Natural Language Engineering **1** (1): 1–38. Printed in the United Kingdom

Data-Driven Deep-Syntactic Dependency Parsing

MIGUEL BALLESTEROS[1],

BERND BOHNET[2],

SIMON MILLE[1],

LEO WANNER[1,3]

[1] *Pompeu Fabra University, Natural Language Processing Group, Roc Boronat 138, 08018 Barcelona, Spain*
e-mail: ***miguel.ballesteros|simon.mille|leo.wanner@upf.edu***
[2] *Google Inc.*
e-mail: ***bohnetb@gmail.com***
[3] *Catalan Institute for Research and Advanced Studies (ICREA)*

(*Received xx xxxx 2014; revised xx xxxx 2015*)

Abstract

"Deep-syntactic" dependency structures that capture the argumentative, attributive and coordinative relations between full words of a sentence have a great potential for a number of NLP-applications. The abstraction degree of these structures is in between the output of a syntactic dependency parser (connected trees defined over all words of a sentence and language-specific grammatical functions) and the output of a semantic parser (forests of trees defined over individual lexemes or phrasal chunks and abstract semantic role labels which capture the frame structures of predicative elements and drop all attributive and coordinative dependencies). We propose a parser that provides deep syntactic structures. The parser has been tested on Spanish, English and Chinese.

1 Introduction

State-of-the-art syntactic dependency parsing delivers *surface-syntactic structures* (SSyntSs), which are *per force* idiosyncratic in that they are defined over the entire vocabulary of a language (including governed prepositions, determiners, support verb constructions, etc.) and language-specific *grammatical functions* such as, e.g., SBJ, OBJ, PRD, PMOD, etc.; see, among others (McDonald *et al.*, 2005; Nivre *et al.*, 2007b; Kübler *et al.*, 2009; Bohnet & Kuhn, 2012; Bohnet & Nivre, 2012; Dyer *et al.*, 2015). On the other hand, semantic (or deep) parsing delivers logical forms (LFs) or semantic structures (SemSs) equivalent to LFs,[1] PropBank (Palmer *et al.*,

[1] The first generation language understanding approaches dealt with abstract conceptual meaning representations that could be mapped onto LFs; see, among others, (Bobrow & Webber, 1981; Dahlgren, 1988; Kasper & Hovy, 1990).

2005) or FrameNet (Fillmore *et al.*, 2002) structures (SemSs). See, for instance, (Allen *et al.*, 2007; Bos, 2008; Oepen & Lønning, 2006; Miyao, 2006) and the DM- and PAS-parsers of the SemEval 2014 shared task (Oepen *et al.*, 2014) for LF outputs. (Johansson & Nugues, 2008a; Zhao *et al.*, 2009; Gesmundo *et al.*, 2009; Henderson *et al.*, 2013) deliver PropBank structure output, and (Das *et al.*, 2014) FrameNet structure output.

Parsers working with LFs tend to abstract not only over surface-oriented linguistic information (such as determination, tense, etc.) but also over distinctive (shallow) semantic relations. Thus, in Boxer (Bos, 2008) and in other parsers that produce LFs, the phrases *the dancing girl* and *the girl dances* will result in the same relation between 'dance' and 'girl'. PropBank and FrameNet structures are forests of trees, defined over disambiguated lexemes or phrasal chunks and thematic roles (A0, A1, ..., ARGM-DIR, ARGM-LOC, etc. in the case of PropBank structures and Agent, Object, Patient, Value, Time, Beneficiary, etc. in the case of frame structures), with usually omitted attributive and coordinative relations (be they within chunks or sentential).

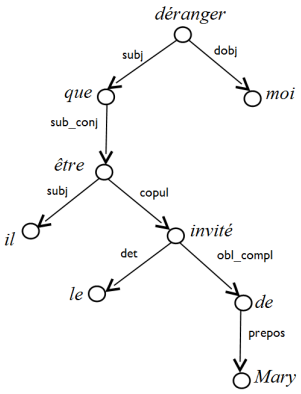
For many NLP-applications, including machine translation, paraphrasing, text simplification, etc., neither SSyntSs nor LFs or SemSs are adequate: the high idiosyncrasy of SSyntSs is obstructive because of the recurrent divergence between the source and the target structures, while the high abstraction of LFs and SemSs is problematic because of the loss of linguistic structure information in the case of LFs and dependencies between chunks and the loss of meaningful content elements in the case of SemSs. "Syntactico-semantic" structures in the sense of *deep-syntactic structures* (DSyntSs) as defined in the *Meaning-Text Theory* (Mel'čuk, 1988) are in this sense arguably more appropriate. DSyntSs are situated between SSyntSs and LFs/SemSs. Compared to SSyntSs, they have the advantage to abstract from language-specific grammatical idiosyncrasies. Compared to LFs, PropBank and Frame structures, they have the advantage to be complete, i.e., capture all and distinguish all argumentative, attributive and coordinative dependencies between the meaning-bearing lexical items of a sentence, and to be connected. As a consequence, for instance, in the context of Machine Translation, DSyntSs help reduce the number and types of divergences between the source language $\mathcal{L}_S$ and destination language $\mathcal{L}_D$ structures to the minimum to make the transfer straightforward (Mel'čuk & Wanner, 2006; Mel'čuk & Wanner, 2008),[2] but are still syntactic and thus reflect the communicative intention of the speaker (Steedman, 2000). Consider, for instance, a French–English sentence pair in (1).

(1) Fr. *Qu'il soit l'invité de Mary me dérange*
lit. 'That he be the invited of Mary me bothers.'
≡
His being Mary's invitee bothers me.

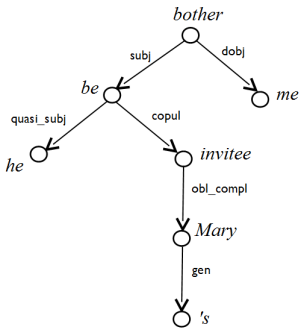
In French, the subject has to be a full clause, hence the presence of a subordi-

[2] As shown by Mel'čuk and Wanner, the remaining morphological and syntactic mismatches at the DSyntS-level can be handled in a principled way.

nating conjunction *que* 'that', which links the embedded verb and the main verb in the SSyntS. In addition, *invité* 'invited' has to bear a determiner, and the genitive construction is realized through the use of the preposition *de* 'of' (which is also possible although less idiomatic in English). Figure 1 shows the corresponding SSyntSs, PropBank structure, and DRS (the Propbank structure and DRS are, in principle, the same for both the French and the English sentence). As can be observed, the SSyntSs differ considerably, while the PropBank structure provides only a partial argumental structure, and the DRS blurs the difference between the main and the embedded clauses.[3] None of the variants is thus optimal for MT.



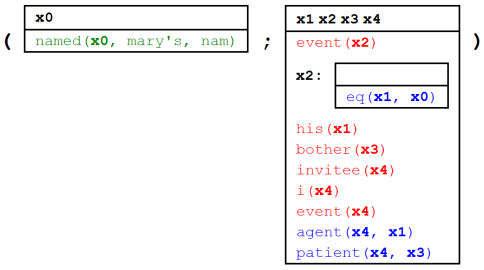
(a) SSyntS Fr. of (1)



(b) SSyntS Engl. of (1)

	His	being	Mary	's	invitee	bothers	me	.
annoy.01		A0					A1	

(c) PropBank structure of (1)



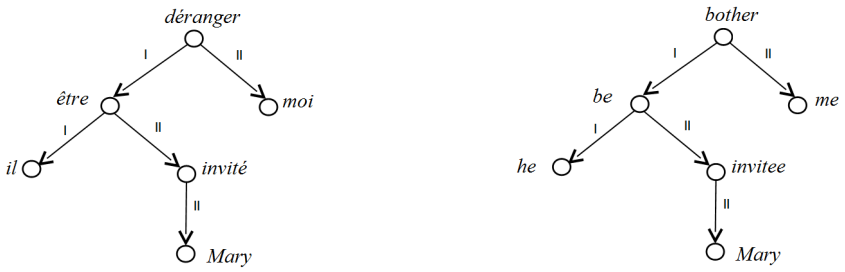
(d) DRS of (1), as obtained with Boxer

Fig. 1: SSyntSs, PropBank structure, and DRS of (1)

The respective DSyntSs in Figure 2 avoid the idiosyncrasies of SSyntSs and the

[3] The DRS has been obtained automatically with the Boxer demo version at `http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Demo`. In other words, we cannot ensure that it is the correct DRS; see (Kamp & Reyle, 1993) for a theoretical presentation of DRSs. However, since we focus on the type of information stored in a DRS, this does not invalidate our argumentation.

over-generalizations of PropBank/DRS. They are isomorphic and facilitate thus a straightforward transfer.



(a) DSyntS Fr. of (1) (b) DSyntS Engl. of (1)

Fig. 2: DSyntSs of (1)

Based on these observations, we propose to put on the research agenda of statistical parsing the task of deep-syntactic parsing. This task is not really novel. Thus the idea of the surface→surface syntax→deep syntax pipeline goes back at least to (Curry, 1961) and is implemented in a number of more recent works; see, e.g., (Klimeš, 2006; Klimeš, 2007), which produce tectogrammatical structures in the sense of the Prague school,[4] (de Groote, 2001), which obtains a deep categorial grammar structure, and (Rambow & Joshi, 1997), which provide deep analysis in the TAG-framework. Moreover, in the SemEval 2014 shared task on Broad-Coverage Semantic Dependency Parsing, the target structures (Oepen *et al.*, 2014) show a similarity with DSyntSs. However, as pointed out above and as will be argued further below in more detail, DSyntSs still show some advantages over most of the other common structures. Nonetheless, the primary goal of this paper is not to push forward the use of DSyntSs. Rather, we aim to propose a novel way to obtain DSyntSs (or structures that are equivalent to DSyntSs) from a SSynt dependency parse using data-driven tree transduction in a pipeline with a syntactic parser.

The paper is an extension of (Ballesteros *et al.*, 2014). Compared to (Ballesteros *et al.*, 2014), it contains a more detailed discussion of the theoretical background and of the data sets, more exhaustive experiments with more challenging baselines not only on Spanish, but also on Chinese and English, and a deeper analysis of the outcome of these experiments. The latest version of the source code and the package distribution of our DSynt parser are available at `https://github.com/talnsoftware/deepsyntacticparsing/wiki`.

The remainder of the paper is structured as follows. In Section 2, we introduce DSyntSs and SSyntSs. Section 3 discusses the fundamentals of SSyntS–DSyntS transduction. Section 4 describes the experiments that we carried out on Spanish, Chinese and English material, and Section 5 presents their outcome. Section 6 summarizes the related work, before in Section 7 some conclusions and plans for future work are presented.

[4] Bojar *et al.* (2008) even discuss an MT-model at the tectogrammatical layer.

2 Linguistic Fundamentals of SSyntS–DSynt Transduction

Before we set out to discuss the principles of the SSyntS–DSynt transduction, we define the notions of DSyntS and SSyntS as used in our experiments and the types of correspondences between the two.

2.1 The Surface- and Deep-syntactic structures

SSyntSs and DSyntSs are directed, node- and edge-labeled dependency trees with standard feature-value structures (Kasper & Rounds, 1986) as node labels and dependency relations as edge labels. Both differ, however, with respect to the abstraction of linguistic information: DSyntSs capture predicate-argument relations between meaning-bearing lexical items, while these relations are not captured by SSyntSs. At the same time, DSyntSs maintain the sentence structure (as the SSyntSs do).

