0.33898</td>
<td>0.34068</td>
<td>0.34068</td>
</tr>
<tr>
<td>NOREFERENCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># key non-coreference links</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
</tr>
<tr>
<td># correct non-coreference links</td>
<td>24468</td>
<td>24494</td>
<td>24494</td>
<td>24502</td>
<td>24504</td>
<td>24507</td>
<td>24509</td>
</tr>
<tr>
<td>recall</td>
<td>0.59208</td>
<td>0.59309</td>
<td>0.59309</td>
<td>0.59332</td>
<td>0.59338</td>
<td>0.59354</td>
<td>0.59354</td>
</tr>
<tr>
<td>precision</td>
<td>0.43032</td>
<td>0.4306</td>
<td>0.4306</td>
<td>0.43062</td>
<td>0.43062</td>
<td>0.43069</td>
<td>0.43066</td>
</tr>
<tr>
<td>F1</td>
<td>0.4984</td>
<td>0.49895</td>
<td>0.49895</td>
<td>0.49904</td>
<td>0.49907</td>
<td>0.49917</td>
<td>0.49915</td>
</tr>
<tr>
<td>Macro average coreference Identification of Mentions</td>
<td>77.3935</td>
<td>77.3935</td>
<td>77.3935</td>
<td>77.3935</td>
<td>77.3935</td>
<td>77.3935</td>
<td>77.3935</td>
</tr>
<tr>
<td>recall</td>
<td>47.4168</td>
<td>47.0468</td>
<td>47.0468</td>
<td>46.8823</td>
<td>46.8469</td>
<td>46.8519</td>
<td>46.8519</td>
</tr>
<tr>
<td>precision</td>
<td>40.3622</td>
<td>40.7462</td>
<td>40.7462</td>
<td>40.7038</td>
<td>40.6898</td>
<td>40.928</td>
<td>40.9269</td>
</tr>
</tbody>
</table>

Table 15: Event coreference results using Word-Sense-Disambiguation thresholds from 0.1 to 1.0. The similarity threshold is set to 2.5 and sentence range for source event is set to 4. WordNet3.0 with cross-part-of-speech relations is used as a resource.

Again the differences are minimal and not significant. For REFERENCE, the highest threshold have best scores for recall, precision and F1. When we look at the differentiated scores for COREFERENCE , we see that lower thresholds result in higher recall: 0.32079 for wsd=0.1, and higher thresholds for more precision: 0.38636 for wsd=1.0. The BLANC macro average shows a similar trend but all differences are within 1 point range.

Finally, Table [16] shows the impact of the resource used for disambiguation. We present WordNet as it is, extended with cross-part-of-speech relations and extended with ontology classes.

For REFERENCE, WordNet extended with the ontological classes performs lower than
Table 16: Event coreference results using different WordNet relations. The similarity threshold is set to 2.5, WSD ranked senses are set to 0.8 of the top-ranked senses and sentence range for source event is set to 4.

<table>
<thead>
<tr>
<th>COREFERENCE</th>
<th>wneng-30</th>
<th>wneng-30.xpos</th>
<th>wneng-30.xpos.fn.eso</th>
</tr>
</thead>
<tbody>
<tr>
<td># reference links</td>
<td>18341</td>
<td>18341</td>
<td>18341</td>
</tr>
<tr>
<td># correct reference links</td>
<td>10722</td>
<td>10722</td>
<td>10550</td>
</tr>
<tr>
<td>recall</td>
<td>0.58459</td>
<td>0.58459</td>
<td>0.57521</td>
</tr>
<tr>
<td>precision</td>
<td>0.42976</td>
<td>0.42976</td>
<td>0.42286</td>
</tr>
<tr>
<td>F1</td>
<td>0.49536</td>
<td>0.49536</td>
<td>0.48741</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOREFERENCE</th>
<th>wneng-30</th>
<th>wneng-30.xpos</th>
<th>wneng-30.xpos.fn.eso</th>
</tr>
</thead>
<tbody>
<tr>
<td># key non-coreference links</td>
<td>17783</td>
<td>17783</td>
<td>17783</td>
</tr>
<tr>
<td># response non-coreference links</td>
<td>24519</td>
<td>24504</td>
<td>24263</td>
</tr>
<tr>
<td># correct non-coreference links</td>
<td>10559</td>
<td>10552</td>
<td>10376</td>
</tr>
<tr>
<td>recall</td>
<td>0.59377</td>
<td>0.59338</td>
<td>0.58348</td>
</tr>
<tr>
<td>precision</td>
<td>0.43065</td>
<td>0.43020</td>
<td>0.42765</td>
</tr>
<tr>
<td>F1</td>
<td>0.49922</td>
<td>0.49907</td>
<td>0.49355</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identification of Mentions</th>
<th>wneng-30</th>
<th>wneng-30.xpos</th>
<th>wneng-30.xpos.fn.eso</th>
</tr>
</thead>
<tbody>
<tr>
<td>recall</td>
<td>46.2098</td>
<td>46.8469</td>
<td>46.4984</td>
</tr>
<tr>
<td>precision</td>
<td>40.3197</td>
<td>40.6898</td>
<td>36.5038</td>
</tr>
<tr>
<td>F1</td>
<td>38.8515</td>
<td>39.2044</td>
<td>36.7065</td>
</tr>
</tbody>
</table>

Based on these evaluations, we can conclude that we only find marginal effects and the lemma-baseline still gives the best results. However, the fine-grained results show that a similarity threshold of 2.5, WSD threshold of 0.8 using WordNet3.0.xpos scores a little better if COREFERENCE is involved, which is also reflected in the BLANC macro average. We expect that these settings are still preferable to the lemma-baseline if the full document is considered and more variation can be expected. We also believe that coreference sets with more variation are better suited for cross-document comparison. In the future, we want to extend the annotation to the full document to validate this claim.

The following settings are recommended for intra-document EventCoreference:

1. sim=2.5
2. wsd=0.8
3. sentence range for source events = 4
4. resource=WordNet3.0.xpos

The evaluation can be reproduced following the procedure explained in the evaluation package.

### 7.1.4 English Nominal coreference

The nominal coreference system in the NewsReader pipeline is Corefgraph, a rule-based system inspired by [Lee et al., 2013].

<table>
<thead>
<tr>
<th>Sieves</th>
<th>Type</th>
<th>CONLL 2011 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sieve 1</td>
<td>Speaker Identification</td>
<td>29.2</td>
</tr>
<tr>
<td>Sieve 2</td>
<td>Exact String Match</td>
<td>45.3</td>
</tr>
<tr>
<td>Sieve 3</td>
<td>Relaxed String Match</td>
<td>45.4</td>
</tr>
<tr>
<td>Sieve 4</td>
<td>Precise Constructs</td>
<td>45.7</td>
</tr>
<tr>
<td>Sieve 5</td>
<td>Strict Head Match A</td>
<td>48.5</td>
</tr>
<tr>
<td>Sieve 6</td>
<td>Strict Head Match B</td>
<td>48.8</td>
</tr>
<tr>
<td>Sieve 7</td>
<td>Strict Head Match C</td>
<td>49.3</td>
</tr>
<tr>
<td>Sieve 8</td>
<td>Proper Head Noun Match</td>
<td>49.5</td>
</tr>
<tr>
<td>Sieve 9</td>
<td>Relaxed Head Match</td>
<td>49.7</td>
</tr>
<tr>
<td>Sieve 10</td>
<td>Pronoun Match</td>
<td>59.3</td>
</tr>
</tbody>
</table>

Table 17: Multi-sieve Pass and CoNLL 2011 dev auto F1 Evaluation.

The approach applies tiers of coreference models one at a time from highest to lowest precision. Each tier builds on the entity clusters constructed by previous models in the sieve, guaranteeing that stronger features are given precedence over weaker ones. Furthermore, each model’s decision is richly informed by sharing attributes across the mentions clustered in earlier tiers. This ensures that each decision uses all of the information available at the time. They implemented all components using only deterministic models. All these components are unsupervised, in the sense that they do not require training on gold coreference links. Furthermore, this framework can be easily extended with arbitrary models, including statistical or supervised models.

This system was the top ranked system at the CoNLL-2011 shared task. Mention detection is included in the package. 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 few of the sieves. Thus, the results show that sieves 1, 2, 5 and 10 provide 97% of the results for that particular evaluation set ([Lee et al., 2011] [Lee et al., 2013]).

Automatic evaluation measures are crucial for coreference system development and comparison. Unfortunately, there is no agreement at present on a standard measure for...
coreference resolution evaluation. First, there are two metrics associated with international
coreference resolution contests: the MUC scorer (Vilain et al., 1995) and the ACE value
(NIST). Second, two commonly used measures, B3 (Bagga and Baldwin, 1998) and
CEAF (Luo, 2005) are also used. Finally, an alternative metric called BLANC was
presented by Recasens et al., 2010. B3 and CEAF are mention-based, whereas MUC and
BLANC are link-based.

These metrics were all used in the CoNLL 2011 and 2012 tasks, and we will be using
the official scorer provided. As or the evaluation of the event-coreference in section 7.1.3
we used the updated CorScorer package developed by Luo et al., 2014. The CorScorer
expects that coreferences are represented in CoNLL2011/2012 format. We therefore use the
package developed within the NewsReader project to convert CAT and NAF annotations
to this format. An example of the output format shown in Figure 7.

Table 18 shows the performance of the English nominal coreference system in News-
Reader (Corefgraph) for a direct comparison with the results shown in Table 17.

<table>
<thead>
<tr>
<th>System</th>
<th>MUC</th>
<th>B3</th>
<th>CEAF</th>
<th>BLANC</th>
<th>CoNLL 2011 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>59.6</td>
<td>68.3</td>
<td>45.5</td>
<td>73.0</td>
<td>59.3</td>
</tr>
<tr>
<td>NewsReader (Corefgraph)</td>
<td>51.0</td>
<td>67.2</td>
<td>43.4</td>
<td>69.7</td>
<td>54.8</td>
</tr>
</tbody>
</table>

Table 18: Multi-sieve Pass and CoNLL 2011 dev-auto Evaluation

The results show that Corefgraph still performs lower than the Stanford System on the
dev-auto dataset. In order to assess if the performance difference is due to the coreference
resolution algorithm, we also evaluated Corefgraph in the supplementary closed track gold
boundaries dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>MUC</th>
<th>B3</th>
<th>CEAF</th>
<th>CoNLL 2011 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>80.05</td>
<td>69.7</td>
<td>66.80</td>
<td>72.18</td>
</tr>
<tr>
<td>NewsReader (Corefgraph)</td>
<td>79.56</td>
<td>68.09</td>
<td>65.44</td>
<td>71.03</td>
</tr>
</tbody>
</table>

Table 19: Multi-sieve Pass and CoNLL 2011 closed track gold boundaries Evaluation

We obtain mixed results: on the closed track gold boundaries evaluation, we obtain
results much closer to the Stanford Multi-Sieve Pass system (Lee et al., 2013). This
difference is larger when evaluated on the dev-auto dataset. Although it is difficult to draw
firm conclusions on these results, this suggest that the differences in performance when
evaluated with the dev-auto corpus may be related with the mention detection algorithm.
Overall, the performance shown seems to indicate that the system behaves competitively
when compared with other publicly available Coreference system such as the Stanford
system.

95 Vilain1995model
96 https://code.google.com/p/reference-coreference-scorers/
97 https://github.com/cltl/coreference-evaluation
We also evaluated the performance on the MEANTIME gold standard annotated within the NewsReader project. The results are shown in Table 20.

<table>
<thead>
<tr>
<th>System</th>
<th>MUC</th>
<th>B^3</th>
<th>CEAF</th>
<th>CoNLL 2011 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (Corefgraph)</td>
<td>19.70</td>
<td>18.34</td>
<td>18.96</td>
<td>19.00</td>
</tr>
</tbody>
</table>

Table 20: MEANTIME Nominal Coreference Evaluation

Apart from the usual amount of errors that a coreference evaluation usually produces, the results on the MEANTIME dataset are not surprising if we consider a number of issues:

- **Singletons**: In the MEANTIME only some singletons are annotated. The evaluation provided by Table 20 removes singletons after annotation. This is related to the next point.

