XLike
FP7-ICT-2011.4.2
Contract no.: 288342
www.xlike.org

Deliverable D2.2.2

Final Deep Linguistic Processing Prototype

Editor:	Xavier Carreras, Universitat Politècnica de Catalunya
Author(s):	Xavier Carreras, Universitat Politècnica de Catalunya; Lluís Padró, Universitat Politècnica de Catalunya
Deliverable Nature:	Prototype (P)
Dissemination Level: (Confidentiality)	Public (PU)
Contractual Delivery Date:	M21
Actual Delivery Date:	M21
Suggested Readers:	All partners of the XLike project consortium and end-users
Version:	1.0
Keywords:	Linguistic analysis, natural language processing, dependency parsing, semantic role labeling, frame extraction

Disclaimer

Full Project Title:	XLike – Cross-lingual Knowledge Extraction
Short Project Title:	XLike
Number and Title of Work package:	WP2 – Multilingual Linguistic Processing
Document Title:	D2.2.2 – Final Deep Linguistic Processing Prototype
Editor (Name, Affiliation)	Xavier Carreras, UPC
Work package Leader (Name, affiliation)	Xavier Carreras, UPC
Estimation of PM spent on the deliverable:	13PM

Executive Summary

This document presents the final prototype that implements deep linguistic processing methods in XLike. Specifically we describe methods for syntactic analysis of language, and methods for analysis of predicate-argument semantic relations. In addition to linguistic analysis, we present a prototype for extracting semantic frames, a representation of linguistic relations between entities that builds on syntactic and semantic role analysis.

An early version of this prototype was described in D2.2.1 “Early deep linguistic processing prototype” at month 12. This final version completes the methods for predicting semantic roles, and develops a prototype for semantic frame extraction. In all, this deliverable is a comprehensive description of deep processing methods in XLike.

Table of Contents

List of Figures

List of Tables

Abbreviations

CoNLL	Conference on Computational Natural Language Learning (http://ifarm.nl/signll/conll)
D	Deliverable
NLP	Natural Language Processing
PoS	Part of Speech tag
SOA	Service Oriented Architecture
XML	eXtensible Markup Language
XLike	Cross-lingual Knowledge Extraction
WP	Work Package

Definitions

Frame	A schematic representation of a situation involving various participants.
Node	An element participating in a frame. Can be an entity, a word, or a frame.
Pipeline	Refers to the flux of different processes that are applied to a set of raw data in order to analyze it and interpret it. In NLP, a pipeline is a process that receives raw text and computes linguistic analysis, by a series of processes that perform morphological, syntactic and semantic analysis.
Treebank	A corpus of text documents in which each document is annotated with syntactic and semantic structures. It is used by machine learning methods in NLP in order to train statistical models of language analysis.