The features of the node labels in **SSyntSs** are *lex*, which captures the name of the lexical item, and "syntactic grammemes" of this name, i.e., *number*, *gender*, *case*, *person* for nouns and *tense*, *mood* and *finiteness* for verbs. The value of *lex* can be any (either full or functional) lexical item. The edge labels of a SSyntS are grammatical functions 'subj', 'dobj', 'det', 'modif', etc. In other words, SSyntSs are syntactic structures of the kind as encountered in the standard dependency treebanks: dependency version of the Penn Treebank (Johansson & Nugues, 2007) for English, Prague Dependency Treebank for Czech (Hajič *et al.*, 2006), AnCora for Spanish (Taulé *et al.*, 2008), Copenhagen Dependency Treebank for Danish (Buch-Kromann, 2003), etc. In formal terms, which we need for the outline of the transduction below, a SSyntS is defined as follows:

Definition 1 (SSyntS)
An SSyntS of a language $\mathcal{L}$ is a quintuple $T_{SS} = \langle N, A, \lambda_{l_s \to n}, \rho_{r_s \to a}, \gamma_{n \to g} \rangle$ defined over all lexical items L of $\mathcal{L}$, the set of syntactic grammemes G_{synt}, and the set of grammatical functions R_{gr}, where

- the set N of nodes and the set A of directed arcs form a connected tree,
- $\lambda_{l_s \to n}$ assigns to each $n \in N$ an $l_s \in L$,
- $\rho_{r_s \to a}$ assigns to each $a \in A$ an $r \in R_{gr}$,
- $\gamma_{n \to g}$ assigns to each $\lambda_{l_s \to n}(n)$ a set of grammemes $G_t \in G_{synt}$.

The top structure in Figure 3 shows a sample SSyntS, where the feature-value information for three nodes is made explicit for illustration. A more common graphical representation of a SSyntS (which does not show explicitly the features and their corresponding values) is displayed in Figure 4(a).

The features of the node labels in **DSyntSs** are *lex* and "semantic grammemes" of the value of *lex*, i.e., *number* and *definiteness* for nouns and *tense*, *finiteness*, *mood*, *voice* and *aspect* for verbs.[5] In contrast to *lex* in SSyntS, DSyntS's *lex* can

[5] Most of the grammemes have a semantic and a surface interpretation; see (Mel'čuk, 2013).

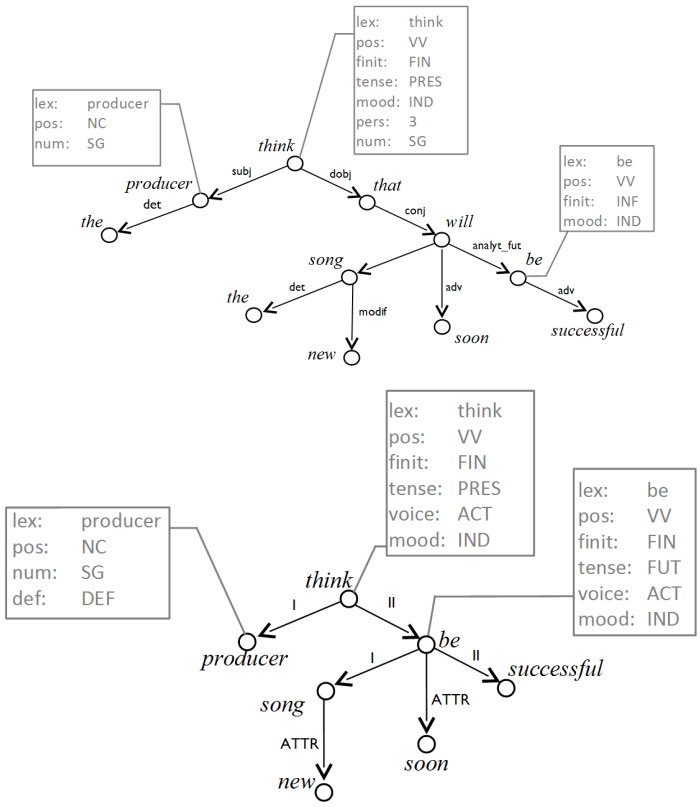


Fig. 3: SSyntS (top) and DSyntS (bottom) for the sentence *The producer thinks that the new song will be successful soon*

be any *full*, but not a *functional* lexeme. In accordance with this restriction, in the case of *look after a person*, AFTER will not appear in the corresponding DSyntS since it is a functional (or governed) preposition. In contrast, AFTER in *leave after the meeting* will remain in the DSyntS because there it has its own meaning of "succession in time". The edge labels of a DSyntS are "deep-syntactic" relations I,...,VI, ATTR, COORD, APPEND. 'I',...,'VI' are argument relations, analogous to *A*0, *A*1, etc. in the PropBank annotation. 'ATTR' subsumes all (circumstantial) ARGM-x PropBank relations as well as the modifier relations not captured by the PropBank and FrameNet annotations. 'COORD' is the coordinative relation as in: *John*-COORD→*and*-II→*Mary*, *publish*-COORD→*or*-II→*perish*, and so on. APPEND subsumes all parentheticals, interjections, direct addresses, etc., as, e.g., in *Listen, John!*: *listen*-APPEND→*John*. DSyntSs thus show a strong similarity with PropBank structures, with four important differences: (i) their lexical labels are not disambiguated;[6] (ii) instead of circumstantial thematic roles of the kind

[6] As a matter of fact, in genuine DSyntSs as defined in (Mel'čuk, 1988), lexical labels are disambiguated. It is only in our current interpretation that they are not.

ARGM-LOC, ARGM-DIR, etc. they use a unique ATTR relation; (iii) they capture all existing dependencies between meaning-bearing lexical nodes; and (iv) they are connected. Formally, a DSyntS is defined as follows:

Definition 2 (DSyntS)
A DSyntS of a language $\mathcal{L}$ is a quintuple $T_{DS} = \langle N, A, \lambda_{l_s \to n}, \rho_{r_s \to a}, \gamma_{n \to g} \rangle$ defined over the full lexical items L_d of $\mathcal{L}$, the set of semantic grammemes G_{sem}, and the set of deep-syntactic relations R_{dsynt}, where

- the set N of nodes and the set A of directed arcs form a connected tree,
- $\lambda_{l_s \to n}$ assigns to each $n \in N$ an $l_s \in L_d$,
- $\rho_{r_s \to a}$ assigns to each $a \in A$ an $r \in R_{dsynt}$,
- $\gamma_{n \to g}$ assigns to each $\lambda_{l_s \to n}(n)$ a set of grammemes $G_t \in G_{sem}$.

The bottom structure in Figure 3 and Figure 4(b) show examples of DSyntSs.

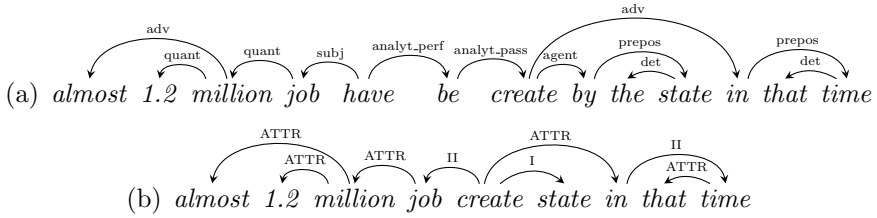


Fig. 4: SSyntS and DSyntS for the sentence *Almost 1.2 million jobs have been created by the state in that time*

As mentioned, a number of other annotations have resemblance with DSyntSs. In particular, as already pointed out, DSyntSs show some resemblance but also some important differences with PropBank structures, mainly due to the fact that the latter concern phrasal chunks and not individual nodes. Figure 5 shows the PropBank structure that corresponds to the SSyntS and DSyntS in Figure 4. The square brackets in the PropBank structure indicate the constituents that implicitly form part of the arguments of A1 and AM-TMP, respectively.

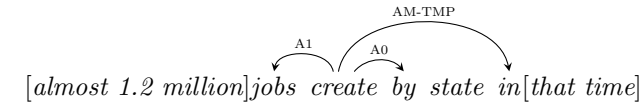


Fig. 5: PropBank structure of the sentence *Almost 1.2 million jobs have been created by the state in that time*

The target structures of the SemEval 2014 shared task on Broad-Coverage Semantic Dependency Parsing (Oepen *et al.*, 2014) also show some similarities with DSyntSs. For instance, the DELPH-IN annotation, which is a rough conversion of the Minimal Recursion Semantics treebank (Oepen & Lønning, 2006) into bi-lexical dependencies, also captures the lexical argument (or valency) structure and eliminates some functional elements (such as *be* copula and prepositions). The Enju

annotation (Miyao, 2006) is a pure predicate-argument graph over all the words of a sentence. However, it distinguishes arguments of functional elements (auxiliaries, infinitive and dative TO, THAT, WHETHER, FOR complementizers, passive BY) in that they are attached to the semantic heads of these elements (rather than to the elements themselves). This facilitates the disregard of functional elements—as in DSyntSs (cf. (Ivanova *et al.*, 2012) for a more complete overview of Enju and DELPH-IN).

The degree of "semanticity" of DSyntSs can be directly compared to Prague's tectogrammatical structures (PDT-tecto (Hajič *et al.*, 2006)), which contain *autosemantic* words only. *Synsemantic* elements such as determiners, auxiliaries, prepositions and conjunctions are not kept in tectogrammatical structures. Thanks to the distinction between argumental and non-argumental edges, tectogrammatical structures are trees, not graphs. That is, as in the DSyntSs, they maintain the syntactic structure of the sentence. The main differences between DSyntSs and tectogrammatical structures are: (i) in tectogrammatical structures, no distinction is made between governed and non governed prepositions and conjunctions, and (ii) in tectogrammatical structures, the vocabulary used for edge labels emphasizes "semantic" content over predicate-argument information. For instance, a label like ADDR (*addressee*) indicates that the dependent is an argument of its governor, but does not say which slot is occupied in the valency frame of the latter. At the same time, this tag indicates that the dependent is the recipient of a message, which a simple ARG2 label for instance does not encode.[7] DSyntSs, on the other hand, have the advantage to directly encode predicate-argument structures and thus be straightforwardly connected to existing lexical resources such as PropBank or NomBank, and through these to deeper representation such as VerbNet (Schuler, 2005) and FrameNet structures; see (Palmer, 2009).