- **Guidelines**: The coreference annotation guidelines for MEANTIME are quite different to the Ontonotes and CoNLL 2011/2012 guidelines. In particular, singletons are annotated, but not all of them. If we do not use the singletons for evaluation, we under-generate mentions whereas if leave the singletons after annotation then Corefgraph over-generates. This is due to the fact that in Corefgraph and Ontonotes, every named entity, personal pronoun and, crucially, every other noun phrase (except in a few cases) is considered to be a mention. For example, in file 8983 of the airbus segment of MEANTIME, there are some singletons annotated such as “50 787s”, but many others are left out: “The Air Indian order”, “The Air Canada deal”, “a deal”, “a further US 7 billion”, “two last orders for new aircraft”, “11 billion of aircraft deals”.

- **Cascading errors**: We have seen in Section 7.1.1 that there is a misalignment between the corpus used to train the NERC module (CoNLL 2003) and the MEANTIME guidelines for named entities. As many of the mentions are named entities, the performance of Corefgraph suffers from cascading errors in the pipeline. For example, the NERC module will correctly (according to CoNLL 2003 guidelines) annotate “Heathrow” as named entity and thus as a mention but in the MEANTIME annotation the named entity annotation is “Heathrow airport”.

The evaluations can be reproduced following the procedure explained in the nominal coreference evaluation package.

### 7.1.5 English Semantic Role Labelling

In NewsReader, Semantic Role Labelling for English is carried out using the MATE-tools ([Björkelund et al., 2009](https://github.com/newsreader/evaluation/tree/master/nominal-coreference-evaluation)). This software is a pipeline that includes linguistic processors that perform lemmatization, part-of-speech tagging, dependency parsing, and semantic role
labeling of a sentence. The dependency parser had the top score for English for dependency parsing and SRL on the CoNLL shared task 2009 (Hajić et al., 2009). The performance of the current version of the system on that task is given in Table 21.

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled precision</td>
<td>( \frac{19137 + 10036}{22467 + 10818} )</td>
<td>87.65%</td>
</tr>
<tr>
<td>Labeled recall</td>
<td>( \frac{19137 + 10036}{24748 + 10818} )</td>
<td>82.02%</td>
</tr>
<tr>
<td>Labeled F1</td>
<td></td>
<td>84.74%</td>
</tr>
<tr>
<td>Unlabeled precision</td>
<td>( \frac{20697 + 10818}{22467 + 10818} )</td>
<td>94.68%</td>
</tr>
<tr>
<td>Unlabeled recall</td>
<td>( \frac{20697 + 10818}{24748 + 10818} )</td>
<td>88.61%</td>
</tr>
<tr>
<td>Unlabeled F1</td>
<td></td>
<td>91.55%</td>
</tr>
</tbody>
</table>

Table 21: Performance of MATE on the English dataset of CoNLL-2009

For the NewsReader pipeline we have developed a wrapper that includes only the dependency parser and the SRL system of MATE-tools. That means that the rest of the analysis used as input for this modules, such as lemmatization and part-of-speech tagging, are obtained by the tools included in the NewsReader pipeline. In order to evaluate this configuration we have checked the performance of the SRL module on the MEANTIME gold standard of the NewsReader project. This dataset contains 120 files with 597 sentences. Applying CoNLL-2009 scorer, we obtain the results in Table 22.

<table>
<thead>
<tr>
<th></th>
<th>Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled precision</td>
<td>( \frac{1032 + 1636}{5373 + 2423} )</td>
<td>34.22%</td>
</tr>
<tr>
<td>Labeled recall</td>
<td>( \frac{1032 + 1636}{5490 + 2058} )</td>
<td>35.35%</td>
</tr>
<tr>
<td>Labeled F1</td>
<td></td>
<td>34.78%</td>
</tr>
<tr>
<td>Unlabeled precision</td>
<td>( \frac{1149 + 1636}{5373 + 2423} )</td>
<td>35.72%</td>
</tr>
<tr>
<td>Unlabeled recall</td>
<td>( \frac{1149 + 1636}{5490 + 2058} )</td>
<td>36.90%</td>
</tr>
<tr>
<td>Unlabeled F1</td>
<td></td>
<td>36.30%</td>
</tr>
</tbody>
</table>

Table 22: Performance of MATE on MEANTIME

Looking at the figures in Table 22 the MATE-tools seem to perform very poorly on the MEANTIME dataset. However, several considerations must be taken into account. The MEANTIME corpus was developed from raw text with the purpose of identifying the semantic interpretations we pursue in NewsReader. Annotators thus first annotated events, participants and times and then identified relations between them. The CoNLL datasets were combine dependency structures with SRL annotations. Their semantic role annotations thus took syntactic structures into account. It was not feasible to include additional syntactic annotations in the already elaborate annotation efforts within NewsReader. Regarding SRL, it seems that the main problems using the evaluation framework of CoNLL09 are:

- The manual annotations do not cover all predicates and mentions.

MATE detects more predicates than those annotated (2,423 compared to 2,058). Additionally, MATE usually provides arguments for these predicates which obviously are also not annotated. This means that we can not correctly calculate
the precision of MATE. However, if we consider only the 2,058 predicates annotated, MATE correctly identifies 1,636. That is, 79% recall.

- Some arguments have been manually annotated as CLINKs or SLINKs instead of propBank arguments:

  These are cases such as:

  “He said he closed the door”. While MATE considers “he closed the door” as arg1 of “said”, the annotation contains an SLINK (subordinate link) between “said” and “closed”.

  “I started to run”. While MATE considers “to run” as argument C-arg1 of “started”, the annotation contains a GLINK (Grammatical link) between “started” and “run”.

  Thus, in these cases, the current evaluation of MATE is wrongly penalized.

- The manual annotations of the arguments correspond to spans and not heads.

  CoNLL09 evaluation expects heads instead of spans. This means it expects one head per argument. This is why we only used the heads in our evaluation. Thus, if we annotate in the gold-standard the full span, let’s say five tokens, the scorer evaluates five gold arguments instead of only one. This is why the system returns a low recall.

  Obviously, the combination of the three problems produce quite misleading and unreliable results. For that reason, we have also evaluated the system taking the whole span from NAF instead of using the head of the arguments. The results are show in Table 23. In this case, the labeled and unlabeled recall reaches to 67% and 80%, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Labeled precision</th>
<th>Labeled recall</th>
<th>Labeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>(\frac{3464 + 1636}{26481 + 2423})</td>
<td>17.64%</td>
<td></td>
</tr>
<tr>
<td>Recall</td>
<td>(\frac{3464 + 1636}{5490 + 2058})</td>
<td>67.57%</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>(\frac{3464 + 1636}{26481 + 2423})</td>
<td>27.98%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Unlabeled precision</th>
<th>Unlabeled recall</th>
<th>Unlabeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>(\frac{4422 + 1636}{26481 + 2423})</td>
<td>20.96%</td>
<td></td>
</tr>
<tr>
<td>Recall</td>
<td>(\frac{4422 + 1636}{5490 + 2058})</td>
<td>80.26%</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>(\frac{4422 + 1636}{26481 + 2423})</td>
<td>33.24%</td>
<td></td>
</tr>
</tbody>
</table>

Table 23: Performance of MATE on MEANTIME with the full span from NAF.

The evaluation can be reproduced following the procedure explained in the SRL evaluation package.

https://github.com/newsreader/evaluation/tree/master/srl-evaluation
7.1.6 English Temporal processing

For the evaluation of temporal processing modules we developed a scorer based on the evaluation methodology used for the TempEval3 task (UzZaman et al., 2013). It uses the script relation_to_timegraph.py of the TempEval3 evaluation toolkit.100

The scorer has been used for the EVENTI-Evalita 2014 evaluation campaign.101 The scorer takes input files in the CAT labelled format. We thus developed a package that converts NAF layers in CAT labelled format. During this conversion the non text-consuming TIMEX3 representing the document creation time is deleted. The document creation time is extracted from the metadata of a document and not from the text. It is important to annotate it with a non text-consuming TIMEX3 when extracting the temporal relations between events. However, in the MEANTIME corpus the document creation time is explicitly expressed in texts and during the manual annotation temporal relations have been built using the text-consuming document creation time. This is a specificity of the corpus which has been done for helping the annotators.

For time expression recognition and normalization two evaluations are done: strict matching and relaxed matching. For example, if the gold annotation contains “Tuesday evening” and the system detects “Tuesday”, then they will get credit in relaxed matching but not in exact matching. We compute the F1-score of attributes (type and value) by multiplying attribute’s accuracy by the F1-score obtained for time expression recognition.

For temporal relation extraction we provide three evaluations: strict matching, relaxed matching and temporal awareness. In the first case two relations match if their sources and their targets strictly match, as well as their types (BEFORE, AFTER, INCLUDES, etc.). In the second case, a relaxed matching is considered between the sources and the targets. The TLINK relations are also evaluated using the evaluation methodology of UzZaman and Allen, 2011. This evaluation has been used for the TempEval3 task (UzZaman et al., 2013). The metric proposed by UzZaman and Allen, 2011 captures the temporal awareness of an annotation in terms of precision, recall and F1-score.

The intra-document annotation Guidelines (Tonelli et al., NWR2014-2-2) allows the annotation of non text-consuming TIMEX3, for example in order to represent the begin point of a duration. Currently the TimePro module for English does not extract non text-consuming TIMEX3. We decided to do the evaluation without taking into account the non text-consuming TIMEX3. The evaluation will be computed again when TimePro will be extended with new functionalities to annotate non text-consuming TIMEX3.

In Table 24 we present the results of TimePro on the 4 subcorpora and the micro-average on the whole corpus. The measure uses is the recall, precision and F1-score. We provide the evaluation on three aspects: the recognition of time expression extents, their classification (date, time, set or duration) and their normalization.

The temporal relation extraction module (TempRelPro) extracts relations between two event mentions or between an event mention and a time expression or between two time expressions.
Table 24: TimePro performance

<table>
<thead>
<tr>
<th></th>
<th>recognition</th>
<th></th>
<th>classification</th>
<th></th>
<th>normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>recall</td>
<td>precision</td>
<td>F1</td>
<td>F1 type</td>
<td>F1 value</td>
</tr>
<tr>
<td>strict match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.820</td>
<td>0.948</td>
<td>0.879</td>
<td>0.841</td>
<td>0.783</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.761</td>
<td>0.909</td>
<td>0.828</td>
<td>0.817</td>
<td>0.817</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.724</td>
<td>0.907</td>
<td>0.805</td>
<td>0.780</td>
<td>0.606</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.636</td>
<td>0.865</td>
<td>0.733</td>
<td>0.687</td>
<td>0.595</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.725</td>
<td>0.905</td>
<td><strong>0.805</strong></td>
<td>0.774</td>
<td><strong>0.685</strong></td>
</tr>
<tr>
<td>relaxed match</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.865</td>
<td>1</td>
<td>0.928</td>
<td>0.870</td>
<td>0.792</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.848</td>
<td>0.974</td>
<td>0.907</td>
<td>0.860</td>
<td>0.814</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.791</td>
<td>0.981</td>
<td>0.876</td>
<td>0.826</td>
<td>0.628</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.715</td>
<td>0.973</td>
<td>0.824</td>
<td>0.763</td>
<td>0.656</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.795</td>
<td>0.982</td>
<td><strong>0.879</strong></td>
<td>0.824</td>
<td><strong>0.711</strong></td>
</tr>
</tbody>
</table>

To understand the results better, it is important to take the evaluation of event detection into account (see Table 12 in Section 7.1.3) and of time expression (see Table 24). In fact, through this evaluation we are evaluating entity pairs extraction and classification as well as events and time expressions extraction.

