Deliverable D2.2.1

Early Deep Linguistic Processing Prototype

Editor:	Xavier Carreras, UPC
Author(s):	Xavier Carreras (UPC), Lluís Padró (UPC), Xavier Lluís (UPC), Ariadna Quattoni (UPC)
Deliverable Nature:	Prototype (P)
Dissemination Level: (Confidentiality)[1]	Public (PU)
Contractual Delivery Date:	M12
Actual Delivery Date:	M12
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

[1] Please indicate the dissemination level using one of the following codes:
• **PU** = Public • **PP** = Restricted to other programme participants (including the Commission Services) • **RE** = Restricted to a group specified by the consortium (including the Commission Services) • **CO** = Confidential, only for members of the consortium (including the Commission Services) • **Restreint UE** = Classified with the classification level "Restreint UE" according to Commission Decision 2001/844 and amendments • **Confidentiel UE** = Classified with the mention of the classification level "Confidentiel UE" according to Commission Decision 2001/844 and amendments • **Secret UE** = Classified with the mention of the classification level "Secret UE" according to Commission Decision 2001/844 and amendments

Full Project Title:	Cross-lingual Knowledge Extraction
Short Project Title:	XLike
Number and Title of Work package:	WP2 – Multilingual Linguistic Processing
Document Title:	D2.2.1 – Early 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:	12 PM

Executive Summary

This document presents the early prototype that implements deep linguistic processing methods. Specifically we describe methods for syntactic analysis of language, and methods for analysis of predicate-argument semantic relations. In addition to linguistic analysis, we describe how syntactic and semantic linguistic structures can be exploited in order to extract relations.

This document is the second of three (D2.1.1 Y1, D2.2.1 Y1, and D2.2.2 Y2) that are associated with developing tools for linguistic analysis of standard texts. The implementation of deep methods follows the software architecture that was presented in D2.1.1. Here, to complement D2.1.1, we present an evaluation of the methods presented there, together with an evaluation of the early prototypes on benchmark tests. Overall, year 1 has been successful in putting together six multilingual pipelines for linguistic analysis, which run in a flexible and distributed architecture. Evaluations show that the performance of our implementations is slightly below the state-of-the-art. During year 2 we will analyze the causes for this, and make improvements to meet state-of-the-art accuracies. This is compatible with the general goals of WP2 for adapting linguistic analysis tools to informal text.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
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

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

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, or named entities.
- **Deep linguistic analysis:** Annotate sentences with hierarchical linguistic structure, such as syntactic trees.
- **Extraction:** Extract target elements from the linguistic structure computed by the previous methods, such as entities or relations.

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, which will be described later. Next we give an overview of each set of methods.

1.2 Shallow linguistic analysis

The first effort in WP2 was to provide methods for shallow linguistic processing for all languages. This was the main effort until M6 of the project. The deliverable D2.1.1 documents this set of tools. In essence, the tools are the following:

- **Language Identification (id):** Identifies the language of the document and returns the language code.
- **Tokenization (tok):** Segments and tokenizes the input document. That is, the input is a document free text, and the output is a structure that identifies the sentences and tokens of the document.