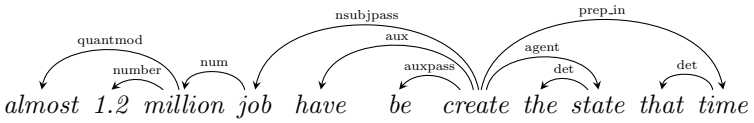


Fig. 6: Collapsed Stanford Dependency structure of the sentence *Almost 1.2 million jobs have been created by the state in that time*

Although the annotations are not really of the same nature, DSyntSs can be furthermore contrasted to the *Collapsed Stanford Dependencies (SD)* (de Marneffe & Manning, 2008). Collapsed SDs differ from DSyntSs (apart from the fact that that they may be (sometimes) disconnected graphs) in that: (i) in the same fashion as in the Prague Dependency Treebank, they collapse only (but all) prepositions, conjunctions and possessive clitics, whereas DSyntSs omit all functional nodes (all auxiliaries, some determiners, and some prepositions and conjunctions); (ii) they

[7] See (Oepen *et al.*, 2014) for a parallel illustration of DELPH-IN, Enju and tectogrammatical structures.

do not involve any removal of (syntactic) information since the meaning of the preposition remains encoded in the label of the collapsed dependency, while DSyntSs omit or generalize the purely functional elements; (iii) they do not add semantic information compared to the surface annotation. That is, Collapsed SDs keep the surface-syntactic information, representing it in a different format, while DSyntSs keep only deep-syntactic information. Consider Figure 6 for illustration.[8]

As in all mentioned annotations (except in SD), the opposition between active and passive voice is neutralized in the DSynSs—for instance, both the first object of an active verb and the subject of a passive verb are annotated as second arguments. As in PDT-tecto, PropBank and SD, in DSyntSs multi-word expressions (MWEs) are handled through a specific dependency relation; in DRS and DELPH-IN, special predicates exist, which take as arguments the components of a MWE, while in Enju, MWEs are not annotated. In our current version of the DSyntSs (as in SD, DRS, DELPH-IN, and Enju), predicates are not disambiguated and light verb constructions, which are the most common type of MWEs, are annotated as regular constructions. In contrast, in the PropBank and PDT-tecto annotations verbs and nouns are disambiguated, and an independent resource with lexical units and their valency frames is compiled (PropBank lexicon and PDT-VALLEX). In PropBank and PDT-tecto, light verb constructions are also annotated: as MWEs in PDT-tecto, and as independent lexical units in the PropBank lexicon. Finally, in DSyntSs, argument sharing is not represented, since at this level the structures must be trees and one node can thus receive one and only one incoming arc. In the Meaning-Text framework, argument sharing is made explicit at the semantic layer, where the structures are predicate-argument graphs. The PDT-tecto annotation is also arborescent, but its authors made the choice to annotate argument sharing by duplicating shared arguments in the tree for control and coordinate structures.[9] PropBank, SD, DRS, DELPH-IN and Enju, are graph representations, so shared arguments in coordinate and control constructions are not an issue. However, in PropBank and SD, special relations are used in some case of control constructions (and other phenomena), respectively *C-AM* and *xsubj* relations.

2.2 SSyntS–DSyntS correspondences

The implementation of the transduction from SSyntS to DSyntS requires a prior detailed analysis of the correspondences between elements of SSyntS and DSyntS. Let us thus discuss the correspondences between the two types of structures, based on the example in Figure 7 (in which the grammemes are not shown for the sake of clarity); we use a Spanish example (instead of, e.g., an English one) because it allows us to illustrate all relevant phenomena.

[8] Figure 6 has been obtained through manual revision of the output of the online Stanford demo page (`http://nlp.stanford.edu:8080/parser/index.jsp`); we are responsible for possible erroneous dependencies.

[9] Actually, given their annotation of coordinations, shared arguments do not always have to be duplicated: with the conjunction as the governor of all the conjuncts, shared arguments are simply made dependents of the conjunction.

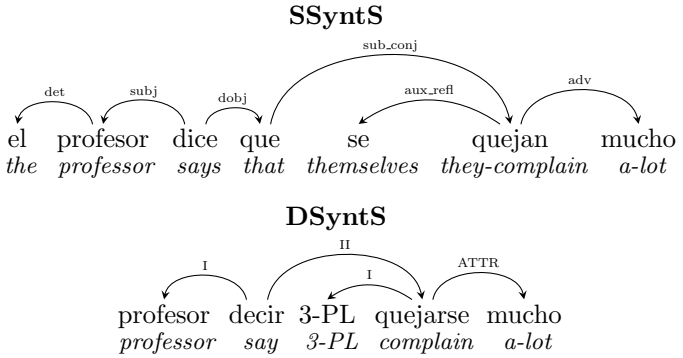


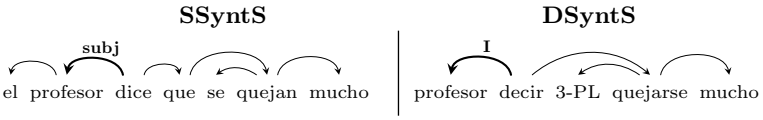
Fig. 7: SSyntS and DSyntS of the sentence *el profesor dice que se quejan mucho* 'the professor says that they complain a lot'

The following correspondences between the SSyntS S_{ss} and DSyntS S_{ds} of a sentence need to be taken into account during the SSyntS–DSyntS transduction:[10]

(i) a node in S_{ss} is a node in S_{ds}:

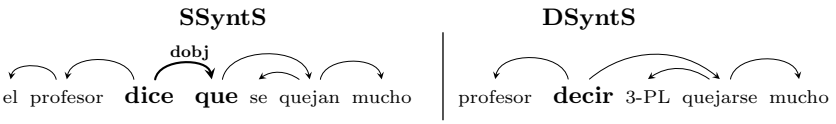
The node *mucho* 'a-lot' has a single correspondent in the DSyntS. This is also the case of the node *profesor*.

(ii) a relation in S_{ss} corresponds to a relation in S_{ds}:



The SSynt relation *subj* is mapped to the DSynt relation *I*. Note that the relation-to-relation mapping is not necessarily unique. Thus, *subj* is mapped to *II* (rather than to *I*) if the verb in the SSyntS is in passive.

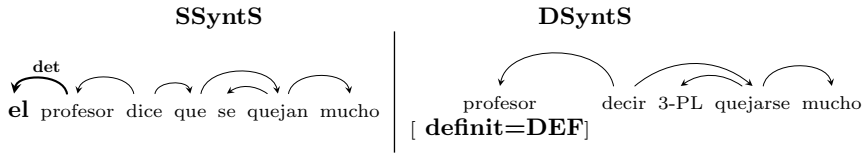
(iii) a fragment of the S_{ss} tree corresponds to a single node in S_{ds}:



The words *dice* 'say' and *que* 'that' and the dependency between them (*dobj*) correspond to one single node in DSynt (*decir* 'say'); in other words, *que* 'that' is not reflected in the DSyntS.

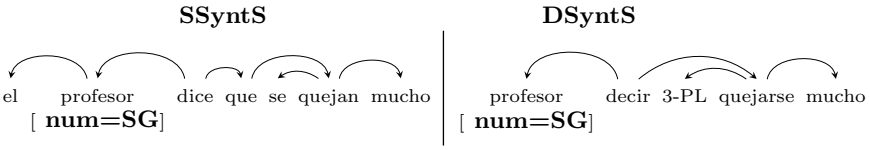
[10] In what follows, the structures in Figure 7 are used to illustrate the correspondences (i)–(vii); we are aware that the citation of the same structures is somehow repetitive, but we believe that it makes the presentation much clearer. For English glosses of the Spanish words, see Figure 7.

(iv) a relation with a dependent or governor node in S_{ss} is a grammeme in S_{ds}:



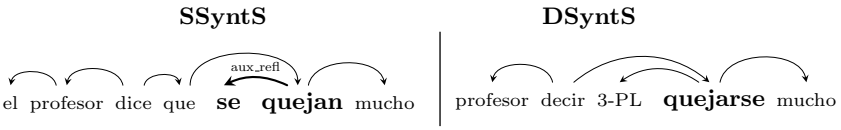
The relation *det* and its dependent, the definite determiner *el* 'the', are stored in the DSyntS as the grammeme of definiteness associated to the node *profesor* 'professor'. Similarly, the auxiliary relations and their governors correspond to a grammeme of voice, tense, or aspect on the node of the dependent verb.[11]

(v) a grammeme in S_{ss} is a grammeme in S_{ds}:



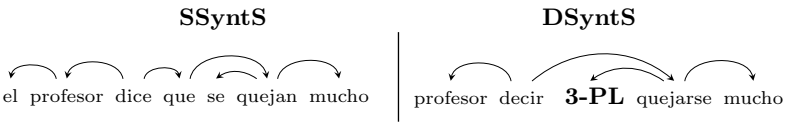
Number grammemes are maintained on nodes which can carry semantic number (that is, on nodes which do not have a number only for agreement reasons, as it can be the case for verbs in English, verbs, determiners and adjectives in Spanish and other languages, etc.), such as singular number on the node *profesor* 'professor'. Other grammemes, such as those of tense, mood, or finiteness are mapped the same way.

(vi) a node in S_{ss} is conflated with another node in S_{ds}:



For this correspondence, the reflexive pronoun *se* 'itself'/'each other' is part of the lemma of the verb in the DSyntS. In the SSyntS, it is separated in order to produce the sentence *se quejan*, lit. 'themselves they-complain'.

(vii) a node in S_{ds} has no correspondence in S_{ss}:



In Spanish, which is a pro-drop language, the subject of a finite verb does not need to be realized, even though there is a node at the DSynt level which accounts for the agreement found on the verb for instance (3^{rd} person plural in this case).

[11] Note that the fact that the determiner and its dependency are mapped onto a grammeme entails that the governing noun actually has a 1 to 1 correspondence with its DSyntS counterpart, even though we consider that both nodes form a hypernode (see Section 3.1).

3 SSyntS–DSyntS Transduction

In this section, we first flesh out the principles of the transduction between SSyntSs and DSyntSs and detail it then step by step.

3.1 Principles of the SSyntS–DSyntS Transduction

In the above list of SSynt-DSynt correspondences, the grammeme correspondences (iv) and (v) and the "pseudo" correspondences in (vi) and (vii) are few or idiosyncratic and are best handled in a rule-based post-processing stage; see Section 3.5. The main task of the SSyntS–DSyntS transducer is thus to cope with the correspondences (i)–(iii). For this purpose, we consider SSyntS and DSyntS trees as two-dimensional matrices $I = N \times N$ (with N as the set of nodes $\{1, \ldots, m\}$ of a given tree and $I(i,j) = \rho_{r_s \to a}(n_i, n_j)$ if $n_i, n_j \in N$ and $(n_i, n_j) = a \in A$ $(i, j = 1, \ldots, m; i \neq j)$ and $I(i,j) = 0$ otherwise.[12] That is, for a given SSyntS, $I(i,j)$ contains in the cell (i,j), $i,j = 1, \ldots, m$ (with $i \neq j$) the name of the SSynt-relation that is encountered in the given tree between the nodes n_i and n_j. If no relation holds between n_i and n_j, the cell $I(i,j)$ contains '0'. In analogy, for a given DSyntS, the cells contain DSyntS-relations between the corresponding nodes.

Starting from the matrix I_s of a given SSyntS, the task is therefore to obtain the matrix I_d of the corresponding DSyntS, that is, to identify correspondences between i_s respectively j_s, (i_s, j_s) and groups of (i_s, j_s) of I_s with i_d respectively j_d and (i_d, j_d) of I_d; see (i)–(iii) above. In other words, the task consists in identifying and removing all functional lexemes, and attach correctly the remaining nodes between them.[13]

As already the projection of a chain of tokens onto an SSyntS, the SSyntS–DSyntS projection can be viewed as a classification task. However, while the 'chain→surface-syntactic tree' projection is isomorphic, the latter is not (see iii). In order to make it appear as an isomorphic projection, it is convenient to interpret SSyntS and the targeted DSyntS as collection of *hypernodes*; cf. Definition 3:

Definition 3 (Hypernode)
Given a SSyntS S_s with its matrix I_s and a DSyntS S_d with its matrix I_d, a node partition p (with $|p| \geq 1$) of I_s/I_d is a hypernode h_{s_i} / h_{d_i} iff p corresponds to a partition p' (with $|p'| \geq 1$) of S_d/S_s.