In Table 25 we present the scores of the TempRelPro module. The tempeval3 evaluation methodology is based on a timegraph; taking the timex-timex relations into account enables the creation of a more complete timegraph and consequently results in a more accurate evaluation of TempRelPro performance. The TempRelPro system thus extracts relation between time expressions while these relations are not annotated in the gold standard corpus. We do not take timex-timex relations in the strict matching and relaxed matching evaluation into account, because these relations would have all been counted as false positives.

The system achieved a micro-average F1 score of 22.6 using the temporal evaluation methodology proposed by [UzZaman and Allen, 2011]. The temporal relation extraction results are reasonable given the fact that the event detection performance is 71.0 F1 score (see Table 12) and the time expression recognition is 80.5 F1 score (see Table 24). In comparison: the best system in TempEval 3, ClearTK-2, obtained a F1 score of 30.98 for the task ABC on temporal relation extraction from raw text, achieving a F1 score of 82.71 on time expression recognition and 77.34 on event detection.

If we evaluate the system considering only the relations between entities (events and/or timexes) matching between the reference and the prediction, the F1 score obtained by TempRelPro is 31.5.

We see two problems in the evaluation of such system that can explain the low results. First of all, only a small subset of possible pairs are manually annotated, namely the most central and obvious relations. In the NewsReader Guidelines the annotation procedure is divided into 5 subtasks: TLINKs between event mentions and the document creation.
<table>
<thead>
<tr>
<th></th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>strict match</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.392</td>
<td>0.122</td>
<td>0.187</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.305</td>
<td>0.089</td>
<td>0.138</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.445</td>
<td>0.140</td>
<td>0.213</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.197</td>
<td>0.057</td>
<td>0.089</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.347</td>
<td>0.105</td>
<td>0.161</td>
</tr>
<tr>
<td>relaxed match</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.407</td>
<td>0.127</td>
<td>0.194</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.324</td>
<td>0.095</td>
<td>0.147</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.455</td>
<td>0.143</td>
<td>0.217</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.216</td>
<td>0.063</td>
<td>0.097</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.362</td>
<td>0.110</td>
<td>0.169</td>
</tr>
<tr>
<td>temporal awareness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.479</td>
<td>0.190</td>
<td>0.272</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.383</td>
<td>0.140</td>
<td>0.205</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.509</td>
<td>0.182</td>
<td>0.268</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.344</td>
<td>0.089</td>
<td>0.142</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.423</td>
<td>0.155</td>
<td>0.226</td>
</tr>
</tbody>
</table>

Table 25: TempRelPro performance

time, TLINKs between main event mentions, TLINKs between main event mentions and subordinated event mentions in the same sentence, TLINKs between event mentions and time expressions in the same sentence and TLINKs between time expressions. Following these subtasks, an annotator should be able to annotate the most central relations, but not a complete timegraph between all events, while a system would extract all possible relations.

The system is based on a machine learning method and is trained on TimeBank and AQUAINT corpora (corpora distributed for TempEval3). These two corpora have been annotated following TimeML guidelines, which gives the following instruction for the annotation of TLINK: “A TLINK has to be created each time a temporal relationship holding between events or an event and a time needs to be annotated”. The resulting annotation of TLINKs differ from the annotation done in NewsReader, that is to say that the training corpus differs from the evaluation corpus, which probably partially explains the low results.

The hypothesis that the low performance is caused by differences in training and evaluation data rather than problems with the system is strengthened by the fact that our system ranked first out of five systems and achieved a F1-score of 40.0 at the QA TempEval task at SemEval 2015 (http://alt.qcri.org/semeval2015/task5/). The performance of the system was measured through the performance of a QA system using temporal relations automatically extracted by systems.

The evaluation can be reproduced following the procedure explained in the time pro-
The module currently used for identifying factuality is the first version of a new module for factuality recognition. It consists of two components: a factuality identifier that is trained on the most embedded layer of FactBank and a rule-based approach that aims at determining whether an event is situated in the future or not. A more detailed explanation is provided in Section 3.17. The module assigns factuality values to all events identified in the pipeline. The current evaluation provides results for events found in the gold standard and the output of the pipeline.

Table 26 provides an overview of the performance of the factuality module on the four subcorpora and the MEANTIME corpora as a whole. The first three rows provide the accuracy on determining polarity, certainty and time, respectively, whereas the last line indicates where the system assigned the correct label to all three. Overall, the accuracy on polarity and certainty is relatively high, whereas time classification performs relatively poorly. This can be explained by the fact that the current implementation classifies all events for which the verbal morphology of the target word or its syntactic head does not provide a direct indication of future, past or present tense receives the label ‘UNDERSPECIFIED’, whereas the gold only classifies events as such that are explicitly underspecified as far as the time of occurrence is concerned. In practice, output that is underspecified by the module will be interpreted as NON_FUTURE, which should lead to better results.

One of the reasons for the high accuracy of the factuality module is that factuality values have very strong dominant classes. The far majority of events are presented as certain and with positive polarity. This bias is found both in FactBank and the gold standard data. In order to gain insight into performance of factuality, it is thus necessary to investigate how it performs on individual classes.

Table 27 provides an overview of the performance of the factuality module on individual classes. The results confirm that the high results are mainly due to the dominant classes: recall, precision and F-score are both in the nineties for positive polarity and certain events, but significantly lower for the values that occur less often. The results furthermore confirm the explanation provided above for relatively poor results on determining time: underspecified time has extremely poor precision and recall of the other two values is relatively low.
Future research will focus on improving the values of the minority classes for factuality values and providing a richer system for determining whether an event takes place in the future. The former will mainly consist of improving the features and classifiers (recall that the current release is the very first version of this module) and incorporating modifiers when identifying the time of an event.

The evaluation can be reproduced following the procedure explained in the factuality evaluation package.103

7.2 Dutch

The Dutch pipeline contains components for Named Entity Recognition and Classification (NERC), Named Entity Disambiguation (NED), Event Recognition and Temporal Expressions, Semantic Role Labelling and Event Coreference. The following subsection display the results.

7.2.1 Dutch NERC

The Dutch pipeline uses the \textit{ixa-pipe-nerc} system \cite{Agerri2014} \footnote{https://github.com/ixa-ehu/ixa-pipe-nerc} for named entity recognition, just like the English and Spanish pipelines. As mentioned in previous sections, \textit{ixa-pipe-nerc} uses the perceptron algorithm described by \cite{Collins2002} \footnote{https://github.com/newsreader/evaluation/tree/master/factuality-evaluation} to train its models. The module has been evaluated on two standard datasets: CoNLL02 and SoNar. The results are presented in Table \ref{table:nerc}. Ixa-pipe-nerc performs higher than competitive systems on both CoNLL and SoNar. This confirms the result also obtained for other languages that \textit{ixa-pipe-nerc} is, to our knowledge, the highest performing system currently available.

We also present the results on the MEANTIME corpus in Table \ref{table:nerc}. The results were generated with the same evaluation package as employed for the other languages.\footnote{https://github.com/newsreader/evaluation/tree/master/nominal-coreference-evaluation}

\begin{table}[h]
\centering
\begin{tabular}{llll}
\hline
  & recall & precision & F1  \\
\hline
POL_POS  & 91.8% & 95.1% & 93.4%  \\
POL_NEG  & 33.3% & 50.0% & 40.0%  \\
POL_UNDERSPECIFIED & 30.6% & 15.9% & 21.0%  \\
CERT_CERTAIN & 93.0% & 93.5% & 93.3%  \\
CERT_PROBABLE & 50.0% & 55.6% & 52.6%  \\
CERT_POSSIBLE & 5.4% & 57.1% & 9.9%  \\
CERT_UNDERSPECIFIED & 28.3% & 12.3% & 17.2%  \\
TIME_FUTURE & 17.8% & 86.8% & 29.6%  \\
TIME_NON_FUTURE & 70.2% & 91.2% & 79.3%  \\
TIME_UNDERSPECIFIED & 55.0% & 1.9% & 3.7%  \\
\hline
\end{tabular}
\caption{Performance of factuality module on individual classes} \label{table:factuality}
\end{table}
results outperform those for Italian and are comparable to those for Spanish and, for the outer evaluation, English.

<table>
<thead>
<tr>
<th>mention extent</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner phrase-based</td>
<td>57.13</td>
<td>72.57</td>
<td>63.93</td>
</tr>
<tr>
<td>Inner token-based</td>
<td>65.62</td>
<td>75.56</td>
<td>70.24</td>
</tr>
<tr>
<td>Outer phrase-based</td>
<td>56.62</td>
<td>72.02</td>
<td>63.40</td>
</tr>
<tr>
<td>Outer token-based</td>
<td>66.29</td>
<td>75.69</td>
<td>70.68</td>
</tr>
</tbody>
</table>

Table 29: NERC Intra-document Benchmarking on Dutch MEANTIME corpus.

### 7.2.2 Dutch Named Entity Disambiguation

Table 30 provides an overview of the results of Named Entity Disambiguation for Dutch following the same evaluation process as for the other languages described in Section 7.1.2. The results for Dutch Named Entity Disambiguation are around 10 points lower compared to English, Italian and Spanish. This result is comparable to those reported in [Daiber et al., 2013b] for DBpedia Spotlight, where the accuracy for Dutch lies around 20% lower than English. For future versions of the pipeline, we will investigate the performance of various disambiguation algorithms. [Daiber et al., 2013b] also describe a new algorithm that significantly improves results for English and Dutch. It is at present not clear how easy it would be to integrate this in the pipeline.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Gold</th>
<th>System-NED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>46.00%</td>
<td>35.66%</td>
<td>40.17%</td>
<td>258</td>
<td>200</td>
</tr>
<tr>
<td>Airbus</td>
<td>48.18%</td>
<td>54.32%</td>
<td>51.06%</td>
<td>243</td>
<td>274</td>
</tr>
<tr>
<td>GM</td>
<td>53.57%</td>
<td>64.66%</td>
<td>58.59%</td>
<td>232</td>
<td>280</td>
</tr>
<tr>
<td>Stock</td>
<td>55.62%</td>
<td>56.25%</td>
<td>55.93%</td>
<td>176</td>
<td>178</td>
</tr>
<tr>
<td>Total</td>
<td>50.84</td>
<td>52.72</td>
<td>51.44</td>
<td>909</td>
<td>932</td>
</tr>
</tbody>
</table>

Table 30: Performance of the NED module on the MEANTIME dataset.
7.2.3 Dutch Intra-document Event coreference

The semantic role labeller in the Dutch pipeline considers all verbs identified by Alpino (the parser used for Dutch) as predicates. A WordNet-based ontotagger identifies nominal predicates that refer to an event. Each predicate corresponds to an event in our evaluation. Table 31 provides an overview of the outcome of event detection. With the exception of the Dutch recall compared to that for Spanish, the results for event detection lack behind the other languages. This can largely be explained by the method used to identify them: the SRL labeller considers all verbs identified by Alpino as event mentions. This leads to both over-classifications of copula being identified as events and events of other part-of-speech being missed. The nominal-event identifier recovers some of the missed nominal events, but events expressed by other part-of-speech cannot be identified by this approach.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># key mentions</td>
<td>2235</td>
</tr>
<tr>
<td># response mentions</td>
<td>2335</td>
</tr>
<tr>
<td># missed mentions</td>
<td>1046</td>
</tr>
<tr>
<td># invented mentions</td>
<td>946</td>
</tr>
<tr>
<td># strictly correct identified mentions</td>
<td>1289</td>
</tr>
<tr>
<td># partially correct identified mentions</td>
<td>0</td>
</tr>
<tr>
<td>recall</td>
<td>0.58</td>
</tr>
<tr>
<td>precision</td>
<td>0.55</td>
</tr>
<tr>
<td>F1</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 31: Detection of event mentions

Table 32 shows the results for event coreference in Dutch. As for the rest of the languages, we present two calculations of the results. First we give the cumulated results over MEANTIME as given by CorefScorer for REFERENCE links, COREFERENCE links and NOREFERENCE links. The latter are event mentions that have no coreference relations (true singletons), COREFERENCE links are mention coreference relations and REFERENCE links are the sum of both. From the totals given by BLANC, we calculated recall, precision and F1 measures. The last part gives the macro averaged results according to the output of the CorefScorer. BLANC calculates an average across NOREFERENCE and COREFERENCE which is slightly different from the standard calculation on the totals reported.