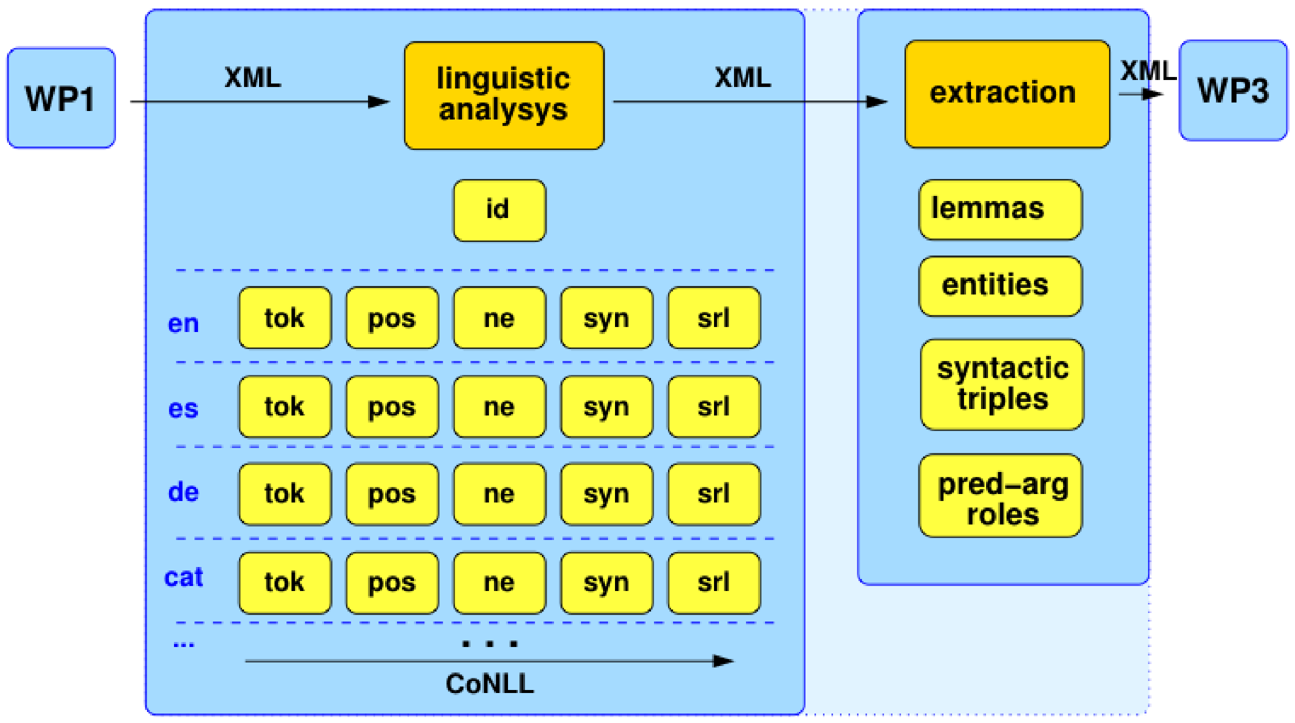

Figure 1 Architechture of Linguistic Processing Services in WP2

- **Pos Tagging (pos):** Performs lemmatization and part-of-speech disambiguation of sentences, using a statistical tagger.
- **Named Entity Recognition (ne):** Recognizes the named entities and classifies them according to their semantic type (i.e. Person, Organization, Location, ...), using a statistical entity tagger.

Except the Language Identification method, all other methods are language dependent, meaning that for each language we have a specific language model. Hence, each language has its own pipeline.

1.3 Deep linguistic analysis

During M7-12 the main effort has been in adding syntactic parsers for all languages. Disambiguating syntax, and therefore having access to the syntactic tree, will allow to extract much more meaningful relations. The methods for deep processing are:

- **Dependency Parsing (syn):** Performs syntactic disambiguation, producing a dependency tree for each sentence.
- **Semantic Role Labeling (srl):** Analyzes the predicate-argument structures of the sentence. Specifically, it identifies the lexical predicates of the sentence, and annotates their arguments, each tagged with a semantic role. This task lies in between syntactic and semantic disambiguaiton.

1.4 Extraction methods

This set of methods receives the output of the linguistic analysis, represented in CoNLL Format, and extract target patterns that are required for the rest of WP in XLike. We distinguish the methods with respect to the type of patterns they extract.

- **Lemmas:** Extracts the lemma of each token. It is trivial after PoS Tagging the sentences.

- **Entities:** Extracts the named entities. It is trivial after running the Named Entity Recognition method. In future, we will run coreference methods as well, which will allow us to link an entity to all its mentions in a document.
- **Syntactic Triples:** Extracts syntactic relation triples of the form subject-verb-object. It can be done after running Dependency Parsing.
- **Semantic Triples:** Extracts predicate-argument relations, a semantic version of syntactic triples. It is trivial after running Semantic Role Labeling on sentences.