In other words, a SSyntS hypernode, known as *syntagm* in linguistics, is any SSyntS configuration with a cardinality ≥1 that corresponds to a single DSyntS node. The notion of hypernode is quite generic. It subsumes several types of correspondences discussed in Section 2.2 (namely, (i), (iii), (iv), and (vi)). For instance, *dice que* 'says that', *el profesor* 'the professor', and *se quejan* '(they) complain'

[12] As the reader will have noticed, we use here the graph notation '(n_i, n_j)' to refer to an arc between the (starting) node n_i and the (target) node n_j in a tree. See also the formal definitions of SSyntS and DSyntS in the previous section.

[13] Particularly challenging is the identification of functional prepositions: based on the information found in the corpus only, our system must decide if a given preposition is a full or a functional lexeme. That is, we do not resort to any external lexical resources.

from the example above constitute hypernodes. Hypernodes can also contain more than two nodes, as in the case of more complex analytical verb forms, e.g., *ha sido invitado* 'he-has been invited', which corresponds to the node *invitar* 'invite' in the DSyntS.

In this way, the SSyntS–DSyntS correspondence boils down to a correspondence between individual hypernodes and between individual arcs, such that the transduction embraces the following three (classification) subtasks: (i) hypernode identification, (ii) DSynt tree reconstruction, and (iii) DSynt arc labeling, which are completed by (iv) post-processing.

3.2 Hypernode identification

The hypernode identification consists of a binary classification of the nodes of a given SSyntS as nodes that form a hypernode of cardinality 1 (i.e., nodes that have a one-to-one correspondence to a node in the DSyntS) vs. nodes that form part of a hypernode of cardinality > 1. In practice, hypernodes of the first type (henceforth, "type 1" or '$h1$') will be formed by: 1) noun nodes that do not govern (in)definite determiner or functional preposition nodes, 2) full verb nodes that are not governed by any auxiliary verb nodes and that do not govern any functional preposition node, and 3) adjective, adverbial, and semantic preposition nodes which do not govern functional preposition nodes.

Hypernodes of the second type (henceforth, "type 2" or '$h2$') will be formed by: 1) noun nodes + (in)definite determiner + functional preposition nodes they govern, 2) verb nodes + auxiliary nodes they are governed by + functional preposition nodes they govern + reflexive pronoun *se* 'oneself' when it is part of the lemma of the verb, and 3) adjective, adverbial, and semantic preposition nodes + functional preposition nodes they govern.

The following sentence shows different examples of hypernodes of type 1 (*h1*) and type 2 (*h2*):

(2) [*El capitán de*]$_{h2}$ [*la embarcación*]$_{h2}$ [*se ha puesto a*]$_{h2}$ [*cantar*]$_{h1}$ [*cuando*]$_{h1}$ [ha visto a]$_{h2}$ [*cuatro*]$_{h1}$ [*delfines*]$_{h1}$ [*adultos*]$_{h1}$ [*saltar*]$_{h1}$ [*cerca de*]$_{h2}$ [*nosotros*]$_{h1}$.
'[The captain of]$_{h2}$ [the boat]$_{h2}$ [(reflexive+has) started to]$_{h2}$ [sing]$_{h1}$ [when]$_{h1}$ [he-has seen *prep*]$_{h2}$ [four]$_{h1}$ [dolphins]$_{h1}$ [adults]$_{h1}$ [jump]$_{h1}$ [next to]$_{h2}$ [us]$_{h1}$.'

3.3 DSynt tree reconstruction

The outcome of the hypernode identification stage is thus the set $H_s = H_{s_{|p|=1}} \cup H_{s_{|p|>1}}$ of hypernodes of two types. With this set at hand, we can define an isomorphic function $\tau : H_s \rightarrow H_{d_{|p|=1}}$ (with $h_d \in H_{d_{|p|=1}}$ consisting of $n_d \in N_{ds}$, i.e., the set of nodes of the target DSyntS). τ is the identity function for $h_s \in H_{s_{|p|=1}}$. For $h_s \in H_{s_{|p|>1}}$, τ maps the functional nodes in h_s onto grammemes (attribute-value tags) of the meaning-bearing node in h_d and identifies the meaning-bearing

Algorithm 1: DSyntS tree reconstruction

for $\forall n_i \in N_d$ **do**
  if $\exists n_j : (n_j, n_i) \in S_s \wedge \tau(n_j) \in N_d$ **then**
  // the equivalent of the governor node of n_i is
  // included in DSyntS
  $(n_j, n_i) \rightarrow S_d$
  else if $\exists n_j, n_a : (n_j, n_i) \in S_s \wedge \tau(n_j) \notin N_d \wedge$
      $\tau(n_a) \in N_d$ **then**
      //n_a is the first ancestor of n_j that has
      //an equivalent in DSyntS
      //the equivalent of the governor node of n_i
      //is not included in DSyntS, but the
      //ancestor n_a is
      $(n_a, n_i) \rightarrow S_d$
    else
      //the equivalent of the governor node of n_i
      //is not included in DSyntS, but several
      //ancestors of it are
      $n_b := BestHead(n_i, S_s, S_d)$
      $(n_b, n_i) \rightarrow S_d$
endfor

node as governor. Some of the dependencies of the obtained nodes $n_d \in N_{ds}$ can be recovered from the dependencies of their sources. Due to the node removals (e.g., the projection of functional nodes to grammemes), some dependencies will be also missing and must be introduced. Algorithm 1 recalculates the dependencies for the target DSyntS S_d, starting from the matrix I_s of SSyntS S_s to obtain a connected tree.

BestHead recursively ascends S_s from a given node n_i until it encounters one or several governor nodes $n_d \in N_{ds}$. In case of several encountered governor nodes, the one which governs the highest frequency dependency is returned. Consider Figure 8 for illustration.

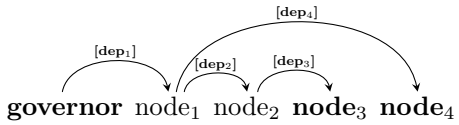


Fig. 8: A sentence in its surface representation that shows two paths: [dep$_1$] + [dep$_2$] + [dep$_3$] for the *node*$_3$ and [dep$_1$] + [dep$_4$] for *node*$_4$. The nodes *governor*, *node*$_3$ and *node*$_4$ are kept in the deep structure. The other nodes (*node*$_1$ and *node*$_2$) are not included in the deep structure. The system has to decide whether *node*$_3$ or *node*$_4$ are attached to the governor.

3.4 DSynt arc labeling

The tree reconstruction stage produces a "hybrid" connected dependency tree $S_{s \to d}$ with DSynt nodes N_{ds}, and arcs A_s labeled by SSynt relation labels (cf. left part of Figure 9), i.e., a matrix I^-, whose cells (i, j) contain SSynt labels for all $n_i, n_j \in N_{ds} : (n_i, n_j) \in A_s$ and '0' otherwise. The next and last stage of SSynt-to-DSyntS transduction is thus the projection of SSynt relation labels of $S_{s \to d}$ to their corresponding DSynt labels, or, in other words, the mapping of I^- to I_d of the target DSyntS (see Tables 2 and 3 for concrete examples).

There are some labels that have a direct transduction (see Table 2 for direct SSynt-DSynt label correspondences in Spanish), while others have several candidates. For instance, and as shown in Table 3 for Spanish, the labels *coord* and *copred* are always transduced to *COORD* and *ATTR* respectively, while *obl_obj* may be mapped to *II*, *III*, *IV*, or *VI*, depending on the other dependents of the governor of the current node. This is why it is necessary to include higher-order features based on the siblings of the node that is about to be transduced. Figure 9 shows an example of the relabeling: on the left side of the figure, the dependency labels are superficial (dep_x), whereas on the right side of the figure, the labels are the ones usually found in a deep-syntactic structure (dep_{Deepx}).

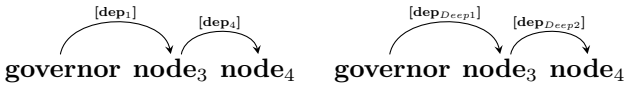


Fig. 9: Input (left) and output (right) of DSynt arc relabeling

The system learns the SSynt-to-DSynt label projection (in training time) in order to be able to infer it during the test time. The training procedure outputs a multi-class classifier that detects the best DSynt label for each node, taking into account the features that are included in the procedure. Again, this module allows for two kinds of features: local features related to a node and higher-order features related to the governor node of a node that is being processed and features related to the sibling nodes.

3.5 Postprocessing

As mentioned in Section 2, there is a limited number of idiosyncratic correspondences between elements of SSyntS and DSyntS. The correspondences (iv–vii), depicted in Section 2.2 can be straightforwardly handled by a rule-based post-processor because (a) they are non-ambiguous, i.e., $a \leftrightarrow b, c \leftrightarrow b \Rightarrow a = c \wedge a \leftrightarrow b, a \leftrightarrow d \Rightarrow b = d$, and (b) they are few. The rule-based post-processor creates/copies grammemes and creates respectively collapses some nodes in the DSyntS:

1. Tense and voice grammemes are introduced for verbal lexemes in accordance with the corresponding SSynt dependency relation (e.g., *analyt_fut*

gives rise to 'tense=FUT(ure)', *analyt_perf* to 'tense=PAST', *analyt_pass* to 'voice=PASS(ive)', etc.); definiteness grammemes are introduced for nominal lexemes (e.g., *the*←*det*- gives rise to 'def=DEF').
2. If a number or tense grammeme is already assigned to a node n_s in the SSyntS, it is copied to the node n_d corresponding to n_s in the DSyntS.
3. A reflexive verb particle that is part of the verb lemma (as, e.g., *se* in Spanish or *si* in Italian) or a pronoun (as, e.g., *sich* in German) and its verbal governor in the SSyntS are collapsed in the DSyntS into a single node.
4. If a pronoun in the SSyntS of a pro-drop language is omitted, a pronoun node is created and related to its verbal governor in the DSyntS. So far, this case has been implemented only for the zero subject in Spanish, for which the pronoun node is created, furnished with the number and person grammemes derived from the SSyntS and related to its verbal governor by an actantial relation which depends on the voice of the verb: *I* for *active* or *II* for *passive* respectively.

4 Experiments

In order to validate the SSyntS–DSyntS transduction described in Section 3 and to assess its performance in combination with a surface dependency parser, i.e., starting from a plain sentence, we carried out a number of experiments in which we implemented the transducer and integrated it into the pipeline. Figure 10 shows the whole pipeline we set up.

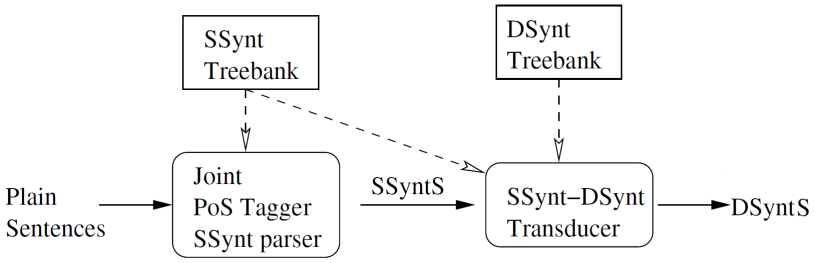


Fig. 10: Setup of a deep-syntactic parser

4.1 The SSyntS and DSyntS Treebanks

We carried out experiments on Spanish, English and Chinese.[14]

For Spanish, we use the AnCora-UPF SSyntS and DSyntS treebanks (Mille *et al.*, 2013) in CoNLL format,[15] which we adjusted for our needs. In particular, we removed from the 79-tag SSyntS treebank the semantically and information structure

[14] The Spanish treebank served us as the main source for the development of the system. Therefore, as the reader will notice below, we used a development dataset for Spanish, but not for English and Chinese.