With an F-score of 26.78 the results for Dutch event coreference significantly lack behind English (41.57), Italian (49.36) and, to a less extend, Spanish (30.37). The difference in performance for event coreference is correlated to performance on event identification. The pipeline cannot identify coreference between event mentions it missed in the identification step and coreferents that should not be event lead to errors in precision. Because event identification scores around 82 and 77 for Italian and English respectively (compared to 56.41 for Dutch), it is likely that poor event identification is the sole cause of the lower performance for identifying event coreference.
### Table 32: Results for event coreference

<table>
<thead>
<tr>
<th></th>
<th># reference links</th>
<th># response reference links</th>
<th># correct reference links</th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference (BLANC)</td>
<td>22272</td>
<td>28010</td>
<td>6869</td>
<td>0.31</td>
<td>0.25</td>
<td>0.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># key coreference links</th>
<th># response coreference links</th>
<th># correct coreference links</th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coreference</td>
<td>680</td>
<td>1123</td>
<td>104</td>
<td>0.15</td>
<td>0.09</td>
<td>0.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th># key non-coreference links</th>
<th># response non-coreference links</th>
<th># correct non-coreference links</th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noreference</td>
<td>21592</td>
<td>26887</td>
<td>6765</td>
<td>0.31</td>
<td>0.25</td>
<td>0.28</td>
</tr>
</tbody>
</table>

Macro average coreference identification of Mentions: 58.89
 recall: 26.42
 precision: 26.78
 F1: 26.78

### 7.2.4 Dutch Semantic Role Labelling

The Dutch semantic role labeller is trained on SoNar. The module pulls all main verbs and its arguments from the output of Alpino, extracts relevant features and uses TimBL to classify the predicate’s argument. We performed 10-fold cross-validation on the correctly identified predicates on SoNar and the labeler obtained an F1-score of 74.01 which is comparable to that obtained by the original implementation on gold predicates.

Because the SoNar semantic role annotations were only applied to verbs, we carry out two additional steps to identify nominal predicates referring to events and to identify roles of prepositional phrases associated with these roles. The results of the semantic role labeller on the MEANTIME corpus are provided in Table 33.

<table>
<thead>
<tr>
<th></th>
<th>Labeled precision</th>
<th>Labeled recall</th>
<th>Labeled F1</th>
<th>Unlabeled precision</th>
<th>Unlabeled recall</th>
<th>Unlabeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntactic head only</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Labeled precision</td>
<td>((89 + 884) / (3209 + 1293)) * 100</td>
<td>21.61%</td>
<td></td>
<td>((415 + 884) / (3209 + 1293)) * 100</td>
<td>28.85%</td>
<td></td>
</tr>
<tr>
<td>Labeled recall</td>
<td>(89 + 884) / (1364 + 1405) * 100</td>
<td>35.14%</td>
<td></td>
<td>(415 + 884) / (1364 + 1405) * 100</td>
<td>46.91%</td>
<td></td>
</tr>
<tr>
<td>Labeled F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35.73%</td>
</tr>
</tbody>
</table>

Table 33: Performance of Dutch SRL module on MEANTIME corpus
Table 34: Performance of Dutch SRL module on MEANTIME corpus (full span from NAF)

The Dutch srl evaluation faces the same challenges with the MEANTIME corpus as the other languages (see Section 7.1.5). As was also done for the other languages, we evaluated both the system taking the whole span from NAF instead as well as the system using the head of the arguments. According to the span evaluation, the labeled and unlabeled recall reaches around 39% and 67%, respectively. However, precision for this evaluation is exceptionally poor leading to lower f-scores compared to the head-based evaluation. Moreover, this evaluation is still inaccurate because it does not consider the full span as a single argument (see Section 7.1.5). This is why the precision results are relatively low. In order to make a proper evaluation using the CoNLL-2009 scorer, we should annotate the heads in the gold standard and deal with the rest of not annotated predicates (and arguments) and the CLINKs and SLINKs as arguments. The overall results are comparable to those for the other languages.

### 7.2.5 Dutch Temporal Processing

<table>
<thead>
<tr>
<th></th>
<th>recognition</th>
<th>classification</th>
<th>normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>recall</td>
<td>precision</td>
<td>F1</td>
</tr>
<tr>
<td><strong>strict match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.535</td>
<td>0.657</td>
<td>0.590</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.457</td>
<td>0.492</td>
<td>0.474</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.495</td>
<td>0.500</td>
<td>0.497</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.476</td>
<td>0.446</td>
<td>0.461</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.492</td>
<td>0.513</td>
<td><strong>0.502</strong></td>
</tr>
<tr>
<td><strong>relaxed match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.547</td>
<td>0.671</td>
<td>0.603</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.557</td>
<td>0.615</td>
<td>0.585</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.611</td>
<td>0.628</td>
<td>0.619</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.562</td>
<td>0.536</td>
<td>0.548</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.570</td>
<td>0.604</td>
<td><strong>0.587</strong></td>
</tr>
</tbody>
</table>

Table 35: Time performance

In Table 35 we present the results of Heideltime on the 4 subcorpora and the micro-
average on the whole corpus for Dutch. The measure uses is the recall, precision and F1-score. We provide the evaluation on three aspects: the recognition of time expression extents, their classification (date, time, set or duration) and their normalization (see Section 7.1.6). The results for identifying temporal expressions are around 30% lower in F-score when compared to the results for Italian, English and Spanish. Heideltime is a pattern and rule based system and the patterns for Dutch were developed for a biographical dictionary and not for (financial) news. FBK developed the system for Italian and English using in-domain machine learning. The Spanish results were also obtained by Heideltime, which leads to the conclusion that the relatively low results are mainly due to the Dutch resources being developed for another domain. The high quality of TimePro combined with the correspondence of the domain compared to a system using out-of-domain rules provides the main explanation between the two systems. In addition, it turned out that the Dutch translations of MEANTIME still contain English dates. Some errors were thus the result of a translation error rather than an error of the system.

7.3 Italian

7.3.1 Italian Named Entity Recognition and Classification

In Table 36 we present the results obtained by EntityPro, the named entity recognition module for Italian, on the MEANTIME corpus. The results were generated with the same evaluation package as employed for the other languages (see Section 7.1.1). The results are lower than those obtained for English, Dutch and Spanish.

<table>
<thead>
<tr>
<th>mention extent</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner phrase-based</td>
<td>48.91</td>
<td>44.95</td>
<td>46.85</td>
</tr>
<tr>
<td>Inner token-based</td>
<td>61.04</td>
<td>53.07</td>
<td>56.77</td>
</tr>
<tr>
<td>Outer phrase-based</td>
<td>46.87</td>
<td>43.13</td>
<td>44.92</td>
</tr>
<tr>
<td>Outer token-based</td>
<td>60.51</td>
<td>48.76</td>
<td>54.00</td>
</tr>
</tbody>
</table>

Table 36: NERC Intra-document Benchmarking with MEANTIME.

One of the reason for obtaining low results on MEANTIME is the entity extent definition. The extents in the NewsReader annotations differ from those in the corpus used to train EntityPro. In MEANTIME we have included in the extent of entities definite articles and articulated prepositions (e.g. *Il New York Times*) whereas they are not annotated by the NERC module (e.g. *New York Times*). EntityPro participated to the Named Entity Recognition task at EVALITA 2007\textsuperscript{106} and it was the best performing (82.1% F1). This confirms that the module for Italian is state-of-the-art software. The lower results compared to the other languages provides additional support the observations made above that *ixa-pipe-nerc* which is used for the other languages is robust across domains compared to other tools.

\textsuperscript{106}http://www.evalita.it/2007/tasks/ner
7.3.2 Italian Named Entity Disambiguation

Table 37 provides the results obtained by the EHU-ned module on Italian. The evaluation process is the same as for the other languages and it is described in Section 7.1.2. The overall F-score is close to that obtained for English, but the system obtains higher recall and lower precision for Italian.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Gold</th>
<th>System-NED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>75.21%</td>
<td>63.08%</td>
<td>68.62%</td>
<td>279</td>
<td>234</td>
</tr>
<tr>
<td>Airbus</td>
<td>79.34%</td>
<td>63.30%</td>
<td>70.42%</td>
<td>267</td>
<td>213</td>
</tr>
<tr>
<td>GM</td>
<td>73.58%</td>
<td>41.64%</td>
<td>53.18%</td>
<td>281</td>
<td>159</td>
</tr>
<tr>
<td>Stock</td>
<td>67.65%</td>
<td>25.27%</td>
<td>36.80%</td>
<td>182</td>
<td>68</td>
</tr>
<tr>
<td>Total</td>
<td>75.37</td>
<td>50.35</td>
<td>60.37</td>
<td>1009</td>
<td>674</td>
</tr>
</tbody>
</table>

Table 37: Performance of the NED module on the MEANTIME dataset

7.3.3 Italian Intra-document Event Coreference

The evaluation of event coreference against the MEANTIME corpus follows the same method as for English (see Section 7.1.3). The main explanations of values are repeated below for convenience.

The BLANC scorer provides separate statistics on the mentions of events in the key data and the system response. Table 38 show the results of EventPro for the detection of event mentions in Italian.

<table>
<thead>
<tr>
<th># key mentions</th>
<th>2150</th>
</tr>
</thead>
<tbody>
<tr>
<td># response mentions</td>
<td>2053</td>
</tr>
<tr>
<td># missed mentions</td>
<td>414</td>
</tr>
<tr>
<td># invented mentions</td>
<td>317</td>
</tr>
<tr>
<td># strictly correct identified mentions</td>
<td>1736</td>
</tr>
<tr>
<td># partially correct identified mentions</td>
<td>0</td>
</tr>
<tr>
<td>recall</td>
<td>0.81</td>
</tr>
<tr>
<td>precision</td>
<td>0.85</td>
</tr>
<tr>
<td>F1</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 38: Detection of event mentions

We can see that there are 317 response mentions invented (not annotated in the key data) but also 414 mentions missed. The recall is 81% and precision is 85%. The event detection system participated at the EVENTI task at Evalita 2014 and obtained 86.7% F1. The F-score is higher than what obtained for English, Spanish and Dutch.

Table 39 shows the results of the event coreference system. We present two calculations of the results. First, we give the cumulated results over all the files across the 4 subcorpora.
as are given by CorefScorer for REFERENCE links, COREFERENCE links and NOREF-
ERENCE links. The latter are event mentions that have no coreference relations (true
singletons), COREFERENCE links are mention coreference relations and REFERENCE
links are the sum of both. From the totals given by BLANC, we calculated recall, precision
and F1 measures. The last part gives the macro averaged results according to the output
of the CorefScorer. BLANC calculates an average across NOREFERENCE and COREF-
ERENCE which is slightly different from the standard calculation on the totals reported.
Compared to English, the recall obtained for Italian is lower, but the precision and the
overall F-score are higher. The performance of the system for Italian are better than for
Spanish and Dutch.