Note that to extract lemmas and entities only shallow methods need to be run. To extract triples (whether syntactic or semantic), it is necessary to run both the shallow and the deep methods.

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 relation extraction.

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.

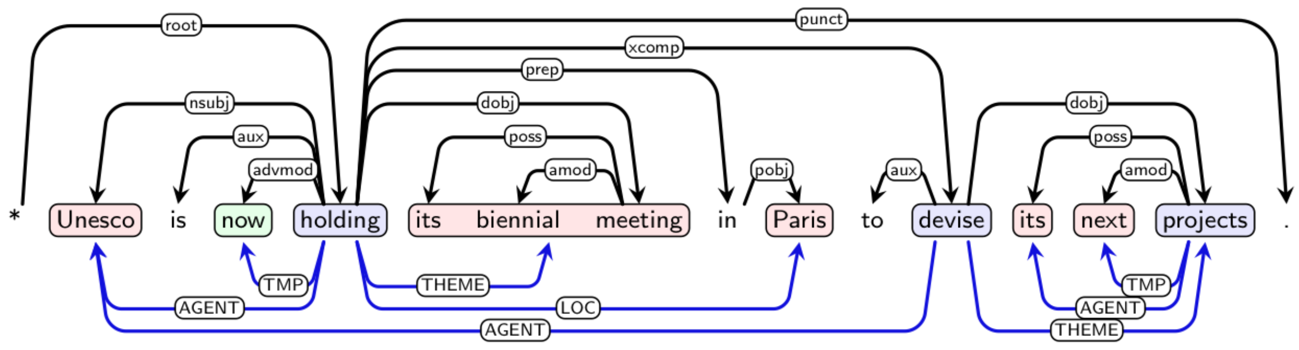

Figure 2 Example of a sentence annotated with a dependency tree (structure above, in black) and predicate argument structures (below, in blue)

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]	Stanford Converter [MMM06]	48
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 "holding", "devise" and "projects". The dependency between "holding" and "Unesco" indicates that "Unesco" is the argument of the predicate "hold" 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. Unfortunately, no treebank annotated with semantic roles exists for Slovene.

An early prototype for English is being integrated in the English pipeline.

2.3 Relation Extraction

After running deep processing methods, it is easy to extract relations of the sentence. In XLike we are interested in extracting triple relations:

- Syntactic: subject-verb-object
- Semantic: agent-predicate-theme

The following example illustrates how these triples look in the dependency graph.

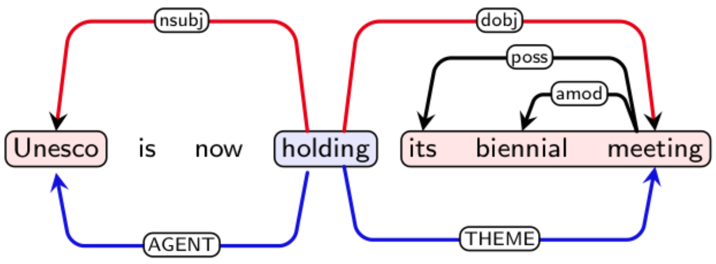

Figure 3 An example of syntactic (top) and semantic (bottom) triples

Extraction of triples is straightforward. All that is required is the label for syntactic dependencies indicating the subject and object, and the label for semantic dependencies indicating agent and theme. These labels are dependant on the treebank that was used to develop the statistical models. We should also note that in a dependency-based representation the relations are in terms of words, but it is trivial to obtain the "constituent" that a word is representing. In the example, it is trivial to follow the dependencies that start at "meeting" in order to extract that the object/theme is "its biennial meeting" rather than just "meeting".