1 Introduction

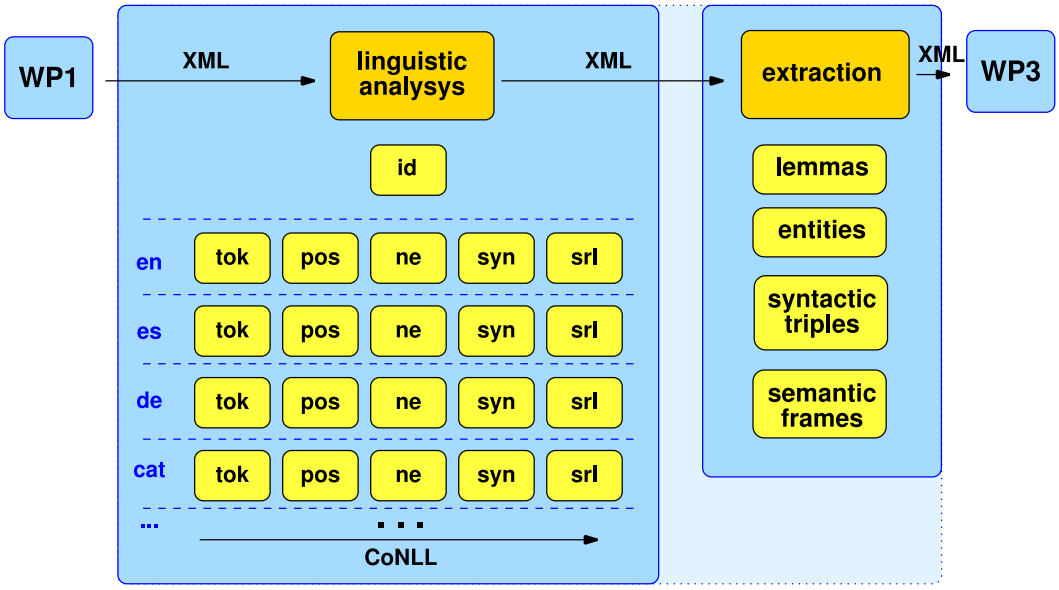

Figure 1 Architecture of Linguistic Processing Services in WP2

1.1 Linguistic Processing in XLike

The goal of WP2 within XLike is to develop methods to analyze documents and extract the entities that appear in the documents, together with their relations. The methods in WP2 should be able to analyze multiple languages ---in particular, the six target languages of the project (see Table 1 for the list of languages together with its code). In addition, methods in WP2 should be able to analyze documents of different domains, and of different levels of formality (from standard text we find in the news or Wikipedia, to informal language we find in forums and Twitter). Methodologically, Tasks 2.1 and 2.2 of WP2 focus on developing methods to perform multilingual analysis of standard texts of the news domain. The main effort behind this task is to put together a system that implements the state-of-the-art in NLP for multilingual analysis. The other aspects of WP2, namely adaptation of the linguistic processing methods to other domains and to informal texts, are the dealt in Tasks 2.3 and 2.4 of the WP.

Table 1 Language codes in XLike based on the ISO 639-1 Standard.

Language	*Code*	*Language*	*Code*
English	en	**Catalan**	ca
Spanish	es	**Slovene**	sl
German	de	**Chinese**	zh

Figure 1 gives an overview of the architecture of WP2 methods. WP2 itself can be seen as a toolbox that receives free-text documents from WP1, annotates these documents with linguistic structure, extracts target linguistic patterns, and passes this structured document to WP3. The structured documents are represented in XML. Within WP2, we distinguish between three types of linguistic processing methods:

- **Shallow linguistic analysis:** Annotate sentences with flat linguistic structure, such as the language code of the document, morpho-syntactic information for tokens, and named entities.

- **Deep linguistic analysis:** Annotate sentences with hierarchical linguistic structure, such as syntactic relations and semantic roles.
- **Extraction:** Extract target elements from the linguistic structure computed by the previous methods, such as lemmas, entities, syntactic triples, or semantic frames.

In WP2, the methods are organized in a pipeline: the first methods of the pipeline annotate sentences with basic linguistic structure, which is required by the subsequent methods of the pipeline. These linguistic annotations are represented using the CoNLL format.

In this deliverable we describe methods for deep linguistic analysis, as well as methods to extract semantic frames using the linguistic analysis provided by the deep analysis.

2 Deep Linguistic Processing

In this section we describe the methods for deep linguistic processing. We exploit three types of methods: syntactic parsing, semantic role labeling and frame extraction. Here is an illustration of the deep linguistic structures that we use in XLike:

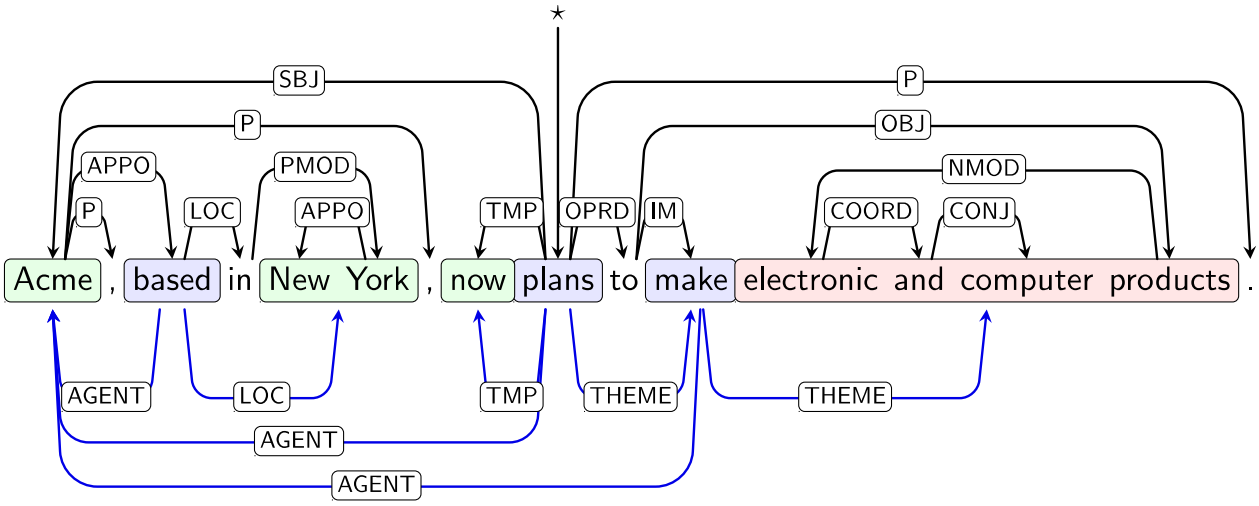

Figure 2 Example of a sentence annotated with a dependency tree (structure above, in black) and predicate argument structures (below, in blue)

2.1 Syntactic Parsing

Syntactic parsing is the process of recovering the syntactic structure of a sentence. Over the last decade, statistical natural language approaches to parsing have greatly advanced, in part because of the progress in statistical and machine learning methods in NLP, in part because of the availability of available treebanks, which are required in any supervised machine learning technique.