[15] The corpus underlying both treebanks is the same: AnCora from 2008 (Taulé *et al.*, 2008).

influenced relation tags to obtain an annotation granularity closer to the granularities used for previous parsing experiments (55 relation tags, see (Mille *et al.*, 2012)). Unlike, e.g., PTB, in which syntactic (Penn TreeBank) and semantic role (PropBank/NomBank) annotations are superimposed in the same CoNLL repository, in AnCora-UPF the SSyntS and DSyntS are separate treebanks,[16] which has been validated manually (Mille *et al.*, 2013).

The treebanks have been divided into: (i) a *training set* (3,036 sentences, 57,665 tokens in the DSyntS treebank and 86,984 tokens in the SSyntS treebank); (ii) a *development set* (219 sentences, 3,271 tokens in the DSyntS treebank and 4,953 tokens in the SSyntS treebank); a (iii) a held-out *test set* for evaluation (258 sentences, 5,641 tokens in the DSyntS treebank and 8,955 tokens in the SSyntS treebank).

For English, we use Penn Treebank (PTB) 3 (Marcus *et al.*, 1994)'s dependency version (Hajič *et al.*, 2009) as SSynt annotation. To derive from it the DSynt annotation,[17] we implemented graph-transduction grammars in the MATE environment (Bohnet & Wanner, 2010).[18] The derivation removes all determiners, auxiliaries, *that* complementizers, infinitive markers *to*, punctuations and functional prepositions of verbs and predicative nouns. In order to obtain a DSynt annotation of a quality that is close to the quality of our annotation of the Spanish corpus, we used existing (manually annotated) lexical resources during the derivation, namely, PropBank (Kingsbury & Palmer, 2002) and NomBank (Meyers *et al.*, 2004).[19] In these two resources, 11,781 disambiguated predicates (5,577 nouns and 6,204 verbs) are described and their semantic roles are listed. For each of them, an important share of functional prepositions can be retrieved. To access the list of arguments of each predicate and for each argument the list of its functional prepositions, we draw upon two fields of the XML files of these resources: the last word of the field "descr" in "roles", and the first word of the field of the corresponding role in "example". In this way, we obtain, for instance, for the lexical unit *beg.01* (Figure 11), the preposition *from* for the semantic role *1*, and the preposition *for* for the role *2*. From the example field, we also retrieve *for* for the role *2*.

For each disambiguated predicate of PropBank and NomBank, we add a new entry with the semantic roles and associated functional prepositions. The resulting dictionary allows us to obtain a DSynt layer freed from around 25,000 such prepositions.

[16] The separation of SSyntS and DSyntS benefits our experiments since treebanks in which SSyntS and DSyntS are superimposed are problematic for training *pro-drop* language models (such as Spanish) because some nodes that do not appear in SSyntS are introduced in the DSyntS.

[17] For English, our derived DSyntS annotation of the PTB is the first DSyntS annotation for English.

[18] (Ribeyre *et al.*, 2014b) perform a similar automatic conversion of the French Treebank.

[19] We use PropBank and NomBank instead of VerbNet because (i) the latter covers 5 times less (2,380) predicates found in the Penn Treebank, and (ii) one predicate can be associated to more than one class, i.e., the valency pattern can be ambiguous. We are carrying out experiments on merging all these resources together, (Mille & Wanner, 2015).

```
- <roleset vncls="58.2 58.1" name="beg" id="beg.01">
  - <roles>
    - <role n="0" descr="begger, appealer">
        <vnrole vncls="58.2" vntheta="Agent"/>
        <vnrole vncls="58.1" vntheta="Agent"/>
      </role>
    - <role n="1" descr="appealed to, begged from">
        <vnrole vncls="58.2" vntheta="Recipient"/>
        <vnrole vncls="58.1" vntheta="Patient"/>
      </role>
    - <role n="2" descr="begged/appealed for">
        <vnrole vncls="58.2" vntheta="Proposition"/>
        <vnrole vncls="58.1" vntheta="Proposition"/>
      </role>
  </roles>
  - <example name="all args">
      <text> ``Just a blind fear of the unknown is causing
        them to beg the regulators for protection.'' </text>
      <arg n="0">them</arg>
      <rel>beg</rel>
      <arg n="1">the regulators</arg>
      <arg n="2">for protection</arg>
  </example>
  <note> </note>
</roleset>
```

Fig. 11: Sample PropBank entry

Table 1 shows the quality of the obtained DSynt layer. The quality figures are based on the comparison of the DSyntSs of 300 sentences (6,979 SSynt and 4976 DSynt tokens) of the PTB annotated manually with their automatically obtained equivalents. According to our error analysis, most errors of the automatic annotation are due to the fact that during the annotation, the only information that is available concerns verbs and nouns which govern preposition(s). In other words, functional prepositions governed by adjectives, adverbs or prepositions (e.g., *thanks **to***) cannot be identified automatically. Neither can be identified argument slots in genitive noun compounds, as explained in Section 4.3.1–3 for Spanish. However, the automatic annotation is still of reasonable quality that allows us to use it for our experiments.

For our experiments, we kept the same training and test dataset split as in the CoNLL Shared Task 2009 (Hajič *et al.*, 2009): 39,279 sentences for the training set and 2,399 sentences for the test set. This meant in the case of the training set 958,167 tokens in the SSyntS treebank and 711,491 tokens in the DSyntS treebank, and in the case of the test set 57,676 tokens in the SSyntS treebank and 42,467 tokens in the DSyntS treebank.

For Chinese, we use the Chinese Dependency Treebank (Xue *et al.*, 2004), which was mapped to the DSyntSs along the same lines as Penn Treebank 3, but using a graph transduction grammar tuned to Chinese syntax and without lexical resources. The mapping removes (i) aspectual markers (PoS *AS*), (ii) prepositions in beneficiary constructions (PoS *BA*), in verbal modifier constructions (PoS *DER* and *DEV*), in passives (PoS *LB*), and in verbal and nominal constructions with PoS *DEC*, *DEG*, and *P* in the case of the word *jiu*, (iii) localizers when they are

Node removal/addition	
Measure	Automatic Annotation
p_h	97.00 (4974/5128)
r_h	99.96 (4974/4976)
$F1_h$	98.46
Attachment and labeling	
Measure	Automatic Annotation
LAP	88.53 (4540/5128)
UAP	93.88 (4814/5128)
LA-P	90.00 (4615/5128)
LAR	91.24 (4540/4976)
UAR	96.74 (4814/4976)
LA-R	92.75 (4615/4976)

Table 1: Quality of the automatic annotation of the PTB with the DSyntS layer

combined with prepositions, (iv) certain particles (PoS *SP* and a subclass of *MSP*) and (v) punctuations.[20]

The Chinese treebank has been divided into a training set of 31,131 sentences (718,716 tokens in the SSyntS treebank and 553,290 tokens in the DSyntS treebank) and a test set of 10,180 sentences (241,247 tokens in the SSyntS treebank and 186,710 tokens in the DSyntS treebank).

4.2 Getting the SSyntS

To obtain the SSyntS of all three languages with which we experiment, we use (Bohnet & Nivre, 2012)'s transition-based parser, which combines PoS tagging and syntactic labeled dependency parsing. The parser uses a number of various techniques to obtain competitive accuracy such as beam search, a hash kernel that can employ a large number of features, and a graph-based completion model that re-scores the beam to capture the tree structure in terms of completed structures composed by up to three edges.

The parser was trained in 25 training iterations, using in each iteration the model from the preceding iteration for further processing. Given that the parser combines PoS and dependency parsing, we let the parser choose between the two best PoS tags. The threshold for the inclusion of PoS tags was set to a score of 0.25, and the size for the beam of the alternative PoS tags to 4.

[20] We are conscious that this fully automatically obtained treebank cannot be of highest quality and is thus not optimal as training material. However, we believe that it is useful to demonstrate that our proposal is applicable to typologically quite different languages.

4.3 From SSyntS to DSyntS

In what follows, we first present the realization of the SSyntS–DSyntS transducer and then the baseline that we use for the evaluation of the performance of the transducer. Given that we did the main development work on the Spanish treebanks, the examples in this subsection are given for Spanish and the performance reported for the development data set is for Spanish.

4.3.1 SSyntS–DSyntS transducer

As outlined in Section 3, the SSyntS–DSyntS transducer is composed of three main submodules (1. Hypernode identification, 2. Tree reconstruction, and 3. Relation label classification) and a post-processing submodule. Let us discuss each of them separately.

1. Hypernode identification: For hypernode identification, we trained a binary SVM with polynomial kernel from LIBSVM (Chang & Lin, 2001). The SVM allows for both features that are related to the processed node and higher-order features, which can be related to the governor node of the processed node or to its sibling nodes. After several feature selection trials, we chose the following features for each node n:

- lemma or stem of the label of n,
- label of the relation between n and its governor,
- surface PoS of n's label,[21]
- label of the relation between n's governor to its own governor,
- surface PoS of the label of n' governor node.

After an optimization round of the parameters available in the SVM implementation, the hypernode identification achieved over the Spanish gold development set 99.78% precision and 99.02% recall (and thus 99.4% F1).[22] That is, only very few hypernodes are not identified correctly. The main (if not the *only*) error source are *governed prepositions*; cf. Section 2: the classifier has to learn when to assign a preposition an own hypernode (i.e., when it is lexically meaning-bearing) and when it should be included into the hypernode of the verb/noun (i.e., when it is functional). Our interpretation is that the features we use for this task are appropriate, but that the training data set is too small. As a result, some prepositions are erroneously removed from or left in the DSyntS.

2. Tree reconstruction: The implementation of the tree reconstruction module shows an unlabeled dependency attachment precision of 98.18% and an unlabeled dependency attachment recall of 97.43% over the Spanish gold development set. Most of the errors produced by this module have their origin in the previous module,

[21] The SSynt and DSyntS treebanks distinguish between surface and deep PoS.
[22] For the definition of the evaluation measures we use, see Section 5.1.

that is, in the hypernode identification. When a node has been incorrectly removed, the module errs in the attachment because it cannot use the node in question as the destination or the origin of a dependency, as it is the case in the gold-standard annotation; cf., Figure 12.[23]

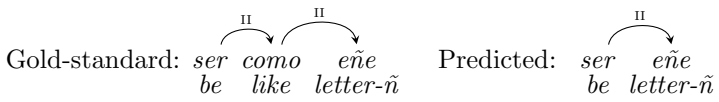


Fig. 12: Sample gold-standard and predicted DSyntSs: node erroneously removed from the DSyntS

When a node has erroneously not been removed, no dependencies between its governor and its dependent can be established since DSyntS must remain a tree (which gives the same LAS and UAS errors as when a node has been erroneously removed); cf. Figure 13.