The mentioned triplets are defined in terms of verbs, or predicates in general. In other words, a relation between two entities (subject and object) exists if there is a *trigger* verb that links them. Often it is desirable to know if there is a linguistic relation between any two entities. For example, we may detect the named entities of a sentence, and ask if there is a relation between any pair. In that case, it is very useful to consider the dependency path that connects these entities, as this path is a strong indicator of the type of relation that may exist between them. The following example illustrates syntactic and semantic paths between "Unesco" and "Paris":

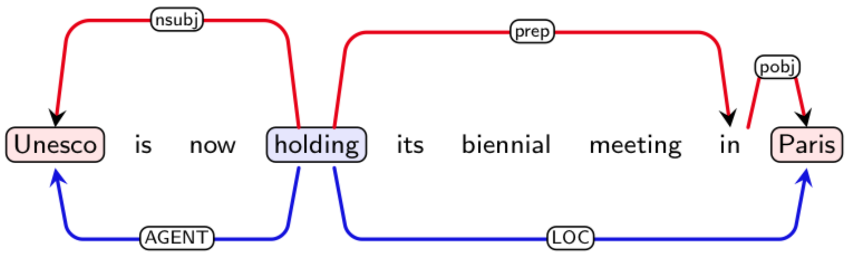

Figure 4 Syntactic (top) and semantic (bottom) dependency paths between named entities.

We have developed a tool that extracts such paths between any two entities. We may then have a classifier deciding if there is a relation between the entities, and of which kind. At the moment, we have developed a method that makes that decision based on regular expressions on the path. The main advantage is that it is relatively simple to write regular expressions that recognize a number of simple but frequent patterns. In future, we may consider applying machine learning methods, which would require some sort of supervision.

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 architechture, deep linguistic services are offered by UPC as stand-alone services. For example, UPC offers specific services for syntactic and semantic parsing for German: these services require a sentence analyzed with pos tags, and return syntactic and 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
Relation Extraction	base_url/upc_linguistic_services/**relations**	**lang:** language code **conll:** the sentence in CoNLL Format, fully parsed	syn: a list of syntactic triples sem: a list of semantic triples

See [D2.1.1] for a description of the CoNLL format, and [D6.2.1] for a specification of all services and of the XML schema.

4 Evaluation

We present an evaluation of the linguistic analysis methods implemented so far in WP2. We divide the evaluations with respect to the predicted layer of linguistic analysis: lemms and part-of speech tags, named entities, and syntactic dependencies.

4.1 Lemmas and Part of Speech

We evaluate the accuracy of the lemmatizers and part of speech (PoS) taggers using the Treebanks in Table 2 as gold annotations. The evaluation metric is accuracy: the percentage of lemmas/PoS tags correctly predicted in the test. For Slovene, the experiment was done using 10-fold cross-validation on the entire treebank ---more details are in [GKD12] (in Slovene). For the rest of languages, we used the standard test set of the treebank.

To perform this evaluation, we took the tokenization of the treebank and ran the PoS taggers with tokenized input. Note that if we run our tools with raw sentences (as opposed to tokenized ones) we would obtain some inconsistencies in the tokenization that would complicate the evaluation.

The results are the following:

Table 4 Lemma and PoS accuracies

	Lemmas	*Part-of-Speech*
en	96.6	96.6
es	96.3	95.1
de	70.7	88.0
ca	97.1	93.7
zh	100	90.8
sl	97.9	92.5

In some cases, the results are lower than what we would expect. This is specially the case for German, but also for the other languages in general. In some cases, the type of PoS tags of the test set are different than those predicted by the tagger. During year 2 we will analyze the source of this error and provide solutions (either adapt the test corpus, or change the type of predictions to be consistent with the test corpus).

4.2 Named Entities

We evaluate named entity recognition systems using the methodology of the CoNLL 2003 Shared Task [TM03]. In general we distinguish four types of entites, namely locations (LOC), person names (PER), organizations (ORG) and miscellaneous entities. The evaluation metrics are based on precision and recall at the entity level:

- Precision: the percentage of entities predicted by the system that are correct
- Recall: the percentage of correct entities that are predicted by the system
- F1: the geometric mean between Precision and Recall

For an entity to be considered correctly predicted, the words forming the entity and the type of entity have to be correct. That is, there is no credit for partially recognizing an entity.