Dependency representations of syntax have become especially popular in statistical approaches to parsing, and are the type of representation we choose for the syntactic parsers in XLike. A syntactic dependency is a direct relation between two words that provides a simple, intuitive representation of syntactic functions. Figure 2 shows an English sentence together with a syntactic dependency tree (top), where dependencies are labeled with syntactic functions.

Computationally, dependency methods have efficient parsing algorithms to calculate the best dependency tree of a sentence [Eis00,MCP05]. Having efficient parsing algorithms has been key in order to develop machine learning techniques that learn a syntactic parser using training data [MCP05,NN05,MP06]. Furthermore, since a syntactic dependency is a relation between words, it facilitates very much the incorporation of lexical features into the statistical method, which is crucial in all disambiguation tasks.

In XLike, we use the so-called graph-based methods for dependency parsing, introduced in [MCP05]. In particular we use the following tools:

- **Treeler:** This is a library developed by the UPC team that implements several methods for dependency parsing, as well as other statistical methods for tagging and parsing. For dependency

parsing, the implementation is based on [Car07,KCC08,CCK08], which in turn is based on the ideas by [MCP05]. The library is available at http://treeler.lsi.upc.edu.

- **MSTParser:** This is the implementation provided by the authors of [MCP05,MC06]. For the Chinese language, the THU group uses this implementation. The code of the parser is available at: http://sourceforge.net/projects/mstparser.

We use these tools in order to train dependency parsers for all XLike that then we integrate in the XLike linguistic pipelines. Table 2 summarizes the treebanks used to develop the parsers. In some cases the treebanks are not in dependency format, and we use available tools that make that conversion. The third column of Table 2 indicates the method used to convert to dependencies, while the fourth column indicates the number of dependency relations of the dependency representation.

Table 2 Treebanks used to develop XLike syntactic parsers

Language	*TreeBank*	*conversion to dependencies*	*number of dependency relations*
en	Penn Treebank [PTB]	CoNLL-09 [HCJ+09]	69
es	Ancora [Anc]	Ancora	49
de	Tiger [Tiger]	CoNLL-09 [HCJ+09]	46
ca	Ancora [Anc]	Ancora	50
sl	Učni [Ucni]	Učni	10
zh	CSDN [CSDN]	CSDN	48

2.2 Semantic Role Labeling

Semantic role labeling (SRL) is the task of identifying the arguments of lexical predicates in a sentence and labeling them with semantic roles [GJ02; MCL+08]. SRL is an important shallow semantic task in NLP since predicate-argument relations directly represent semantic properties of the type "who" did "what" to "whom", "how", and "why" for events expressed by lexical items (typically verbs and nouns). Figure 2 shows an example, where the dependencies below the sentence correspond to predicate-argument relations labeled with semantic roles. In that sentence there are three predicates, namely "based", "plans" and "makes". The dependency between "based" and "Acme" indicates that "Acme" is the argument of the predicate "base" with semantic role AGENT. Semantic roles represent an abstraction of the syntactic form of a predicative event. While syntactic functions of arguments change with the form of the event (e.g., active vs. passive forms), the semantic roles of arguments remain invariant to their syntactic realization.

Predicate-argument relations are strongly related to the syntactic structure of the sentence. Thus, a method that predicts predicate-argument relations is run after a syntactic parser, and uses the output of the syntactic parser as the backbone structure in order to recognize semantic relations. The model consists of two types of classifiers. The first type decides whether a word is a predicate or not, and optionally output a semantic sense of the predicate. The second type of classifiers considers argument candidates of a predicate (by looking at words syntactically connected to the predicate) and decides the the semantic role of the candidate. This approach was introduced by [GJ02], and there has been a great body of work, together with evaluations of systems like those of the CoNLL-2009 shared task [HCJ+].

As with syntactic parsing, we are developing SRL methods with the Treeler library. In order to train models, we will use the treebanks made available by the CoNLL-2009 shared task, which provided data annotated with predicate-argument relations for English, Spanish, Catalan, German and Chinese. No treebank annotated with semantic roles exists for Slovene.

A prototype of SRL has been integrated in all pipelines (except the Slovene pipeline). The method implemented follows a pipeline architecture described in [LCM13].

2.3 Frame Extraction

In the previous deliverable on deep processing methods [D2.2.1] we described a tool that extracts syntactic triple relations of the form subject-verb-object. Our tool relies on a set of manually-developed extraction rules that look for subject-verb-object patterns in the syntactic dependency tree. The advantage of the tool is that it is relatively simple to write a extraction rules that recognize a number of simple but frequent patterns. These rules, however, are treebank-dependant, and it is generally hard to write rules that capture more complex patterns.