<table>
<thead>
<tr>
<th>REFERENCE (BLANC)</th>
</tr>
</thead>
<tbody>
<tr>
<td># reference links</td>
</tr>
<tr>
<td># response reference links</td>
</tr>
<tr>
<td># correct reference links</td>
</tr>
<tr>
<td>recall</td>
</tr>
<tr>
<td>precision</td>
</tr>
<tr>
<td>F1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COREFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td># key coreference links</td>
</tr>
<tr>
<td># response coreference links</td>
</tr>
<tr>
<td># correct coreference links</td>
</tr>
<tr>
<td>recall</td>
</tr>
<tr>
<td>precision</td>
</tr>
<tr>
<td>F1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOREFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td># key non-coreference links</td>
</tr>
<tr>
<td># response non-coreference links</td>
</tr>
<tr>
<td># correct non-coreference links</td>
</tr>
<tr>
<td>recall</td>
</tr>
<tr>
<td>precision</td>
</tr>
<tr>
<td>F1</td>
</tr>
</tbody>
</table>

Macro average coreference
Identification of Mentions 82.32
recall 47.08
precision 59.69
F1 49.36

Table 39: Results for event coreference

7.3.4 Italian Semantic Role Labelling

As we did in the English pipeline, we have checked the performance of the SRL module
on the MEANTIME gold standard of the NewsReader project. Applying the CoNLL-2009
scorer, we obtain the results in Table 40. The Italian SRL evaluation faces the same
challenges as the evaluation for English (see Section 7.1.5).

We have also evaluated the system taking the whole span from NAF instead of using
the head of the arguments. The results are show in Table 41.
Labeled precision \[
\frac{523 + 1775}{3170 + 2095} \approx 43.65 \%
\]
Labeled recall \[
\frac{523 + 1775}{7109 + 2162} \approx 24.79\%
\]
Labeled F1 \[
31.62\%
\]

Unlabeled precision \[
\frac{879 + 1793}{3170 + 2095} \approx 50.75\%
\]
Unlabeled recall \[
\frac{879 + 1793}{7109 + 2162} \approx 28.82\%
\]
Unlabeled F1 \[
36.76\%
\]

Table 40: Performance of fbk-srl on MEANTIME

Labeled precision \[
\frac{1972 + 1775}{12978 + 2095} \approx 24.86\%
\]
Labeled recall \[
\frac{1972 + 1775}{7109 + 2162} \approx 40.42\%
\]
Labeled F1 \[
30.78\%
\]

Unlabeled precision \[
\frac{3523 + 1793}{12978 + 2095} \approx 35.27\%
\]
Unlabeled recall \[
\frac{3523 + 1793}{7109 + 2162} \approx 57.34\%
\]
Unlabeled F1 \[
43.67\%
\]

Table 41: Performance of fbk-srl on MEANTIME with the full span from NAF.

Using this more accurate evaluation, the labeled and unlabeled recall reaches to 40% and 57%, respectively. Nevertheless, as was also the case for English, this evaluation is still inaccurate because it does not consider the full span as a single argument. This is why the precision results are relatively low. In order to make a proper evaluation using the CoNLL-2009 scorer, we should annotate the heads in the gold standard and deal with the rest of not annotated predicates (and arguments) and the CLINKs and SLINKs as arguments.

Italian SRL reveals good performance: the results are lower that what obtained for English but higher that those obtained for Dutch and Spanish.

7.3.5 Italian Temporal Processing

In Table 42 we present the results of TimePro on the 4 subcorpora and the micro-average on the whole corpus. The measure used is the recall, precision and F1-score. We provide the evaluation on three aspects: the recognition of time expression extents, their classification (date, time, set or duration) and their normalization (see Section 7.1.6). Compared to the rest of the languages, TimePro performances better for the task of time expression recognition.

Table 43 presents the scores of the TempRelPro module. We follow the same procedure as for English repeating the approach here for convenience. The TempRelPro system extracts relations between time expressions while these relations are not annotated in the gold standard corpus. We do not take into account timex-timex relations in the strict matching and relaxed matching evaluation, because these relations would have all been counted as false positive. Whereas as the tempeval3 evaluation methodology is based on a timegraph, taking into account the timex-timex relations enables the creation of a more
<table>
<thead>
<tr>
<th></th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
<th>classification F1 type</th>
<th>normalization F1 value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>strict match</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.881</td>
<td>0.857</td>
<td>0.869</td>
<td>0.842</td>
<td>0.688</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.789</td>
<td>0.877</td>
<td>0.830</td>
<td>0.819</td>
<td>0.749</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.822</td>
<td>0.862</td>
<td>0.841</td>
<td>0.810</td>
<td>0.579</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.838</td>
<td>0.925</td>
<td>0.879</td>
<td>0.738</td>
<td>0.610</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.834</td>
<td>0.882</td>
<td><strong>0.857</strong></td>
<td>0.799</td>
<td><strong>0.646</strong></td>
</tr>
<tr>
<td><strong>relaxed match</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.945</td>
<td>0.920</td>
<td>0.932</td>
<td>0.896</td>
<td>0.715</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.844</td>
<td>0.938</td>
<td>0.889</td>
<td>0.877</td>
<td>0.772</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.907</td>
<td>0.935</td>
<td>0.921</td>
<td>0.874</td>
<td>0.606</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.905</td>
<td>0.993</td>
<td>0.947</td>
<td>0.806</td>
<td>0.629</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.903</td>
<td>0.949</td>
<td><strong>0.926</strong></td>
<td>0.859</td>
<td><strong>0.669</strong></td>
</tr>
</tbody>
</table>

Table 42: TimePro performance

The system achieved a micro-average F1 score of 13.1% using the temporal evaluation methodology proposed by [UzZaman and Allen, 2011]. The results are lower than those obtained for English. This can be explained by the fact that even if the methods of the two systems are the same, the annotation guidelines followed in order to annotate the English and Italian training datasets are different.

The version of TempRelPro used for Italian extracts only temporal relations between two entities (events and/or timexes) in the same sentence. Whereas in the reference MEAN-TIME corpus, temporal relations are annotated between event mentions and the document creation time, between main event mentions, between main event mentions and subordinated event mentions in the same sentence, between event mentions and time expressions in the same sentence and between time expressions. The very low results are explained by this difference in annotating the relations between the manual evaluation corpus and the system. If we evaluate the system considering only the relations between entities (events and/or timexes) matching between the reference and the prediction, the F1 score obtained by TempRelPro is 15.1%.

At the EVENTI task at EVALITA 2014[^107], TimePro was the best performing system on time expression recognition (82.7% F1) and class detection (80% F1). On the temporal relation extraction task and temporal relation classification task TempRelPro was the only system participating and obtained an F1 score of 26.4% and 73.8% respectively.

[^107]: [http://www.evalita.it/2014/tasks/eventi](http://www.evalita.it/2014/tasks/eventi)
### Event Detection, version 3

<table>
<thead>
<tr>
<th></th>
<th>recall</th>
<th>precision</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>strict match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.076</td>
<td>0.324</td>
<td>0.123</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.045</td>
<td>0.310</td>
<td>0.078</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.064</td>
<td>0.268</td>
<td>0.104</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.079</td>
<td>0.287</td>
<td>0.124</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.066</td>
<td>0.296</td>
<td>0.108</td>
</tr>
<tr>
<td><strong>relaxed match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.078</td>
<td>0.333</td>
<td>0.127</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.047</td>
<td>0.310</td>
<td>0.083</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.066</td>
<td>0.277</td>
<td>0.107</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.079</td>
<td>0.287</td>
<td>0.124</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.068</td>
<td>0.303</td>
<td>0.111</td>
</tr>
<tr>
<td><strong>temporal awareness</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.125</td>
<td>0.238</td>
<td>0.164</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.070</td>
<td>0.181</td>
<td>0.101</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.096</td>
<td>0.171</td>
<td>0.123</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.103</td>
<td>0.180</td>
<td>0.131</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.099</td>
<td>0.194</td>
<td>0.131</td>
</tr>
</tbody>
</table>

Table 43: TempRelPro performance

#### 7.3.6 Italian Factuality

In Table 44, we present the performance of the module for event factuality annotation in Italian (FactPro). For each corpus as well for the overall MEANTIME corpus, we give the accuracy on determining polarity, certainty and time. The last line of the table contains the accuracy of the system on identifying all three attributes correctly for the same event.

<table>
<thead>
<tr>
<th></th>
<th>apple</th>
<th>airbus</th>
<th>gm</th>
<th>stock</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>polarity</td>
<td>97.7%</td>
<td>97.4%</td>
<td>76.4%</td>
<td>99.1%</td>
<td>92.2%</td>
</tr>
<tr>
<td>certainty</td>
<td>90.5%</td>
<td>88.5%</td>
<td>75.0%</td>
<td>86.5%</td>
<td>84.8%</td>
</tr>
<tr>
<td>time</td>
<td>83.6%</td>
<td>77.6%</td>
<td>70.6%</td>
<td>87.4%</td>
<td>79.5%</td>
</tr>
<tr>
<td>all 3 correct</td>
<td>77.3%</td>
<td>71.7%</td>
<td>58.8%</td>
<td>81.4%</td>
<td>71.9%</td>
</tr>
</tbody>
</table>

Table 44: Overall accuracy of the factuality module

The system for event factuality annotation in Italian performs better than the one for English.

NewsReader: ICT-316404

February 1, 2016
7.4 Spanish

7.4.1 Spanish Named Entity Recognition and Classification

As previously mentioned, in NewsReader we use the ixa-pipe-nerc system\footnote{https://github.com/ixa-ehu/ixa-pipe-nerc} \cite{Agerri:2014} (Agerri et al., 2014) off-the-self to train our NERC models; ixa-pipe-nerc learns supervised models via the Perceptron algorithm as described by \cite{Collins:2002b}. First we evaluate our NERC system on the CoNLL 2002 official testset and we obtained the following results: 84.18 precision, 84.15 recall and 84.16 F1.

Table \ref{table:nerc_results} presents the results of the ixa-pipe-nerc best model on the MEANTIME corpus in terms of phrase- and token-based F1 for both inner and outer extents for the 3 classes which map from the CoNLL 2002 to the MEANTIME datasets, namely, person, organization and location.

<table>
<thead>
<tr>
<th>System</th>
<th>Mention extent</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Inner phrase-based</td>
<td>61.75</td>
<td>62.54</td>
<td>62.14</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Inner token-based</td>
<td>65.33</td>
<td>65.75</td>
<td>65.54</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Outer phrase-based</td>
<td>58.29</td>
<td>59.60</td>
<td>58.94</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>Outer token-based</td>
<td>71.1</td>
<td>60.94</td>
<td>65.63</td>
</tr>
</tbody>
</table>

Table \ref{table:nerc_results}: NERC Intra-document Benchmarking with MEANTIME.

The NERC system integrated in NewsReader obtains the best results in the very competitive CoNLL 2002 evaluation and it loses some performance when evaluated on the MEANTIME dataset. The results can be reproduced following the procedure explained in the nerc evaluation package\footnote{https://github.com/newsreader/evaluation/tree/master/nerc-evaluation}.

7.4.2 Spanish Named Entity Disambiguation

In NewsReader, Named Entity Disambiguation is performed using the DBpedia Spotlight technology. More specifically, we use the DBpedia Spotlight probabilistic models. 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.

We have also checked the performance of the NED module on the MEANTIME gold standard of the NewsReader project. We have evaluated the entities disambiguated in the first six sentences of the 120 documents. The entities are automatically obtained using the best model of the ixa-pipe-nerc module. Table \ref{table:ned_results} presents the evaluation results, the number of entities manually annotated as NAM or PRE.NAM, the number of entities automatically identified by the NERC module and the number of entities disambiguated by the NED module. The precision and recall are obtained comparing the manually disambiguated entities with the information contained in the entities layer of the NAF files obtained with the NewsReader pipeline.

<table>
<thead>
<tr>
<th>System</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>61.75</td>
<td>62.54</td>
<td>62.14</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>65.33</td>
<td>65.75</td>
<td>65.54</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>58.29</td>
<td>59.60</td>
<td>58.94</td>
</tr>
<tr>
<td>NewsReader (ixa-pipe-nerc combined)</td>
<td>71.1</td>
<td>60.94</td>
<td>65.63</td>
</tr>
</tbody>
</table>
Overall, the results obtained on the MEANTIME dataset are lower compared to those obtained in the TAC 2012 datasets, but the differences vary depending on the corpus. In the case of Apple, Airbus and Stock, the differences are smaller while the General Motors corpus reveals a bigger drop in precision. There is one exception though. The recall of the Airbus corpus is considerably bigger than the recall values obtained in the rest of the datasets. We also observed differences in performance in the English dataset, but there the Apple dataset clearly outperformed the other sets. Overall the system performs better for Spanish than for English (more than 5 points in F-score over the entire corpus).