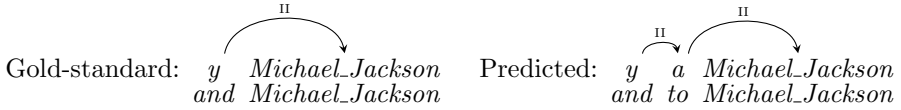


Fig. 13: Sample gold-standard and predicted DSyntSs: node erroneously left in the DSyntS

3. Relation label classification: For relation label classification, we use a multi-class linear SVM. The label classification procedure depends on the concrete annotation schemata of the SSyntS and DSyntS treebanks on which the parser is trained. Some DSynt relation labels may be easier to derive from the original SSyntS relation labels than others. In Tables 2 and 3, we summarize the DSynt relation label derivation for the Spanish treebank.[24] Table 2 lists all Spanish SSynt relation labels that have a straightforward mapping to DSyntS relation labels, i.e., (i) neither their dependent nor their governor are removed, and (ii) the SSyntS label always maps to the same DSynt label. Table 3 shows SSyntS relation–DSyntS relation label correspondences that are not straightforward.

Given that SSyntS is highly language-dependent, the SSyntS–DSyntS mappings

[23] Note that a large majority of prepositions and conjunctions that have to be removed are second arguments, and that their dependents are by chance their second arguments too. If the example given in Figure 12 gives a labeled and unlabeled attachment errors, the labeling accuracy is not impaired (the node is in both cases the dependent of an edge II). This explains why the labeling accuracy is significantly higher than the labeled attachment score (see Section 5.2).

[24] We show explicitly only the derivation for Spanish, firstly, because the SSyntS–DSyntS projection for Spanish is the most complex in our collection of treebanks. Secondly, because showing additionally the derivation for English and Chinese would not provide any further argumentation or illustration, while occupying several pages of the paper.

SSynt	DSynt
abbrev	ATTR
abs_pred	ATTR
adv	ATTR
adv_mod	ATTR
appos	ATTR
attr	ATTR
aux_phras	NAME
aux_refl_dir	II
aux_refl_indir	III
bin_junct	ATTR
compl1	II
compl2	III
compl_adnom	ATTR
coord	COORD
copul	II
copul_clitic	II
copul_quot	II
dobj_clitic	II
dobj_quot	II
elect	ATTR
juxtapos	APPEND
modal	II
modif	ATTR
num_junct	COORD
obj_copred	ATTR
prepos	II
prepos_quot	II
prolep	APPEND
quant	ATTR
quasi_coord	COORD
quasi_subj	I
relat	ATTR
restr	ATTR
sequent	ATTR
subj_copred	ATTR

Table 2: Straightforward SSynt to DSyntS DepRel mappings (Spanish)

must necessarily capture these idiosyncrasies. For instance, for Spanish as a pro-drop language we need to create in the DSyntS nodes that stand for zero subjects (i.e., subjects that do not appear in the SSyntS). Since the data-driven hypernode classifier only removes or keeps nodes, we implemented a simple rule-based approach for node creation. The system adds a node in the DSyntS when there is a finite verb that does not have a dependent which is a subject. This new node inherits

SSynt DepRel	**Mapping operations to DSynt**
agent	remove Dep; DepRel I
analyt_fut	remove Gov and Dep; add tense=FUT
analyt_pass	remove Gov; invert I and II; add voice=PASS
analyt_perf	remove Gov; add tense=PAST
analyt_progr	remove Gov; add tem_constituency=PROGR
aux_refl_lex	remove Dep; add *se* at the end of Gov's lemma
aux_refl_pass	remove Dep; invert I and II; add voice=PASS
compar	remove Dep, if conjunction; map to DepRel II
compar_conj	remove Dep, if governed preposition
coord_conj sub_conj	map to DepRel II
det	**IF Dep=el—un**: remove Dep add definiteness=DEF/INDEF **IF Dep=possessive**: map to DepRel ATTR|I|II|III **IF Dep=other**: map to DepRel ATTR
dobj	remove Dep, if governed preposition map to DepRel I|II
iobj	remove Dep, if governed preposition map to DepRel II|III|IV|V|VI
iobj_clitic	map to DepRel II|III|IV|V|VI
obl_compl	remove Dep, if governed preposition map to DepRel I|II|III|IV|V|VI
obl_obj	remove Dep, if governed preposition map to DepRel II|III|IV|V|VI
punc(_init)	remove Dep
subj	remove Dep, if governed conjunction map to DepRel I|II

Table 3: Complex SSyntS to DSyntS mappings (Spanish); 'Dep' = "dependent", 'Gov' = "governor", 'DepRel' = "DSynt dependency relation"

the person and number from the verbal governor. This strategy is fully applicable

to other languages as well since the system only needs as input the verbal PoS tag and the *subjectival* dependency relation.

The final set of features selected for label classification includes:

- lemma of the dependent node,
- dependency relation to the governor of the dependent node,
- dependency relation label of the governor node to its own governor,
- dependency relation to the governor of the sibling nodes of the dependent node, if any.

After an optimization round of the parameters set of the SVM model, relation labeling achieved 94.00% label precision and 93.28% label recall on the Spanish development set. The recall is calculated considering all the nodes that are included in the gold standard.

The error sources for relation labeling are mostly the dependencies that involve possessives and the various types of objects (see Table 3) due to their differing valency. For instance, the relation *det* in *su* 'his'/'her' $\leftarrow det-coche$ 'car' and *su* 'his'/'her' $\leftarrow det-llamada$ 'phone call' have different correspondences in DSyntS: $su \leftarrow ATTR-coche$ vs. $su \leftarrow I-llamada$. That is, the DSyntS relation depends on the lexical properties of the governor. Once again, more training data is needed in order to classify better those cases.

4. Post-processing: In the post-processing stage for Spanish, the following rules capture non-ambiguous correspondences between elements of the SSynt matrix $I_s = N_s \times N_s$ and DSyntS matrix $I_d = N_d \times N_d$, with $n_s \in N_s$ and $n_d \in N_d$, and n_s and n_d corresponding to each other (we do not list here identity correspondences such as between the number grammemes of n_s and n_d):

- if n_s is either dependent of *analyt_pass* or governor of *aux_refl_pass* relation, then the voice grammeme in n_d is *PASS*;
- if n_s is dependent of *analyt_progr*, then the voice grammeme in n_d is *PROGR*;
- if n_s is dependent of *analyt_fut*, then the tense grammeme in n_d is *FUT*;
- if n_s is governor of *aux_refl_lex*, then add the particle -SE as suffix of node label (word token) of d_d;
- if any of the children of n_s with the dependency label *det* is labeled by one of the tokens UN, UNA, UNOS or UNAS, then the definiteness grammeme in n_d is *INDEF*, and this grammeme is *DEF* for the tokens EL, LA, LOS and LAS;
- if the n_s label is a finite verb and n_s does not govern a *subject* relation, then add to I_d the relation $n_d - I \rightarrow n'_d$, with n'_d being a newly introduced node.

4.3.2 Baseline

For the evaluation of the performance of our SSyntS–DSyntS transducer, we use a rule-based SSyntS–DSyntS mapping baseline.

The baseline carries out the most direct SSynt–DSynt relation label projections following the SSyntS–DSyntS relation mapping tables compiled for each language

(see Tables 2 and 3 for Spanish). It removes all nodes which are systematically absent from the DSynt corpus (determiners, auxiliaries, infinitive markers, punctuations, etc.), and also prepositions and conjunctions involved in a dependency which indicates the possible presence of a governed preposition (e.g., *compar_conj* or *dobj* in Table 3). The baseline always produces connected trees.

The rules of the rule-based baseline look as follows:

```
1  if (deprel==abbrev) then deep_deprel=ATTR
2  if (deprel==obl_obj) then deep_deprel=II
...
n  if (deprel==punc) then remove(current_node)
```

5 Results and Discussion

To assess the performance of our SSyntS–DSyntS transducer in isolation and in a pipeline with a SSyntS parser, we carried out a number of experiments on Spanish, English and Chinese. Before we report on the performance figures obtained during these experiments, let us first introduce the evaluation measures we use.

5.1 Evaluation Measures

To measure the performance of the SSyntS–DSyntS transducer, we came up with a number of evaluation measures for hypernode detection and node attachment.

The measures for hypernode detection are:

- *Precision of the hypernode detection*: $p_h = |H_{corr}|/|H_{pred}|$ (with $|H_{corr}|$ as the number of correctly predicted hypernodes and $|H_{pred}|$ as the total number of predicted hypernodes);
- *Recall of the hypernode detection*: $r_h = |H_{corr}|/|H_g|$ (with $|H_{corr}|$ as the number of correctly predicted hypernodes and $|H_g|$ as the number of hypernodes in the gold standard);
- *F-measure of the hyper-node detection*: $F1_h = 2p_h.r_h/(p_h + r_h)$.

The measures to assess the precision of node attachment are:

- *Unlabeled attachment precision*: $UAP = |N_{governor}|/|N|$ (with $|N_{governor}|$ as the number of nodes with a correctly predicted governor, and $|N|$ as the total number of predicted nodes);
- *Label assignment precision*: $LA - P = |N_{gov.rel.label}|/|N|$ (with $|N_{gov.rel.label}|$ as the number of nodes for whose governing relation the label has been correctly predicted, and $|N|$ as the total number of predicted nodes);
- *Labeled attachment precision*: $LAP = |N_{governor.label}|/|N|$ (with $|N_{governor.label}|$ as the number of nodes with a correctly predicted governor and governing relation label, and $|N|$ as the total number of predicted nodes);

The measures to assess the recall of the node attachment are:

- *Unlabeled attachment recall*: $UAR = |N_{governor}|/|N_g|$ (with $|N_{governor}|$ as the number of nodes with a correctly predicted governor, and $|N_g|$ as the total number of gold nodes);
- *Label assignment recall*: $LA - R = |N_{gov.rel.label}|/|N_g|$ (with $|N_{gov.rel.label}|$ as the number of nodes for whose governing relation the label has been correctly predicted, and $|N_g|$ as the total number of gold nodes);
- *Labeled attachment recall*: $LAR = |N_{governor.label}|/|N_g|$ (with $|N_{governor.label}|$ as the number of nodes with a correctly predicted governor and governing relation label, and $|N_g|$ as the total number of gold nodes);

5.2 SSyntS–DSyntS transducer results

In Tables 4, 5 and 6, the performance of the subtasks of the SSyntS–DSyntS transducer for Spanish, Chinese and English respectively is contrasted to the performance of the rule-based baseline; we do not include the evaluation of the post-processing subtask for Spanish because the one-to-one projection of SSyntS elements to DSyntS captured by the rules of the post-processing submodule guarantees an accuracy of 100% of the operations performed, when starting from gold SSyntS trees.

The transducer has been applied to the gold standard test sets, which are the held-out test sets presented in Section 4.1, with gold standard PoS tags, gold-standard lemmas and gold-standard dependency trees. In the case of Spanish, the transducer outputs in total 5,610 nodes. The rule-based baseline produces an output that contains 5,902 nodes. As mentioned in Section 4.1, our gold standard includes 5,641 nodes. In the case of English, the transducer outputs in total 43,472 nodes. In this case, the rule-based baseline produces an output that contains 43,510 nodes, while the gold standard includes 43,301 nodes. Finally, for Chinese, the transducer outputs in total 186,809 nodes. The rule-based baseline produces an output with 192,078 nodes, while the gold standard has 186,710 nodes.