In this section we present an alternative method to extract relations that directly leverages the linguistic analysis based on semantic roles. The method is based on the notion of frames: a *semantic frame* is a schematic representation of a situation involving various participants. In a frame, each participant plays a *role.* There is a direct correspondence between frames and semantic roles; namely, frames correspond to predicates, and participants correspond to the arguments of the predicate.

We distinguish three types of participants: entities, words, and frames. Looking again at the example sentence in Figure 2, there are three frames in the sentence:

- **Base**, as an person or organization being established or grounded somewhere
    - “Acme” is a participant of type entity that plays the role of agent
    - “New York” is a participant of type entity that plays the role of location
- **Plan**, as a person or organization planning some activity
    - “Acme” is a participant of type entity that plays the role of agent
    - “now” is a participant of type word that plays the role of temporal modifier
    - “make” is a participant of type frame that plays the role of theme, i.e. the activity
- **Make**, as a person or organization creating or producing something
    - “Acme” is a participant of type entity that plays the role of agent
    - “products” is a participant of type word that plays the role of theme, i.e. the thing being created

It is important to note that frames are a more general representation than triples. While triples represent a binary relation between two participants, frames can represent any n-ary relation. For example, the frame for “plan” is a ternary relation because it includes a temporal modifier. It is also important to note that frames can naturally represent higher-order relations: the theme of the frame “plan” is itself a frame, namely “make”.

The frame extraction method consists of the following steps:

1. Obtain the semantic role analysis of a sentence.

2. For each predicate, create a frame, and for each argument of the predicate, create a node representing a participant of the frame. Nodes can either be entities (if the argument has been classified as a named entity), frames (if the argument is a predicate), or words (if the argument is neither an entity or a predicate).
3. Using the WordNet sense of a predicate, compute semantic labels for the participants. Typically, semantic role parsers such as ours produce positional labels (such as argument 0, argument 1, and so on), instead of semantic roles (such as agent, theme, etc.). We use the VerbIndex resource [VN] in order to map positional arguments into semantic roles. The resource can be thought of a table that given the WordNet sense of the predicate provides a mapping from positional arguments to semantic roles. This is an excerpt of the table for the predicates in our example sentence:

Predicate Sense	Argument Mapping		
00636888-v (base, establish, ground)	A0: AGENT	A1: LOC	A2: SOURCE
00704690-v (plan)	A0: AGENT	A1: THEME	
01617192-v (make, create)	A0: AGENT	A1: THEME	A2: PREDICATE

Figure 3 Mapping from positional arguments to semantic roles, based on VerbNet and WordNet

3 Implementation

The software architecture of XLike follows a SOA (Service Oriented Approach), in which functionalities are offered in a decentralized manner using RESTful Web Services. See XLike Deliverable 6.2.1 [D6.2.1] for a detailed description of this architecture. The linguisic services of WP2 naturally follow this approach.

In fact, the specification of the architecture of linguistic services was given in WP2 Deliverable 2.1.1 on Shallow Linguistic methods [D2.1.1], which already contemplated how services for deep linguistic analysis would be integrated. A sketch of these linguistic services is in Figure 1. In particular:

- For each of the languages we offer a service that implements a pipeline consisting of all the linguistic modules for that language. The input/output of this service is in XML format.
- Within the pipeline, each module receives input and generates output in CoNLL format.

To complement this architecture, deep linguistic services are offered by UPC as stand-alone services. For example, UPC offers specific services for syntactic parsing for German and Slovene, and services for semantic parsing for German and Chinese: these services require a sentence analyzed with pos tags, and return syntactic / semantic dependency structures. Then, the *pipeline service* for German analyzes a sentence as follows: first it applies tokenization and PoS tagging methods to the sentence, then it calls the UPC service for syntactic and semantic analysis, and finally it puts all results in the XML structure and returns it.

The specification of linguistic services of UPC is the following:

Table 3 Specification of specific UPC services for deep linguistic processing

Description	*Service*	*Input Parameters*	*Output Parameters*
Dependency Parsing	base_url/upc_linguistic_services/**parse**	**lang:** language code **conll:** the sentence in CoNLL Format (with PoS tags)	**conll:** the sentence in CoNLL format with a dependency tree
Semantic Role Labeling	base_url/upc_linguistic_services/**srl**	**lang:** language code **conll:** the sentence in CoNLL Format (with PoS tags and syntactic dependencies)	**conll:** the sentence in CoNLL format with predicate-argument structures
Frame Extraction	base_url/upc_linguistic_services/**frames**	**lang:** language code **conll:** the sentence in CoNLL Format, fully parsed	An XML document representing semantic frames found in the document

Following the design guidelines of the XLike toolkit, the frames are respresented using XML. Appendix B includes the scheme that specifies this representation. Appendix A gives an example of the example sentence in Figure 2 represented in XML.