### 7.4.3 Spanish Intra-document Event Coreference

The evaluation of event coreference follows the same method as for English (see Section 7.1.3). The BLANC scorer provides separate statistics on the mentions of events in the key data and the system response. Table 47 show the results regarding the detection of event mentions for Spanish.

<table>
<thead>
<tr>
<th></th>
<th># key mentions</th>
<th># response mentions</th>
<th># missed mentions</th>
<th># invented mentions</th>
<th># strictly correct identified mentions</th>
<th># partially correct identified mentions</th>
<th>recall</th>
<th>precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2126</td>
<td>1195</td>
<td>1039</td>
<td>108</td>
<td>1087</td>
<td>0</td>
<td>0.51</td>
<td>0.91</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Table 47: Detection of event mentions

We can see that there are 108 response mentions invented (not annotated in the key data) and 1,039 mentions missed. The recall is quite low (51%), but the precision is high (91%). Compared to the rest of the languages, the recall obtained for Spanish is lower, but the precision is higher in all the cases.

Table 48 shows the results for event coreference in Spanish. As for the rest of the languages, we present two calculations of the results. First we give the cumulated results over MEANTIME as given by CorefScorer for REFERENCE links, COREFERENCE links and NewsReader:

|                      | REFERENCE links | COREFERENCE links | NewsReader: ICT-316404 | February 1, 2016 |
NOREFERENCE links. The latter are event mentions that have no coreference relations (true singletons). COREFERENCE links are mention coreference relations and REFERENCE links are the sum of both. From the totals given by BLANC, we calculated recall, precision and F1 measures. The last part gives the macro averaged results according to the output of the CorefScorer. BLANC calculates an average across NOREFERENCE and COREFERENCE which is slightly different from the standard calculation on the totals reported.

<table>
<thead>
<tr>
<th></th>
<th>NOREFERENCE</th>
<th>COREFERENCE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># reference links</td>
<td>19832</td>
<td>619</td>
<td></td>
</tr>
<tr>
<td># response reference links</td>
<td>5871</td>
<td>206</td>
<td></td>
</tr>
<tr>
<td># correct reference links</td>
<td>4671</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>recall</td>
<td>0.23</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>precision</td>
<td>0.79</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>0.36</td>
<td>0.22</td>
<td></td>
</tr>
</tbody>
</table>

Macro average coreference
Identification of Mentions 66.23
recall 22.69
precision 58.84
F1 30.37

Table 48: Results for event coreference

7.4.4 Spanish Nominal Coreference

For Spanish, we have evaluated the performance of our nominal coreference module on the SemEval 2010 task on Multilingual Coreference resolution and the MEANTIME dataset. After the module had been adapted from English, we evaluated on the publicly available SemEval 2010 datasets. We obtained 64.22 F1, while the best system in the task (SUCCRE) obtained 67.50 F1. We also evaluated the module on the MEANTIME dataset. The results are shown in Table 49. The explanations for the relatively low results for English also hold for Spanish (see Section 7.1.4 for an overview).
Table 49: MEANTIME Nominal Coreference Evaluation

<table>
<thead>
<tr>
<th>System</th>
<th>MUC</th>
<th>B³</th>
<th>CEAF</th>
<th>CoNLL 2011 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewsReader (Corefgraph)</td>
<td>16.00</td>
<td>15.01</td>
<td>16.21</td>
<td>15.74</td>
</tr>
</tbody>
</table>

7.4.5 Spanish Semantic Role Labelling

As mentioned in Subsection 7.1.5, in NewsReader, we have chosen the MATE-tools ([Björkelund et al., 2009](#)) to perform Semantic Role Labelling for English. One of the main advantages of this suite is that it is developed with multilingual capabilities. Spanish is included in the languages covered by the MATE-tools. For that reason, we also selected this software for SRL in Spanish. Table 50 contains the results obtained by the current version of the MATE-tools on the Spanish dataset of the CoNLL-2009 shared task ([Hajič et al., 2009](#)).

<table>
<thead>
<tr>
<th></th>
<th>Labeled precision</th>
<th>Labeled recall</th>
<th>Labeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((8934 + 4394) / (11632 + 5175))</td>
<td>((8934 + 4394) / (11824 + 5175))</td>
<td>78.40%</td>
</tr>
<tr>
<td></td>
<td>79.30%</td>
<td>78.85%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Unlabeled precision</th>
<th>Unlabeled recall</th>
<th>Unlabeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((10642 + 5175) / (11632 + 5175))</td>
<td>((10642 + 5175) / (11824 + 5175))</td>
<td>93.58%</td>
</tr>
<tr>
<td></td>
<td>94.11%</td>
<td>93.05%</td>
<td></td>
</tr>
</tbody>
</table>

Table 50: Performance of MATE on the Spanish dataset of CoNLL-2009

As we did for the English pipeline, we have checked the performance of the SRL module on the MEANTIME gold standard of the NewsReader project. This dataset contains 120 files with 597 sentences. Applying CoNLL-2009 scorer, we obtain the results in Table 51.

<table>
<thead>
<tr>
<th></th>
<th>Labeled precision</th>
<th>Labeled recall</th>
<th>Labeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((634 + 1125) / (2530 + 1204))</td>
<td>((634 + 1125) / (5931 + 2188))</td>
<td>47.11%</td>
</tr>
<tr>
<td></td>
<td>47.11%</td>
<td>21.67%</td>
<td>29.68%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Unlabeled precision</th>
<th>Unlabeled recall</th>
<th>Unlabeled F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>((831 + 1125) / (2530 + 1204))</td>
<td>((831 + 1125) / (5931 + 2188))</td>
<td>52.38%</td>
</tr>
<tr>
<td></td>
<td>52.38%</td>
<td>24.09%</td>
<td>33.00%</td>
</tr>
</tbody>
</table>

Table 51: Performance of MATE on MEANTIME

Because the same evaluation issues explained in Section 7.1.5 apply to Spanish, we have also evaluated the system taking the whole span from NAF instead of using the head of the arguments. The results are show in Table 52.

According to this evaluation, the labeled and unlabeled recall reaches to 42% and 56%, respectively. Nevertheless, this evaluation is still inaccurate because it does not consider the full span as a single argument (see Section 7.1.5). This is why the precision results are relatively low. In order to make a proper evaluation using the CoNLL-2009 scorer, we should annotate the heads in the gold standard and deal with the rest of not annotated predicates (and arguments) and the CLINKs and SLINKs as arguments.
Table 52: Performance of MATE on MEANTIME with the full span from NAF.

7.4.6 Spanish Temporal Processing

In Table 53, we present the results of Heideltime for Spanish on the 4 subcorpora and the micro-average on the whole corpus. The measure uses is the recall, precision and F1-score. We provide the evaluation on three aspects: the recognition of time expression extents, their classification (date, time, set or duration) and their normalization (see Section 7.1.6).

<table>
<thead>
<tr>
<th></th>
<th>recognition</th>
<th>classification</th>
<th>normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>recall</td>
<td>precision</td>
<td>F1 type</td>
</tr>
<tr>
<td><strong>strict match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.743</td>
<td>0.681</td>
<td>0.711</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.854</td>
<td>0.854</td>
<td>0.854</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.752</td>
<td>0.782</td>
<td>0.767</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.742</td>
<td>0.896</td>
<td>0.812</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.766</td>
<td>0.801</td>
<td><strong>0.783</strong></td>
</tr>
<tr>
<td><strong>relaxed match</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apple</td>
<td>0.899</td>
<td>0.891</td>
<td>0.895</td>
</tr>
<tr>
<td>Airbus, Boeing</td>
<td>0.921</td>
<td>0.944</td>
<td>0.932</td>
</tr>
<tr>
<td>GM, Chrysler, Ford</td>
<td>0.884</td>
<td>0.935</td>
<td>0.909</td>
</tr>
<tr>
<td>Stock market</td>
<td>0.801</td>
<td>0.976</td>
<td>0.880</td>
</tr>
<tr>
<td>Micro-average</td>
<td>0.868</td>
<td>0.937</td>
<td><strong>0.901</strong></td>
</tr>
</tbody>
</table>

Table 53: Time performance

7.5 Multilingual comparison

This section summarizes the evaluation results obtained in the standard datasets as well as MEANTIME for all the languages of the project. Every module is evaluated using the standard metrics and, when available, datasets for each task. We furthermore present the performance in out of domain data using the MEANTIME corpus. All NLP modules obtain state of the art performances on standard datasets. The performance drops when evaluating the pipelines in the MEANTIME dataset. These results are coherent with previous assertions on out of domain evaluations. Even though many standard datasets consist of news data, the style of the wikinews articles is rather different from these text.
Moreover, the NewsReader annotation guidelines resulted in annotations that were not always compliant with those present in standard sets. The NewsReader annotations aimed for semantic annotations suitable for our domain. This led (among others) to more fine-grained span annotations and classifications for named entities and different structures for semantic roles.

<table>
<thead>
<tr>
<th>Evaluation metric</th>
<th>English</th>
<th>Dutch</th>
<th>Italian</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>91.18</td>
<td>-</td>
<td>82.10</td>
<td>84.16</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>77.18</td>
<td>70.24</td>
<td>56.77</td>
<td>65.54</td>
</tr>
<tr>
<td>Nominal coref.</td>
<td>CoNLL 2011</td>
<td>-</td>
<td>-</td>
<td>CoNLL 2011</td>
</tr>
<tr>
<td>Standard</td>
<td>CoNLL 2011</td>
<td>-</td>
<td>-</td>
<td>SemEval 2010</td>
</tr>
<tr>
<td>F1</td>
<td>71.03</td>
<td>-</td>
<td>-</td>
<td>64.22</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>19.00</td>
<td>-</td>
<td>-</td>
<td>15.74</td>
</tr>
<tr>
<td>Standard</td>
<td>CoNLL 2009</td>
<td>-</td>
<td>-</td>
<td>CoNLL 2009</td>
</tr>
<tr>
<td>F1</td>
<td>84.74</td>
<td>-</td>
<td>-</td>
<td>78.85</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>34.78</td>
<td>26.76</td>
<td>31.62</td>
<td>29.68</td>
</tr>
<tr>
<td>Time expr.</td>
<td>TempEval3</td>
<td>TempEval3</td>
<td>TempEval3</td>
<td>TempEval3</td>
</tr>
<tr>
<td>Standard</td>
<td>TempEval3</td>
<td>-</td>
<td>EVALITA 2014</td>
<td>-</td>
</tr>
<tr>
<td>F1</td>
<td>79.61</td>
<td>-</td>
<td>82.7</td>
<td>-</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>80.50</td>
<td>58.70</td>
<td>85.7</td>
<td>78.30</td>
</tr>
<tr>
<td>Temporal relation</td>
<td>TempEval3</td>
<td>-</td>
<td>TempEval3</td>
<td>-</td>
</tr>
<tr>
<td>Standard</td>
<td>-</td>
<td>-</td>
<td>EVALITA 2014</td>
<td>-</td>
</tr>
<tr>
<td>F1</td>
<td>-</td>
<td>-</td>
<td>26.4</td>
<td>-</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>22.6</td>
<td>-</td>
<td>13.1</td>
<td>-</td>
</tr>
<tr>
<td>Factuality</td>
<td>Standard R</td>
<td>-</td>
<td>Standard R</td>
<td>-</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>55.45</td>
<td>-</td>
<td>71.9</td>
<td>-</td>
</tr>
<tr>
<td>Event coref.</td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
<td>F1</td>
</tr>
<tr>
<td>MEANTIME</td>
<td>41.57</td>
<td>27.32</td>
<td>49.36</td>
<td>30.37</td>
</tr>
</tbody>
</table>

Table 54: Evaluation results on Standard benchmark datasets and NewsReader MEANTIME in the 4 languages of the project

Table 7.5 provides an overview of these results. Overall, the English pipeline obtains better performance compared to the rest of the languages in the standard datasets. Recall that the poor performance on the MEANTINE corpus is mainly due to the annotation guidelines not always being compliant with those used in standard sets. See explanation in this section and in sections describing the corresponding modules.
and MEANTIME. This outcome is expected: the amount of resources and training data for English go far beyond those for other languages. NLP tools generally perform better on the English language than on others. Taking this in consideration, the results for the other languages exceed our expectations. There are some results that lack a little behind (e.g. Spanish and Dutch event coreference and Dutch temporal expression recognition), but most results come close to those obtained for English. Some Italian and Spanish modules even outperform those for English. The NED module obtains better results for Italian and Spanish when evaluating it in the MEANTIME dataset and the temporal processing module obtains higher results for Italian. The overall results show that NewsReader managed to achieve its ambition to support multilingual event extraction building state-of-the-art pipelines for event extraction in all four languages.