Our data-driven SSyntS–DSyntS transducer is significantly better than the baseline with respect to all evaluation measures. The transducer relies on distributional patterns identified in the training data set, and makes thus use of information that is not available to the rule-based baseline, which merely takes into account one node and its immediate parent at a time.

However, the rule-based baseline results also show that transduction that would remove a few nodes would obtain a performance close to a 100% recall for the hypernode detection because a DSynt tree is a subtree of the SSynt tree (if we ignore the nodes introduced by post-processing). This is also evidenced by the label and attachment recall scores.

For Spanish, which is the language we used for the system development (Ballesteros *et al.*, 2014), the results of the transducer on the test and development sets are quite comparable. For the convenience of the reader, the development set figures are repeated in Table 7.

The hypernode detection is even better on the test set. Label assignment precision and recall are the measures that suffer most from using unseen data during the

Hypernode Detection (Spanish)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	95.15 (5616/5902)	99.79 (5598/5610)
r_h	99.55 (5616/5641)	99.24 (5598/5641)
$F1_h$	97.31	99.51
Attachment and labeling (Spanish)		
Measure	Baseline	SSyntS-DSyntS Transducer
LAP	79.57 (4696/5902)	91.07 (5109/5610)
UAP	88.95 (5250/5902)	98.32 (5516/5610)
LA-P	88.74 (5006/5902)	92.37 (5182/5610)
LAR	83.25 (4696/5641)	90.57 (5109/5641)
UAR	93.07 (5250/5641)	97.78 (5516/5641)
LA-R	88.74 (5006/5641)	91.86 (5182/5641)

Table 4: Performance of the SSyntS–DSyntS transducer and of the rule-based baseline over the Spanish gold-standard held-out test set

Hypernode Detection (English)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	96.96 (42187/43510)	98.67 (42096/42663)
r_h	99.31 (42187/42480)	99.10 (42096/42480)
$F1_h$	98.12	98.88
Attachment and labeling (English)		
Measure	Baseline	SSyntS–DSyntS Transducer
LAP	86.97 (37840/43510)	90.63 (38667/42663)
UAP	90.77 (39494/43510)	93.70 (39974/42663)
LA-P	90.89 (39545/43510)	93.90 (40060/42663)
LAR	89.08 (37840/42480)	91.02 (38667/42480)
UAR	92.97 (37840/42480)	94.11 (39974/42480)
LA-R	93.09 (37840/42480)	94.30 (40060/42480)

Table 5: Performance of the SSyntS–DSyntS transducer and of the rule-based baseline over the English gold-standard held-out test set

development of the system. The attachment figures are more or less equivalent on both sets.

It is also worth noting that the Chinese results confirm that the SSyntS–DSyntS Chinese mapping is rather straightforward. This is why the baseline provides very competitive results. However, the data-driven system is capable of improving these

Hypernode Detection (Chinese)		
Measure	Baseline	SSyntS-DSyntS Transducer
p_h	97.12 (186547/192078)	99.88 (186587/186809)
r_h	99.91 (186547/186710)	99.93 (186587/186710)
$F1_h$	98.50	99.91
Attachment and labeling (Chinese)		
Measure	Baseline	SSyntS–DSyntS Transducer
LAP	96.04 (184463/192078)	99.07 (185069/186809)
UAP	97.02 (186349/192078)	99.83 (186494/186809)
LA-P	96.10 (184601/192078)	99.12 (185161/186809)
LAR	98.80 (184463/186710)	99.12 (185069/186710)
UAR	99.81 (186349/186710)	99.88 (186494/186710)
LA-R	98.87 (184601/186710)	99.17 (185161/186710)

Table 6: Performance of the SSyntS–DSyntS transducer and of the rule-based baseline over the Chinese gold-standard held-out test set

p_h	r_h	UAP	LAP	LA–P	UAR	LAR	LA–R
99.78	99.02	98.18	92.64	94.00	97.43	91.43	93.28

Table 7: Performance of the SSyntS–DSyntS transducer over the Spanish development set

results and even achieve figures that are very close to a perfect mapping. In English, the difference between the baseline and the data-driven system is significant, since, unlike in Chinese, the predicates are annotated using a manually supervised resource (see Section 4.1). The difference is even more striking with Spanish, due to the fact that the DSyntS treebank has been revised manually in several iterations (Mille *et al.*, 2013).

5.3 Results of deep-syntactic parsing

Let us consider now the performance of the complete DSynt parsing pipeline, i.e., PoS-tagger+surface-dependency parser → SSyntS–DSyntS transducer on the held-out test set. Table 8 displays the figures of the Bohnet and Nivre parser for Spanish, English and Chinese respectively. The figures are in line with the performance of state-of-the-art parsers for Spanish (Mille *et al.*, 2012), English and Chinese (Ballesteros & Bohnet, 2014). Note that for Chinese we did not predict the lemmas (there are no lemmatized forms in the treebank), but rather used gold standard forms instead.

	PoS	LEMMA	LAS	UAS
Spanish	96.05	92.10	81.45	88.09
English	98.50	99.46	89.70	92.21
Chinese	93.70		75.72	81.02

Table 8: Performance of Bohnet and Nivre's joint PoS-tagger+dependency parser trained on the Ancora-UPF treebank for Spanish, PTB treebank for English, and the CTB treebank for Chinese.

Hypernode Detection (Spanish)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	92.77 (5416/5838)	97.07 (5391/5554)
r_h	96.01 (5416/5641)	95.57 (5391/5641)
$F1_h$	94.36	96.31
Attachment and labeling (Spanish)		
Measure	Baseline	SSyntS–DSyntS Transducer
LAP	59.44 (3470/5838)	68.31 (5109/5554)
UAP	70.14 (4095/5838)	77.31 (4294/5554)
LA-P	73.93 (4316/5838)	80.47 (4469/5554)
LAR	61.51 (3470/5641)	67.26 (3794/5641)
UAR	72.59 (4095/5641)	76.12 (4294/5641)
LA-R	76.51 (4316/5641)	79.22 (4469/5641)

Table 9: Performance of the rule-based baseline and the SSyntS–DSyntS transducer over the Spanish predicted held-out test set

Tables 9, 10 and 11 show the performance of the pipeline when we feed the outputs of the syntactic parser to the rule-based baseline module and the SSyntS–DSyntS transducer for Spanish, English and Chinese, respectively.

In the case of Spanish, we observe a clear error propagation from the dependency parser (which provides 81.45% LAS) to the SSyntS–DSyntS transducer, which loses in tree quality about 18%: the difference between 90.57% (Table 4) and 67.26% LAS (Table 9) is more than 23%. For Chinese and English, we observe a similar behavior, but in this case the system is capable to recover better from the erroneous output of the surface parser. This is because the mapping from SSyntS to DSyntS is simpler, and thus the system achieves a higher performance (closer to the performance of the surface parser).

To observe the influence of the automatic conversion of the DSyntS layer of the English treebank on the quality of the SSyntS–DSyntS transducer, we ran it on a manually annotated DSyntS test set of 300 sentences over the gold surface-

Hypernode Detection (English)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	96.82 (42104/43488)	98.38 (41974/42665)
r_h	99.11 (42104/42480)	98.81 (41974/42480)
$F1_h$	97.95	98.59
Attachment and labeling (English)		
Measure	Baseline	SSynS–DSyntS Transducer
LAP	78.53 (34152/43488)	81.70 (34856/42461)
UAP	83.21 (36189/43488)	85.80 (36605/42461)
LA-P	86.36 (37557/43488)	88.93 (37941/42641)
LAR	80.40 (34152/42480)	82.05 (34856/42480)
UAR	85.19 (36189/42480)	86.17 (36605/42480)
LA-R	88.41 (37557/42480)	89.31 (37941/42480)

Table 10: Performance of the rule-based baseline and the SSyntS–DSyntS transducer over the English predicted held-out test set

Hypernode Detection (Chinese)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	97.08 (186547/191866)	99.74 (186290/186784)
r_h	99.77 (186547/186710)	99.78 (186290/186710)
$F1_h$	98.41	99.76
Attachment and labeling (Chinese)		
Measure	Baseline	SSyntS–DSyntS Transducer
LAP	75.00 (143903/191866)	77.18 (144163/186784)
UAP	77.92 (149497/191866)	80.06 (149530/186784)
LA-P	85.66 (164348/191866)	88.18 (164696/186784)
LAR	77.07 (143903/186710)	77.21 (144163/186710)
UAR	80.07 (149497/186710)	80.09 (149530/186710)
LA-R	88.02 (164348/186710)	88.21 (164696/186710)

Table 11: Performance of the rule-based baseline and the SSyntS–DSyntS transducer over the Chinese predicted held-out test set

standard held-out test set (Table 12) and over the predicted surface-standard held-out test set (Table 13). Compared to the performance on the automatically obtained DSyntS test set (see Tables 5 and 10), the performance is somewhat lower (due to the fact that manually annotated, i.e., ideal, DSyntSs are more diverging from the SSyntSs than automatically derived ones). However, it is still high enough to provide reasonably well-formed and correct DSyntSs for downstream applications.

Hypernode Detection (English)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	94.46 (4918/5249)	95.72 (4918/5138)
r_h	99.64 (4958/4976)	98.83 (4918/4976)
$F1_h$	96.98	97.25
Attachment and labeling (English)		
Measure	Baseline	SSynS–DSyntS Transducer
LAP	78.66 (4129/5249)	80.03 (4112/5138)
UAP	86.11 (4520/5249)	87.27 (4484/5138)
LA-P	82.98 (4356/5249)	84.84 (4359/5138)
LAR	82.98 (4112/4976)	82.64 (4112/4976)
UAR	90.84 (4520/4976)	90.11 (4484/4976)
LA-R	87.54 (4356/4976)	87.60 (4359/4976)

Table 12: Performance of the rule-based baseline and the SSyntS–DSyntS transducer over the English surface gold-standard held-out test set and the manually annotated DSyntS test set

As we observe in Tables 12 and 13, the recall of the rule-based baseline is a little bit higher than the one obtained with the machine learning approach, however the precision is much higher for the machine-learning system. Since the output trees of the rule-based baseline have more nodes, it provides a more recall oriented system, but it suffers more in precision, leading to lower F1 scores for all measures. Moreover, since the machine learning model is trained on partially (and not fully manually) validated sentences, the parser tries to predict the annotation provided in the partially validated sentences.

6 Related Work

As already pointed out in Section 1, the idea of deep parsing is not novel: it goes back at least to (Curry, 1961) and has already been addressed in some depth in the early days of Natural Language Processing in the context of *language understanding* (Bobrow & Webber, 1981; Dahlgren, 1988). Over the years, some authors continued to work on rule-based proposals for deep parsing in different theoretical frameworks; cf., among others, (Rambow & Joshi, 1997) for a deep analysis proposal in the TAG-framework, (de Groote, 2001) for a proposal in the CCG framework, and (Klimeš, 2006) for a proposal in the Prague School Dependency framework. More recently, the importance of deep linguistic processing for parsing has been reiterated, e.g., by (Baldwin *et al.*, 2007). However, to the best of our knowledge, data-driven deep-syntactic parsing as proposed in this article is novel.