4 Conclusion

This deliverable presents deep linguistic processing methods in XLike. The final representation of documents is based on frame semantics, an expressive representation of syntactic-semantic relations between predicates and entities. The prototypes are based on statistical methods for syntactic-semantic analysis of language. One advantage of this paradigm is that the underlying techniques are largely language-independent, and therefore we can obtain prototypes for different languages as long as there exist treebanks with syntactic annotations.

Frame semantics provides an abstract representation of textual semantics between entities. The formalism is expressive, in the sense that it allows to capture agent-predicate-theme relations, but also adjunct participants that encode temporal, location, causative and other semantic relations. In also supports encoding relations between frames. In all, it seems an appropriate formalism for knowledge extraction.

There are two main goals to pursue from this point. The first is to analyse the performance of the linguistic prototypes for XLike use cases, both in accuracy and speed. In general, moving from the semantic domains reflected in the development corpora (mostly newswire) to new domains (wikipedia, financial and legal reports, specialized blogs) will cause a drop in performance, due to the difference in word distributions, linguistic constructs and semantic concepts. Domain adaptation methods will be required to gather lexical statistics of the new domain, and combine them with the existing linguistic models.

The second goal concerns adapting the tools to process informal language. To some extent, this is an extreme of the domain shift, with the added challenge that basic linguistic constructs can drastically change when moving from formal to informal text. In all, we believe that the core set of statistical parsing tools used to process standard language, and included in this prototype, will constitute the backbone of the pipelines for informal language processing. Domain adaptation methods, combined with weakly supervision strategies, will constitute the major lines of work in order to adapt the linguistic prototypes to robust processing of informal texts.

References

[D2.1.1] XLike deliverable *"D2.1.1 – Shallow linguistic processing prototype"*

[D2.2.1] XLike deliverable *"D2.2.1 – Early deep linguistic processing prototype"*

[D6.2.1] XLike deliverable *"D6.2.1 – Early Prototype"*

[Anc] Mariona Taulé, Maria Antonia Marti, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-2008)*, Marrakesh, Morroco.

[Car07] Xavier Carreras. Experiments with a Higher-Order Pro jective Dependency Parser. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL, pages 957–961. Association for Computational Linguistics, 2007.

[CCK08] Xavier Carreras, Michael Collins, and Terry Koo. TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing. In Proceedings of the 12th CoNLL, pages 9–16. Association for Computational Linguistics, 2008.

[CSDN] The Chinese CSDN corpus.

[Eis00] Jason Eisner. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. In Harry Bunt and Anton Nijholt, editors, Advances in Probabilistic and Other Parsing Technologies, pages 29–62. Kluwer Academic Publishers, 2000.

[GJ02] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. Computational Linguistics, 28(3):245–288, September.

[GKD12] Miha Grčar, Simon Krek, Kaja Dobrovoljc (2012): Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. V T. Erjavec, J. Žganec Gros (ur.): Zbornik Osme konference Jezikovnetehnologije. Ljubljana: Institut Jožef Stefan.

[HCJ+09] Jan Hajič; Massimiliano Ciaramita; Richard Johansson; Daisuke Kawahara; Maria Antònia Martí; Lluís Màrquez; Adam Meyers; Joakim Nivre; Sebastian Padó; Jan Štepánek; Pavel Straňák; Mihai Surdeanu; Nianwen Xue; Yi Zhang.*The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages.* In CoNLL 2009.

[KCC08] Terry Koo, Xavier Carreras, and Michael Collins. Simple Semi-supervised Dependency Parsing. In Proceedings of the 46th ACL, pages 595–603. Association for Computational Linguistics, 2008.

[LCM12] Xavier Lluís, Xavier Carreras, and Lluís Màrquez. Joint Arc-factored Parsing of Syntactic and Semantic Dependencies. Transactions of the Association for Computational Linguistics (TACL), volume 1, pages 219-230. 2013.

[MCL+08] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Label- ing: An Introduction to the Special Issue. Computa- tional Linguistics, 34(2):145–159, June.

[MCP05] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online Large-Margin Training of Dependency Parsers. In Proceedings of the 43rd ACL, pages 91–98. Association for Computational Linguistics, 2005.

[MMM06] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. "Generating Typed Dependency Parses from Phrase Structure Parses." In *Proceedings of LREC-06*.

[MP06] Ryan McDonald and Fernando Pereira. Online Learning of Approximate Dependency Parsing Algorithms. In Proceedings of the 11th EACL, pages 81–88. Association for Computational Linguistics, 2006.

[NN05] Joakim Nivre and Jens Nilsson. Pseudo-Projective Dependency Parsing. In Proceedings of the 43rd ACL, pages 99–106. Association for Computational Linguistics, 2005.

[NHK+07] Joakim Nivre, Johan Hall, Sandra Kubler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL, pages 915–932. Association for Computational Linguistics, 2007.

[PTB] Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn TreeBank. Computational Linguistics, 19.

[SJM+08] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluıs Marquez, and Joakim Nivre. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In Proceedings of the 12th CoNLL, 2008.