For English and German, we use the test sets of CoNLL-2003. For Spanish and Catalan we use the test sets of the Ancora corpus [Anc]. For Chinese, we use the test of CSDN [CSDN] (does not annotate MISC entities), and for Slovene we use the Učni corpus [Ucni] (does not annotate ORG entities).

The results are the following. We give performance per class, and an average for the four classes:

Table 5 Accuracy of Named Entity Recognition and Classification

	LOC			*PER*			*ORG*			*MISC*			*Average*		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
en	84.6	76.8	80.5	78.0	77.2	77.6	57.1	67.2	61.8	70.5	49.3	58.0	71.8	70.6	71.2
es	60.8	69.0	64.7	85.1	75.9	80.2	66.9	72.6	69.6	45.1	42.3	43.6	69.0	70.0	69.5
de	73.1	52.3	61.0	86.0	70.3	77.4	76.6	46.0	57.5	73.2	39.5	51.3	61.6	54.5	57.8
ca	63.7	53.8	58.3	68.6	73.9	71.1	55.8	65.1	60.1	31.3	21.7	25.6	58.8	58.4	58.6
zh	93.1	93.7	93.5	85.0	87.5	86.2	62.4	90.7	73.9	-	-	-	86.9	91.0	88.9
sl	76.6	77.7	77.1	82.1	81.6	81.8	-	-	-	56.3	47.6	51.6	74.1	71.0	72.5

4.3 Syntactic Dependencies

Following standard methodology [NHK+07], we evaluate dependency parsers with the following accuracy measures. Note that in a sentence every token has exactly one syntactic head. The measures are:

- Unlabeled Attachment Score (UAS): percentage of words with the correct head, ignoring the dependency label
- Labeled Attachment Score (LAS): percentage of words with correct head and dependency label

As in PoS tagging, we use the standard test sets of the treebanks in Table 2. The results are the following:

Table 6 Accuracy of Dependency Parsers using predicted PoS Tags

	UAS	*LAS*
en	86.1	83.0
es	86.6	82.5
de	80.0	75.2
ca	85.8	81.5
zh	80.0	72.0

Dependency parsing is highly dependant on the quality of PoS tags. We did an additional evaluation where we run the dependency parser for English using the correct PoS tags, and we obtained UAS=90.5 and LAS=89.2, a result much more comparable to the state-of-the-art [MCP05,MP06,Car07,NHK07+]. Part of the problem is that the PoS tags predicted by the tagger are slightly different than those of the treebank, but not necessarily erroneous. On the other hand, the syntactic parser expects PoS tags as in the treebank. During year 2 we will fix these inconsistencies.

Conclusions

We have described methods in the XLike WP2 subsystem for syntactic and semantic analysis of texts. Our choice of representation of deep linguistic relations is based on dependencies, an intuitive formalism that represents linguistic relations directly as labeled links between words. We have also described what type of patterns can be extracted using these representations.

The methods we have described are a natural complement to the shallow methods developed earlier. Importantly, we have evaluated our methods, from PoS tagging, to Named Entities, to Syntactic Parsing. Our results are below the state-of-the-art, but we believe that an important part of the errors are not necessarily incorrect predictions, but differences in the annotation schemes used by different methods in the pipeline.

The main effort of Year1 has been in designing an architechture to implement six different pipelines, each using a number of different linguistic modules. Year 2 should improve the quality in order to meet state-of-the-art accuracy of linguistic analysis. In fact, adapting annotation schemes so that linguistic annotations are compatible accross modules *within* a pipeline is only a particular case of the reseach challenge posed by XLike: how to find linguistic representions that are compatible in a cross-lingual context.

References

[D2.1.1] XLike deliverable *“D2.1.1 – Shallow 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.

[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