8 Scaling of Text Processing

The details of scaling processing are provided in D2.2 ([Rigau et al., 2015]) and D2.3 ([Beloki et al., 2016]). We provide an overview of the main efforts in this section.

The scaling capabilities of our NLP processing pipelines have been studied during the three cycles of event detection. In the first cycle, we performed some initial experiments and as a result of the analysis 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].

In the second cycle, we designed a fully distributed architecture for event detection following a streaming processing framework using STORM. We also developed a set of scripts (called “VM from scratch”) to automatically create a processing cluster using the distributed system. Furthermore, in collaboration with SURFsara and the Netherlands eScience center, we also implemented the NewsReader processing pipeline following a batch setting on Hadoop.

We furthermore carried out efficiency experiments using both architectures. We measured the average processing time of individual modules as well as more fine-grained module internal analyses. These internal analyses revealed how much time modules need to read in input structure, start up external resources, carry out their task and produce output. An overview of the results can be found in [Beloki et al., 2016]. Based on the outcome of these experiments, the ixa-pipe-srl module was modified so that its machine learning component could run on a server. This new setup reduces processing time significantly with potential saving of up to 30s processing time per document. For shorter to average news articles, this can add up to reducing processing time for this module with 70-80%.

In the last cycle of the project, we implemented the NewsReader integrated system where all the components of the NewsReader processing chain are integrated into one single unified system. Event processing is deployed into clusters of machines, so that the processing is highly optimized and executed in parallel.

111 These numbers are based on the analysis presented in Figure 50 in [Beloki et al., 2016].
During the whole project, we have collected and processed different datasets. The collected data has been used in the experiments to test the scaling capabilities. It was also used in the different Hackathons, end-user evaluations and exploitation meetings. So far, we have stored the following data: LN car company news (EN) : 2.5M articles; TechCrunch (EN) : 43K articles; WikiNews (EN, ES, IT) : 19K English, 7K Spanish and 6K Italian; ECB+ (EN) : 2K articles; FIFA World Cup (EN): LexisNexis, BBC, The Guardian (212K news); Criminal network (EN): 900K articles; Dutch House of Representatives (NL): 627K.

In the first NewsReader Hackathon held in London, the FIFA 2014 World Cup documents were used. They were processed within a 15 day time-frame using 30 copies of the virtual machines. In the second and third Hackathons, the LN car company news articles were used (1.23M in year 2 and 2.5 million in year 3). These were processed using Hadoop. The 1.23M articles processed by Version 2 of the pipeline were created in approximately 11 days. The 2.5 million articles processed with the richer pipeline Version 3 took approximately three weeks. The average processing time with pipeline Version 3 is longer than for Version 2 despite the optimization of the ixa-pipe-time module, because several new modules were added to the pipeline. The new modules for identifying temporal and causal relations in particular involve complex calculations and have a relatively high processing time per document. It should be noted, however, that the Hadoop infrastructure is shared with other researchers and that the overall processing time depends on how crowded the cluster is. The minimum time required for 1M articles in year 2 (using the full structure of 680 cores SURFsara had in the summer of 2014) was 3 days per million articles with pipeline Version 2. For the second batch (using all 1,400 cores SURFsara had in the summer of 2015), the minimum processing time was 2.5 days for 1M articles using pipeline Version 3.

9 Conclusions

This deliverable described the third 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 third cycle of the NewsReader project (Event Detection, version 3) we focused on providing complete pipelines for English, Dutch, Spanish and Italian. Some of the modules have been adapted to the financial domain. We worked on new generic pipelines and modules for Dutch, Spanish and Italian. Table 55 presents the current state of the 40 available modules.

<table>
<thead>
<tr>
<th>Name</th>
<th>3rd year</th>
<th>Version 2nd year</th>
<th>1st year</th>
<th>Language(s)</th>
<th>Third-party software</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>ixa-pipe-tok</td>
<td>no update</td>
<td>1.5.0</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>All</td>
</tr>
<tr>
<td>Package</td>
<td>Type</td>
<td>Version</td>
<td>Status</td>
<td>Languages</td>
<td>Data Source</td>
<td>Tools/Projects</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>---------</td>
<td>--------</td>
<td>-----------</td>
<td>----------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>ixa-pipe-pos</td>
<td>update</td>
<td>1.4.6</td>
<td>✓</td>
<td>EN, ES</td>
<td></td>
<td>Apache OpenNLP project</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WSJ treebank, Anacora</td>
</tr>
<tr>
<td>ixa-pipe-parse</td>
<td>update</td>
<td>1.1.2</td>
<td>✓</td>
<td>EN, ES</td>
<td></td>
<td>Apache OpenNLP project</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Penn Treebank, Anacora</td>
</tr>
<tr>
<td>ixa-pipe-srl</td>
<td>update</td>
<td>1.1.0</td>
<td>✓</td>
<td>EN, ES</td>
<td></td>
<td>Apache OpenNLP project</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mate-tools</td>
</tr>
<tr>
<td>fbk-timepro</td>
<td>update</td>
<td>1.5</td>
<td>✓</td>
<td>EN</td>
<td></td>
<td>timenorm, YamCha, TinySVM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TempEval3 data</td>
</tr>
<tr>
<td>ixa-pipe-nerc</td>
<td>update</td>
<td>1.5.4</td>
<td>✓</td>
<td>All</td>
<td></td>
<td>Apache OpenNLP project</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CoNLL 2002, 2004, OntoNotes, Evalita07, Evalita09</td>
</tr>
<tr>
<td>wsd-ukb</td>
<td>update</td>
<td>2.0.22</td>
<td>✓</td>
<td>EN, ES</td>
<td></td>
<td>WordNet</td>
</tr>
<tr>
<td></td>
<td>new</td>
<td>1.0</td>
<td>✓</td>
<td>EN</td>
<td>IMS system</td>
<td>WordNet</td>
</tr>
<tr>
<td>it_makes_sense-WSD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>system</td>
</tr>
<tr>
<td>ixa-pipe-ned</td>
<td>update</td>
<td>1.1.5</td>
<td>✓</td>
<td>All</td>
<td></td>
<td>DBpedia spotlight</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBpedia spotlight server, Wikipedia</td>
</tr>
<tr>
<td>POCUS</td>
<td>new</td>
<td>1.0</td>
<td>✓</td>
<td>EN</td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td>dbpedia-ner</td>
<td>new</td>
<td>0.4</td>
<td>✓</td>
<td>All</td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>spotlight server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>spotlight server, Wikipedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Wikipedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ixa-pipe-wikify</td>
<td>new</td>
<td>1.3.0</td>
<td>✓</td>
<td>All</td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>spotlight</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>spotlight server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBpedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>spotlight server, Wikipedia</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Wikipedia</td>
</tr>
<tr>
<td>corefgraph</td>
<td>update</td>
<td>0.8</td>
<td>✓</td>
<td>EN, ES, NL</td>
<td></td>
<td>WordNet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>lmf</td>
</tr>
<tr>
<td>vua-eventcoreference</td>
<td>update</td>
<td>3.0</td>
<td>✓</td>
<td>EN, ES, NL</td>
<td></td>
<td>WordNet</td>
</tr>
</tbody>
</table>

NewsReader: ICT-316404        February 1, 2016
<table>
<thead>
<tr>
<th>Project</th>
<th>Version</th>
<th>Language(s)</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbk-temprel</td>
<td>update 2.5.0 ✓</td>
<td>EN, IT</td>
<td>Explicit Discourse Connectives Tagger, YamCha, TinySVM</td>
</tr>
<tr>
<td>fbk-causalrel</td>
<td>update 2.0.0 ✓</td>
<td>EN</td>
<td>Explicit Discourse Connectives Tagger, YamCha</td>
</tr>
<tr>
<td>vua-factuality</td>
<td>new 1.0 ✓ ✓</td>
<td>EN</td>
<td>FactBank TimBL</td>
</tr>
<tr>
<td>opinion-miner</td>
<td>no update 2.0 ✓ ✓</td>
<td>EN, NL</td>
<td>OverNeR Models</td>
</tr>
<tr>
<td>vua-alpino</td>
<td>update 1.0 ✓</td>
<td>NL</td>
<td>Alpino parser</td>
</tr>
<tr>
<td>vua-wsd</td>
<td>update 1.2 ✓</td>
<td>NL</td>
<td>ODWN</td>
</tr>
<tr>
<td>vua-ontotagging</td>
<td>update 1.0 ✓</td>
<td>NL</td>
<td>ODWN</td>
</tr>
<tr>
<td>vua-nominal-event-detection</td>
<td>update 1.0</td>
<td>NL</td>
<td></td>
</tr>
<tr>
<td>vua-srl</td>
<td>new 2.0 ✓</td>
<td>NL</td>
<td></td>
</tr>
<tr>
<td>vua-srl-dutch-nominal-events</td>
<td>update 1.0 ✓</td>
<td>NL</td>
<td></td>
</tr>
<tr>
<td>vua-srl-framenet-classifier</td>
<td>update 1.0 ✓</td>
<td>NL</td>
<td></td>
</tr>
<tr>
<td>ixa-heideltime</td>
<td>update 1.0.4 ✓</td>
<td>ES, NL</td>
<td>HeidelTime</td>
</tr>
<tr>
<td>fbk-tokenpro</td>
<td>no update 2.1 ✓ ✓</td>
<td>EN, IT</td>
<td></td>
</tr>
<tr>
<td>fbk-morphopro</td>
<td>no update 1.3.2-1 ✓ ✓</td>
<td>EN, IT</td>
<td></td>
</tr>
<tr>
<td>fbk-tagpro</td>
<td>update 2.0 ✓ ✓</td>
<td>IT</td>
<td>CRFsuite, Word form list, Wikipedia, Wikinews, Wikisource</td>
</tr>
<tr>
<td>fbk-lemmapro</td>
<td>no update 2.0 ✓ ✓</td>
<td>EN, IT</td>
<td>Yamcha, TinySVM</td>
</tr>
<tr>
<td>fbk-entitypro</td>
<td>no update 1.4.3 ✓ ✓</td>
<td>EN, IT</td>
<td>Yamcha, TinySVM</td>
</tr>
<tr>
<td>fbk-chunkpro</td>
<td>no update 2.0 ✓ ✓</td>
<td>EN, IT</td>
<td>Yamcha, TinySVM</td>
</tr>
</tbody>
</table>
Table 55: EN, IT, NL and ES modules.

In order to perform cross-lingual event detection, all the modules use NAF to harmonize their representations. They also produce semantically compatible information. The NewsReader project has thus developed a cross-lingual method for interpreting events and event components in text using a common language independent semantic representation.

This semantic interoperability is achieved by projecting entities, event mentions and time expressions to language independent knowledge representations. In particular, 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.
Matrix ([López de Lacalle et al., 2014](#)). Finally, time expressions are normalized following the ISO 24617-1 standard.