[Tiger] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER tree-bank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

[TM03] EF Tjong Kim Sang, F De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of CoNLL-2003.

[Ucni] Ucni corpus, http://www.slovenscina.eu/tehnologije/ucni-korpus

[VN] Unified Verb Index, including VerbNet, http://verbs.colorado.edu/verb-index

Annex A Example of an annotated document in XML

```
<?xml version="1.0"?>
<item>
  <services>
    <service name="UPC-analysis" date="2013-10-04"/>
  </services>
  <sentences>
    <sentence id="1">
      <text>Acme, based in New York, now plans to make electronic and computer products.</text>
      <tokens>
        <token pos="NN" end="4" lemma="acme" id="1.1" start="0">Acme</token>
        <token pos="Fc" end="5" lemma="," id="1.2" start="4">,</token>
        <token pos="VBN" end="11" lemma="base" id="1.3" start="6">based</token>
        <token pos="IN" end="14" lemma="in" id="1.4" start="12">in</token>
        <token pos="NP00G00" end="23" lemma="new_york" id="1.5" start="15">New_York</token>
        <token pos="Fc" end="24" lemma="," id="1.6" start="23">,</token>
        <token pos="RB" end="28" lemma="now" id="1.7" start="25">now</token>
        <token pos="VBZ" end="34" lemma="plan" id="1.8" start="29">plans</token>
        <token pos="TO" end="37" lemma="to" id="1.9" start="35">to</token>
        <token pos="VB" end="42" lemma="make" id="1.10" start="38">make</token>
        <token pos="JJ" end="53" lemma="electronic" id="1.11" start="43">electronic</token>
        <token pos="CC" end="57" lemma="and" id="1.12" start="54">and</token>
        <token pos="NN" end="66" lemma="computer" id="1.13" start="58">computer</token>
        <token pos="NNS" end="75" lemma="product" id="1.14" start="67">products</token>
        <token pos="Fp" end="76" lemma="." id="1.15" start="75">.</token>
      </tokens>
    </sentence>
  </sentences>
  <nodes>
    <node type="entity" class="organization" displayName="acme" id="E1">
      <mentions>
        <mention sentenceId="1" id="W5.1" words="Acme">
          <mention_token id="1.1"/>
        </mention>
      </mentions>
      <descriptions>
        <description URI="08677801-n" displayName="acme,apex,peak,vertex" knowledgeBase="WordNet-3.0"/>
        <description URI="&amp;amp;%Region+" knowledgeBase="SUMO"/>
      </descriptions>
    </node>
    <node type="entity" class="location" displayName="new_york" id="E2">
      <mentions>
        <mention sentenceId="1" id="E1.1" words="New York">
          <mention_token id="1.5"/>
        </mention>
      </mentions>
    </node>
    <node type="word" displayName="now" id="W6">
      <mentions>
        <mention sentenceId="1" id="W6.1" words="now">
          <mention_token id="1.7"/>
        </mention>
      </mentions>
      <descriptions>
        <description URI="00048739-r" displayName="at_once,directly,forthwith,immediately,instantly,like_a_shot,now,right_away,straightaway,straight_off"
knowledgeBase="WordNet-3.0"/>
        <description URI="&amp;amp;%SubjectiveAssessmentAttribute+" knowledgeBase="SUMO"/>
      </descriptions>
    </node>
    <node type="word" displayName="product" id="W7">
      <mentions>
        <mention sentenceId="1" id="W7.1" words="products">
          <mention_token id="1.14"/>
        </mention>
      </mentions>
      <descriptions>
        <description URI="04007894-n" displayName="product,production" knowledgeBase="WordNet-3.0"/>
        <description URI="&amp;amp;%Manufacture=" knowledgeBase="SUMO"/>
        <description URI="Mx4rvVkFYpwpEbGdrcN5Y29ycA" knowledgeBase="OpenCYC"/>
      </descriptions>
    </node>
  </nodes>
  <frames>
    <frame sentenceId="1" displayName="base.01" id="F2" tokenId="1.3">
      <argument role="A1:Theme" displayName="acme" id="E1"/>
      <argument role="AM-LOC" displayName="new_york" id="E2"/>
      <descriptions>
        <description URI="00636888-v" displayName="base,establish,found,ground" knowledgeBase="WordNet-3.0"/>
      </descriptions>
    </frame>
    <frame sentenceId="1" displayName="plan.01" id="F3" tokenId="1.8">
      <argument role="A0" displayName="acme" id="E1"/>
      <argument role="AM-TMP" displayName="now" id="W6"/>
      <argument role="A1" displayName="make.01" frame="true" id="F4"/>
      <descriptions>
        <description URI="00704690-v" displayName="plan" knowledgeBase="WordNet-3.0"/>
```

```
    </descriptions>
   </frame>
   <frame sentenceId="1" displayName="make.01" id="F4" tokenId="1.10">
    <argument role="A0:Cause" displayName="acme" id="E1"/>
    <argument role="A1:Theme" displayName="product" id="W7"/>
    <descriptions>
     <description URI="01617192-v" displayName="create,make" knowledgeBase="WordNet-3.0"/>
    </descriptions>
   </frame>
  </frames>
  <conll>
1 Acme acme NN cpos=NN B-ORG 8 SBJ 08677801-n _ A1 A0 A0
2 , , Fc cpos=, _ 1 P _ _ _ _ _
3 based base VBN cpos=VBN _ 1 APPO 00636888-v base.01 _ _ _
4 in in IN cpos=IN _ 3 LOC _ _ AM-LOC _ _
5 New_York new_york NP00G00 cpos=NNP B-LOC 4 PMOD 09119277-n _ _ _ _
6 , , Fc cpos=, _ 1 P _ _ _ _ _
7 now now RB cpos=RB _ 8 TMP 00048739-r _ _ AM-TMP _
8 plans plan VBZ cpos=VBZ _ 0 ROOT 00704690-v plan.01 _ _ _
9 to to TO cpos=TO _ 8 OPRD _ _ _ A1 _
10 make make VB cpos=VB _ 9 IM 01617192-v make.01 _ _ _
11 electronic electronic JJ cpos=JJ _ 14 NMOD 02718497-a _ _ _ _
12 and and CC cpos=CC _ 11 COORD _ _ _ _ _
13 computer computer NN cpos=NN _ 12 CONJ 03082979-n _ _ _ _
14 products product NNS cpos=NNS _ 10 OBJ 04007894-n _ _ _ A1
15 . . Fp cpos=. _ 8 P _ _ _ _ _

</conll>
</item>
```

Annex B XML Schema for XLike Annotated Documents

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema" >

 <xs:element name="item" type="item" />

 <!-- Element 'item' contains information about a news item. -->
 <!--    sentences : analyzed sentences of the item.       -->
 <!--    nodes : nodes of the semantic graph.              -->
 <!--    frames : frames of the semantic graph.            -->
 <!--    conll : analyzer output in CoNLL format.          -->
 <xs:complexType name="item">
  <xs:sequence>
   <xs:element name="services" type="serviceList" minOccurs="0" maxOccurs="1" />
   <xs:element name="sentences" type="sentenceList" minOccurs="0" maxOccurs="1" />
   <xs:element name="nodes" type="nodeList" minOccurs="0" maxOccurs="1" />
   <xs:element name="frames" type="frameList" minOccurs="0" maxOccurs="1" />
   <xs:element name="conll" minOccurs="0" maxOccurs="1" type="xs:string" />
  </xs:sequence>
 </xs:complexType>

 <!-- List of services used to process the item.-->
 <xs:complexType name="serviceList">
          <xs:sequence>
            <xs:element name="service" type="service" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
 </xs:complexType>

 <!-- Service applied to processs the item     -->
 <!-- name : name of the executed service      -->
 <!-- date : when the service was execute      -->
 <!-- duration: how long the execution takes   -->
 <xs:complexType name="service">
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="date" type="xs:date"/>
          <xs:attribute name="duration" type="xs:duration" />
 </xs:complexType>

 <!-- List of sentences in the item.-->
 <xs:complexType name="sentenceList">
          <xs:sequence>
            <xs:element name="sentence" type="sentence" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
 </xs:complexType>

 <!-- Linguistic analysis for a sentence.-->
 <!--    id : sentence id (numbered from 1 for each news item).       -->
 <!--    text : original text of the sentence.                        -->
 <!--    tokens : list of tokens in the sentence.                     -->
 <xs:complexType name="sentence">
  <xs:sequence>
   <xs:element name="text" />
   <xs:element name="tokens" type="tokenList" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string"/>
 </xs:complexType>

 <!-- List of tokens in a sentence -->
 <xs:complexType name="tokenList">
           <xs:sequence>
             <xs:element name="token" type="token" minOccurs="0" maxOccurs="unbounded" />
           </xs:sequence>
 </xs:complexType>

 <!-- linguistic information for a token -->
 <!--    id : token id. Numbered from 1 for each sentence.         -->
 <!--       e.g. token 3 of sentence 2 has id="2.3".               -->
 <!--    pos : token Part of Speech tag.                           -->
 <!--    lemma : token lemma.                                      -->
 <!--    start : token start character position in original text.  -->
 <!--    end   : token end character position in original text.    -->
 <xs:complexType name="token" mixed="true">
  <xs:attribute name="id" type="xs:string" />
  <xs:attribute name="pos" type="xs:string" />
  <xs:attribute name="lemma" type="xs:string" />
  <xs:attribute name="start" type="xs:int" />
  <xs:attribute name="end" type="xs:int" />
 </xs:complexType>