The pipelines that were created provide a rich semantic outcome, including disambiguated senses and named entities, events and event participants with interpretations from propBank, FrameNet and ESO, normalized time expressions and event coreference. For English and Italian, temporal and causal relations between events were also included. To our knowledge, these are the first pipelines that provide such rich semantic information for these languages. The benchmarks for English, Spanish, Dutch and Italian have shown that we obtain state-of-the-art (or above) results when evaluating our pipelines in standard datasets. When evaluating the same pipelines in the MEANTIME dataset the performance is lower compared to the ones obtained in the standard datasets. In fact, these type of results are coherent with previous assertions on out of domain evaluations. All four pipelines have been evaluated on the MEANTIME corpus. Results for English are generally best (which is expected), but the other languages come close to these results and, in some cases, even perform better than English. This result exceeded our expectation. The work described in this deliverable thus displays significant progress in event detection for English, Dutch, Spanish and Italian.

Finally, we have continued processing different datasets in order to test the scaling capabilities of our system and to prepare different Hackathons and end-user evaluations. Among others, we have processed approximately 2.5M articles in English articles and 900K Dutch documents.
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:

```xml
<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:

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

A.3 vua-wsd

The vua-wsd 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:

```xml
<term id="t9" lemma="man" morphofeat="NN" pos="N" type="open">
    <!--[CDATA[man]]></span>
    <target id="w9"/>
</term>
```
If we set the external references to be OpenDutchWordNet synsets instead, we would get this:

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

A.4 wsd-ukb and ims-wsd

The wsd-ukb module provides word sense disambiguation. It associates 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:

```xml
<term id="t15" type="open" lemma="presidente" pos="N" morphofeat="NCMS000">
  <span>
    <!--presidente--><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>
```

The it makes sense WSD system uses the same structure, but points to an alternative resource:

```xml
<term id="t12" lemma="option" morphofeat="NN" pos="N" type="open">
  <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>
```
Both representations can be added to the same term layer resulting in representations such as:

A.5 vua-ontotagger

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.

NewsReader: ICT-316404
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. 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 spanning terms. Example:

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

A.7 ixa-pipe-ned and POCUS

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 resource using the <externalRef> element. The <externalRef> has the resource, reference, source, reftype and confidence attributes. Example:

```
<entities>
  <entity id="e1" type="PERSON">
    ...
    <externalReferences>
      <externalRef resource="spotlight_v1" reference="http://dbpedia.org/resource/Mariano_Rajoy" confidence="1.0" reftype="en" source="en"/>
    </externalReferences>
  </entity>
</entities>
```

For Spanish, if the corresponding English dbpedia-entry is provided, the main <externalRef> element contains another <externalRef> nested to represent the English dbpedia-entry.

```
<entities>
  <entity id="e1" type="PERSON">
    ...
    <externalReferences>
      <externalRef resource="spotlight_v1" reference="http://es.dbpedia.org/resource/Mariano_Rajoy" confidence="0.9999531" reftype="es" source="es"/>
      <externalRef resource="wikipedia-db-esEn" reference="http://dbpedia.org/resource/Mariano_Rajoy" confidence="0.9999531" reftype="en" source="es"/>
    </externalReferences>
  </entity>
</entities>
```

112 The <term> elements are used to write the <span> elements in the output NAF document.
The POCUS module performs additional disambiguation on the found entries. It either adds an interpretation to an existing entity or adds new entities. The output of a file that’s been analyzed by both ixa-pipe-ned and POCUS looks as follows:

```xml
<entity id="e1" type="PERSON">
  <references>
    <span>
      <target id="t1" />
    </span>
  </references>
  <externalReferences>
    <externalRef confidence="1.0" reference="http://dbpedia.org/resource/Gary_Cahill" />
    <externalRef confidence="3.595268E-20" reference="http://dbpedia.org/resource/Darren_Cahill" />
  </externalReferences>
</entity>
```

A.8 dbpedia-ner

The dbpedia-ner module takes tokens and terms as input and aims to find named disambiguated entities. It creates new entities where the type value is taken from DBpedia or left empty if none is found. The sample below provides an illustration of both:

```xml
<entity id="e60" type="Schema:Product , DBpedia:MeanOfTransportation , DBpedia:Aircraft">
  <references>
    <span>
      <target id="t250" />
    </span>
  </references>
  <externalReferences>
    <externalRef confidence="1.0" reference="http://dbpedia.org/resource/Boeing_787_Dreamliner" />
  </externalReferences>
</entity>
```
A.9 ixa-pipe-wikify

The ixa-pipe-wikify provides wikification. It reads <wf> and <term> elements and it performs spotting and disambiguation of relevant terms. It annotates references to DBpedia on the <markables> element. For each relevant term, the module creates a <mark> element which has the source and the lemma attributes. It also links the terms to an external resource using the <externalRef> element. The <externalRef> has the resource, reference, confidence, source and reftype attributes. For example:

```xml
<!−− president−−>
<mark id="m2" source="DBpedia" lemma="president">  
  <span>  
    <target id="w5"/>  
  </span>  
</mark>
```

A.10 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. 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:

```xml
<constituency>
  <tree>
    <t id="ter1">  
      <!−− Mariano−−>  
    </t>  
    <span>
```

113 The <term> elements are used to write the <span> elements in the output NAF document.
### A.11 ixa-pipe-srl, vua-srl and nominal-events

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:

```xml
<dep from="t2" rfunc="NAME" to="t1"/>
<dep from="t3" rfunc="SBJ" to="t2"/>
<dep from="t7" rfunc="NAME" to="t4"/>
<dep from="t8" rfunc="NAME" to="t5"/>
<dep from="t22" rfunc="PRD" to="t15"/>
<dep from="t23" rfunc="PRD" to="t16"/>
<dep from="t14" rfunc="P" to="t8"/>
```

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.

The output of the modules that generate semantic roles for Dutch (vua-srl, vua-nominal-event-detection and vua-srl-nominal-events) follow the same structure.

```xml
<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>
</predicate>
```

A.12 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).

```xml
<sr1>
<predicate id="pr1" <!−− given by the propbank srl −−>
<!−− careen −−>
<externalReferences>
  <externalRef reference="Operate_vehicle" resource="FrameNet"/> <!−− to be added →
    by Chantal −−>
</externalReferences>
<span>
  <target id="t1.49"/>
</span>
</predicate>
```

NewsReader: ICT-316404  February 1, 2016
A.13 VUA-alpino

The VUA-alpino is a morphosyntactic analyzer that performs POS-tagging, constituent parsing and dependency 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.
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 attribute and the <span> element pointing to the term layer. Finally, the <edge> element has the id, from, to and head attributes.

Finally, in the dependency layer the dependencies found between the terms are represented 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:

```xml
<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.14 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:

```xml
<coref id="co1">
    <span>
        <target id="t4"/>
        <target id="t5"/>
        <target id="t6"/>
        <target id="t7"/>
        <target id="t8"/>
    </span>
    <target id="t15"/>
    <target id="t16"/>
    <target id="t17"/>
    <target id="t18"/>
    <target id="t19"/>
    <target id="t20"/>
    <target id="t21"/>
    <target id="t22"/>
    <target id="t23"/>
    <target id="t24"/>
</coref>
```
The event coreference module extends this layer further, applying the exact same structure for events:

A.15 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 spanning words. Example:

A.16 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. The <predicateAnchor> element is used to represent a relation between an event/predicate and the time it occurred. It has an id attribute and 3 other attributes to represent the time anchor: anchorTime, beginPoint and endTime. It also contains a <span> element spanning predicates. Example:

NewsReader: ICT-316404 February 1, 2016
A.17 fbk-eventpro

The EventPro module provides predicates. The <predicate> element is used to represent an event in the document. The <predicate> element has the id and contains <span> elements. The <span> element contains one or more <target> elements with the id attribute. Example:

```xml
<predicate id="pr34">
  !---effettuato--->
  <target id="t271"/>
</span>
</predicate>
```

A.18 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:

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

A.19 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:

```xml
<factvalue eventid="e1" certainty="CERTAIN" polarity="POS" time="NON-FUTURE" specialCases="NONE"/>
```
A.20 fbk-srl

The fbk-srl module provides semantic roles for Italian. For that, it uses the predicates already annotated and add their roles and their references to resources. Each annotate predicate is represented by an <predicate> element. The <predicate> has the <externalReferences>, <span> and <role> elements. 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> elements and their <span> sub-elements were created by the event detection module (A.17).

Example:

```
<sr1>
  <predicate id="pr34">
    <!−− effetruato−−>
    <externalReferences>
      <externalRef resource="EventType" reference="OCCURRENCE" />
      <externalRef resource="PropBank" reference="effettuare.01" />
      <externalRef resource="WordNet" reference="ili−39−01642924−v" />
    </externalReferences>
    <span>
      <target id="t271" />  
    </span>
    <role id="rl56" semRole="A0">
      <!−− Il primo volo del 777→−−>
      <span>
        <target id="t264" />
        <target id="t265" />
        <target id="t266" head="yes" />
        <target id="t267" />
        <target id="t268" />
      </span>
    </role>
    <role id="rl57" semRole="AM−TMP">
      <!−−1994→−−>
      <span>
        <target id="t273" />
      </span>
    </role>
  </predicate>
</sr1>
```

A.21 fbk-eventcoref

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

```
NewsReader: ICT-316404 February 1, 2016
```
A.22 vua-perspective-factuality

The factuality layer provides a flexible representation of factuality values. Each factuality object has a span and it can have one or more factVal elements that specify information concerning its factuality. The resource attribute allows information from alternative ontologies to be used. The current output typically provides a value from factBank, its corresponding NewsReader values and an additional NewsReader value for tense.

A.23 opinion-miner

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.
They had a nightmare with Hilton Hotel Paris. ---

The ixa-pipe-topic provides a set of thesaurus descriptors from the Multilingual Eurovoc thesaurus to describe the input document. For that, it reads the <raw> element and it outputs the <topics> elements that has the set of topics. Each <topic> element has the source, the method, the confidence and the value. For example:

- financial year
- EC
- appointment of staff

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 normalization.

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.
A.27 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.

<table>
<thead>
<tr>
<th>token</th>
<th>PoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>AT0</td>
</tr>
<tr>
<td>CEO</td>
<td>NP0</td>
</tr>
</tbody>
</table>

A.28 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.

<table>
<thead>
<tr>
<th>token</th>
<th>possible morph analyses</th>
<th>PoS</th>
<th>compatible morph analyses</th>
<th>lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>the+adv the+art</td>
<td>AT0</td>
<td>the+art</td>
<td>the</td>
</tr>
<tr>
<td>CEO</td>
<td>ceo+pn</td>
<td>NP0</td>
<td>ceo+pn</td>
<td>ceo</td>
</tr>
</tbody>
</table>

A.29 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.

NewsReader: ICT-316404            February 1, 2016
### A.30 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:

<table>
<thead>
<tr>
<th>token</th>
<th>POS</th>
<th>chunk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steve</td>
<td>NP0</td>
<td>B-NP</td>
</tr>
<tr>
<td>Jobs</td>
<td>NP0</td>
<td>I-NP</td>
</tr>
<tr>
<td>was</td>
<td>VBD</td>
<td>B-VP</td>
</tr>
</tbody>
</table>

### A.31 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.

<table>
<thead>
<tr>
<th>token</th>
<th>PoS</th>
<th>lemma</th>
<th>parser id</th>
<th>head id</th>
<th>dep relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steve</td>
<td>NP0</td>
<td>Steve</td>
<td>1</td>
<td>3</td>
<td>SUBJ</td>
</tr>
<tr>
<td>Jobs</td>
<td>NP0</td>
<td>Jobs</td>
<td>2</td>
<td>1</td>
<td>CONTIN-DENOM</td>
</tr>
<tr>
<td>was</td>
<td>VBD</td>
<td>be</td>
<td>3</td>
<td>0</td>
<td>ROOT</td>
</tr>
</tbody>
</table>
References