 <!-- List of nodes in the item -->
 <xs:complexType name="nodeList">
           <xs:sequence>
             <xs:element name="node" type="node" minOccurs="0" maxOccurs="unbounded" />
           </xs:sequence>
 </xs:complexType>
```

```
<!-- List of frames in the item -->
<xs:complexType name="frameList">
            <xs:sequence>
              <xs:element name="frame" type="frame" minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
</xs:complexType>

<!-- A node of the semantic graph. May correspond to -->
<!-- a named entity or to an ordinay noun phrase.    -->
<!--     id : Node id.  It is prefixed by "E" for entities and by "W" for words      -->
<!--     displayName: string identifying the node to humans (e.g. lemma).            -->
<!--     type : "entity" if the node is an entity, or "word" if it is a regular word -->
<!--     class : If the node type is "entity", semantic class (person, location..)   -->
<!--     lang : language                                                             -->
<!--     weight: weight                                                              -->
<!--     mentions : list of mentions to this node in the text.                       -->
<!--     descriptions : list of ontological descrpintions for this node.             -->
<xs:complexType name="node">
  <xs:sequence>
    <xs:element name="mentions" type="mentionList" />
    <xs:element name="descriptions" type="descriptionList" minOccurs="0" maxOccurs="unbounded"  />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" />
  <xs:attribute name="displayName" type="xs:string" />
  <xs:attribute name="type" type="xs:string" />
  <xs:attribute name="class" type="xs:string" />
  <xs:attribute name="lang" type="xs:string" />
</xs:complexType>

<!-- List of mentions of an entity -->
<xs:complexType name="mentionList">
            <xs:sequence>
              <xs:element name="mention" type="mention" minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
</xs:complexType>

<!-- List of ontological description for an entity -->
<xs:complexType name="descriptionList">
<xs:sequence>
  <xs:element name="description" type="description" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>

<!-- A mention to one particular named entity -->
<!--     mention_token : token id for each word forming this                     -->
<!--     id : mention id (entity id + mention number. e.g.: "E1.1", "E1.2" etc) -->
<!--     sentenceId : Sentence where the mention occurred.                       -->
<!--     words: Surface form of mention as occurred.                             -->
<xs:complexType name="mention">
  <xs:sequence>
    <xs:element name="mention_token" type="mention_token" minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="sentenceId" type="xs:string"/>
  <xs:attribute name="words" type="xs:string" />
</xs:complexType>

<!-- A token of a mention of a Named Entity -->
<!--     id : token id for the mention token -->
<xs:complexType name="mention_token">
  <xs:attribute name="id" />
</xs:complexType>

<!-- Link to one ontology (Wikipedia, WordNet, CYC, ...) for a given node   -->
<!--     knowledgeBASE : ontological resource (WN, Wikipedia, etc).         -->
<!--     URI : locator of the resource in the KB                            -->
<!--     displayName: string identifying the node to humans (e.g. lemma).   -->
<!--     lang : language                                                    -->
<xs:complexType name="description">
  <xs:attribute name="knowledgeBase" type="xs:string" />
  <xs:attribute name="URI" type="xs:string" />
  <xs:attribute name="displayName" type="xs:string" />
  <xs:attribute name="lang" type="xs:string" />
  <xs:attribute name="confidence" type="xs:decimal" />
</xs:complexType>

<!-- A frame of the semantic graph. Corresponds to an event                  -->
<!--     id : Frame id.  It is prefixed by "F" (e.g. F1, F2, F3...).         -->
<!--     tokenId : token id for this mention                                 -->
<!--     sentenceId : Sentence where the mention occurred.                   -->
<!--     displayName : string identifying the frame to humans (e.g. lemma).  -->
<!--     type : If the node is an entity, semantic type (person, location..) -->
<!--     argument : arguments of the predicate                               -->
<!--     description : ontological descriptions for this frame.              -->
<xs:complexType name="frame">
  <xs:sequence>
    <xs:element name="argument" type="argument" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="descriptions" type="descriptionList" minOccurs="0" maxOccurs="1" />
```

```
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="tokenId" type="xs:string" />
    <xs:attribute name="sentenceId" type="xs:string" />
    <xs:attribute name="displayName" type="xs:string" />
  </xs:complexType>

  <!-- A frame argument. May refer to a node, or to another frame.             -->
  <!--     role : Role played by this argument (subject, object, etc).        -->
  <!--     id : node or frame id of argument playing this role.               -->
  <!--     displayName: string identifying the argument to humans.            -->
  <!--     frame : true if the argument is another frame, false otherwise.    -->
  <!--     description : ontological descriptions for this argument.          -->
  <xs:complexType name="argument">
    <xs:sequence>
      <xs:element name="description" type="description" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="role" type="xs:string" />
    <xs:attribute name="id" type="xs:string" />
    <xs:attribute name="displayName" type="xs:string" />
    <xs:attribute name="frame" type="xs:boolean" />
  </xs:complexType>

</xs:schema>
```