As data-driven *semantic role labeling*, *frame-semantic analysis*, and logical form analysis, DSynt parsing has the goal to obtain more semantically oriented struc-

Hypernode Detection (English)		
Measure	Baseline	SSyntS–DSyntS Transducer
p_h	93.34 (4952/5249)	95.53 (4912/5142)
r_h	99.52 (4952/4976)	98.71 (4912/4976)
$F1_h$	96.86	97.09
Attachment and labeling (English)		
Measure	Baseline	SSynS–DSyntS Transducer
LAP	70.81 (3717/5249)	72.19 (3712/5142)
UAP	78.83 (4138/5249)	79.75 (4101/5142)
LA-P	78.83 (4138/5249)	80.69 (4149/5142)
LAR	74.70 (3717/4976)	74.60 (3712/4976)
UAR	83.16 (4138/4976)	82.42 (4101/4976)
LA-R	83.16 (4138/4976)	83.38 (4149/4976)

Table 13: Performance of the rule-based baseline and the SSyntS–DSyntS transducer over the English surface predicted held-out test set and the manually annotated DSyntS test set

tures than those delivered by state-of-the-art syntactic parsing (McDonald *et al.*, 2005; Nivre *et al.*, 2007b; Kübler *et al.*, 2009; Bohnet & Kuhn, 2012; Bohnet & Nivre, 2012). Semantic role labeling received considerable attention in the CoNLL shared tasks for syntactic dependency parsing in 2006 and 2007 (Buchholz & Marsi, 2006; Nivre *et al.*, 2007a), the CoNLL shared task for joint parsing of syntactic and semantic dependencies in 2008 (Surdeanu *et al.*, 2008) and the shared task in 2009 (Hajič *et al.*, 2009). The top ranked systems were pipelines that started with a syntactic analysis (as we do) and continued with predicate identification, argument identification, argument labeling, and word sense disambiguation (WSD); cf. (Johansson & Nugues, 2008b; Che *et al.*, 2009). At the end, a re-ranker that considers jointly all arguments to select the best combination was applied. Some of the systems were based on integrated syntactic and semantic dependency analysis; cf., e.g., (Gesmundo *et al.*, 2009); see also (Lluís *et al.*, 2013) for a more recent proposal along similar lines. However, all of them lack the ability to perform necessary structural changes—as, e.g., introduction of nodes or removal of nodes necessary to obtain a DSyntS.

Logical form analyzers such as Boxer (Bos, 2008) tend also to pipeline syntactic and deep parsing, as we do. In the case of Boxer, a CCG parser is integrated into a pipeline with the Discourse Representation Structure (DRS) parser. However, they output abstract structures that are void of any syntactic dependencies—which can however be important for some applications (such as Machine Translation).

Finally, even though, as discussed in Section 2.1, the deep structures used in SemEval 2014 (Oepen *et al.*, 2014) are different from DSyntSs, the systems solve a similar problem. Among the best performing systems, are Priberam (Martins &

Almeida, 2014) and CMU (Flanigan *et al.*, 2014), which follow graph-based approaches. Alpage (Ribeyre *et al.*, 2014a) and Peking (Du *et al.*, 2014) are similar to our approach since they propose transition-based parsing algorithms for DAGs, similar to (Sagae & Tsujii, 2008), where the usual set of transitions is different in each task. Both Alpage and Peking transform graphs into trees. Turku (Kanerva *et al.*, 2014) is also similar to our proposal since it works with a cascade of classifiers. In contrast to Turku, however, we present a joint transition-based dependency parser tagger for getting the SSyntS from plain text sentences and a cascade of classifiers to transduce the SSyntS then to the DSyntS.

7 Conclusions and Future Work

We have presented a novel data-driven deep-syntactic parsing pipeline which consists of a state-of-the-art dependency parser and a SSyntS–DSyntS transducer. The DSyntSs provided by the pipeline can be used in different applications since they abstract from language-specific grammatical idiosyncrasies of the SSynt structures as produced by state-of-the art dependency parsers, but still avoid the complexities of genuine semantic analysis. DSyntS treebanks needed for data-driven applications can be bootstrapped by the pipeline. If required, a SSyntS–DSyntS structure pair can be also mapped to a pure predicate-argument graph such as the Enju conversion (Miyao, 2006), to an DRS (Kamp & Reyle, 1993), or to a PropBank structure. An online demo (Soler-Company *et al.*, 2015) of our DSynt parser is available at `http://dparse.multisensor.taln.upf.edu/main`.

In the future, we will carry out further in-depth feature engineering for the task of DSynt parsing. It proved to be crucial in semantic role labeling and dependency parsing (Che *et al.*, 2009; Ballesteros & Nivre, 2012); we expect it be essential for our task as well. Furthermore, we will join surface syntactic and deep-syntactic parsing we kept so far separately; see, e.g., (Zhang & Clark, 2008; Lluís *et al.*, 2013; Bohnet & Nivre, 2012) for analogous proposals. Further research is required here since although joined models avoid error propagation from the first stage to the second, they need to bridge a broader abstraction moat—which is why pipelined models still prove to be competitive; cf. the outcome of CoNLL shared tasks.

We will try to improve the English and Chinese DSyntS treebanks we obtained by automatic conversion in order to make them genuine DSyntS treebanks (and thus more comparable to the Spanish DSyntS treebank we work with). This will allow our DSynt parser to also provide genuine DSyntSs, with no traces of SSyntSs left in its output.

Finally, our DSynt parser could be used towards machine translation or summarization by using it jointly with a DSyntS generator such as the one presented by Ballesteros *et al.* (2015).

Acknowledgements

We would like to thank the reviewers for their insightful comments and Alicia Burga for her help with the revision of the paper. The work reported on in this paper has

been partially funded by the European Commission under the contract numbers FP7–ICT–610411 (MULTISENSOR) and H2020–645012–RIA (KRISTINA).

References

Allen, J., Dzikovska, M., Manshadi, M., & Swift, M. 2007. Deep Linguistic Processing for Spoken Dialogue Systems. *Pages 49–56 of: Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing.*

Baldwin, Timothy, Dras, Mark, Hockenmaier, Julia, King, Tracy Holloway, & van Noord, Gertjan. 2007. The Impact of Deep Linguistic Processing on Parsing Technology. *Pages 36–38 of: Proceedings of the 10th Conference on Parsing Technologies.* Association for Computational Linguistics.

Ballesteros, Miguel, & Bohnet, Bernd. 2014. Automatic Feature Selection for Agenda-Based Dependency Parsing. *In: Proceedings of the International Conference on Computational Linguistics (COLING 14).*

Ballesteros, Miguel, & Nivre, Joakim. 2012. MaltOptimizer: A System for MaltParser Optimization. *In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC).*

Ballesteros, Miguel, Bohnet, Bernd, Mille, Simon, & Wanner, Leo. 2014. Deep-Syntactic Parsing. *In: Proceedings of the International Conference on Computational Linguistics (COLING 14).*

Ballesteros, Miguel, Bohnet, Bernd, Mille, Simon, & Wanner, Leo. 2015. Data-driven sentence generation with non-isomorphic trees. *In: Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational linguistics – Human Language Technologies (NAACL HLT 2015).*

Bobrow, R., & Webber, B. 1981. Some Issues in Parsing and Natural Language Understanding. *In: Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics.* Stroudsburg, PA, USA: Association for Computational Linguistics.

Bohnet, B., & Wanner, L. 2010. Open Source Graph Transducer Interpreter and Grammar Development Environment. *In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010).*

Bohnet, Bernd, & Kuhn, Jonas. 2012. The Best of Both Worlds–A Graph-Based Completion Model for Transition-Based Parsers. *In: Proceedings of the Biannual Meeting of the European Chapter of the Association for Computational Linguistics.*

Bohnet, Bernd, & Nivre, Joakim. 2012. A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. *In: Proceedings of the Conference on Empirical Methods in NLP (EMNLP).*

Bojar, J.O., Cinková, S., & Ptáček, J. 2008. Towards English-to-Czech MT via Tectogrammatical Layer. *The Prague Bulletin of Mathematical Linguistics*, **90**, 57–68.

Bos, Johan. 2008. Wide-Coverage Semantic Analysis with Boxer. *Pages 277–286 of:* Bos, Johan, & Delmonte, Rodolfo (eds), *Semantics in Text Processing. STEP 2008 Conference Proceedings.* Research in Computational Semantics. College Publications.

Buch-Kromann, Mathias. 2003. The Danish Dependency Treebank and the DTAG Treebank Tool. *Pages 217–220 of: Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT).*

Buchholz, Sabine, & Marsi, Erwin. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. *Pages 149–164 of: Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL).*

Chang, Chih-Chung, & Lin, Chih-Jen. 2001. *LIBSVM: A library for support vector machines.* Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Che, Wanxiang, Li, Zhenghua, Li, Yongqiang, Guo, Yuhang, Qin, Bing, & Liu, Ting. 2009. Multilingual Dependency-Based Syntactic and Semantic Parsing. *Pages*

49–54 of: Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task. Boulder, Colorado: Association for Computational Linguistics.

Curry, H. B. 1961. Some logical aspects of grammatical structure. *Pages 56–68 of:* Jacobson, R. (ed), *Structure of language and its mathematical aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*. American Mathematical Society.

Dahlgren, K. 1988. *Naive Semantics for Natural Language Understanding*. Dordrecht, NL: Kluwer Academic Publishers.

Das, Dipanjan, Chen, Desai, Martins, André FT, Schneider, Nathan, & Smith, Noah A. 2014. Frame-semantic parsing. *Computational linguistics*, **40**(1), 9–56.

de Groote, Ph. 2001. Towards abstract categorial grammar. *In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

de Marneffe, Marie-Catherine, & Manning, Christopher D. 2008 (August). The Stanford Typed Dependencies Representation. *Pages 1–8 of: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation at the International Conference on Computational Linguistics (COLING)*.

Du, Yantao, Zhang, Fan, Sun, Weiwei, & Wan, Xiaojun. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. *Semeval 2014*, 459.

Dyer, Chris, Ballesteros, Miguel, Ling, Wang, Matthews, Austin, & Smith, Noah A. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. *In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL 2015)*. Association for Computational Linguistics.

Fillmore, Charles J., Baker, Collin F., & Sato, Hiroaki. 2002. The FrameNet Database and Software Tools. *In: Proceedings of the Third International Conference on Language Resources and Evaluation*, vol. IV. Las Palmas: LREC.

Flanigan, Jeffrey, Thomson, Sam, OConnor, Brendan, Bamman, David, Schneider, Nathan, Dodge, Jesse, Swayamdipta, Swabha, Dyer, Chris, & Smith, Noah. 2014. CMU: Arc-Factored, Discriminative Semantic Dependency Parsing. *SemEval 2014*, 176.

Gesmundo, A., Henderson, J., Merlo, P., & Titov, I. 2009. Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages. *Pages 37–42 of: Proceedings of the CoNLL 2009 Shared Task, Conference on Computational Natural Language Learning*.

Hajič, Jan, Panevová, Jarmila, Hajičová, Eva, Sgall, Petr, Pajas, Petr, Štěpánek, Jan, Havelka, Jiří, Mikulová, Marie, & Žabokrtský, Zdeněk. 2006. *Prague Dependency Treebank 2.0. Linguistic Data Consortium, Philadelphia*.

Hajič, Jan, Ciaramita, Massimiliano, Johansson, Richard, Kawahara, Daisuke, Martí, Maria Antònia, Màrquez, Lluís, Meyers, Adam, Nivre, Joakim, Padó, Sebastian, Štěpánek, Jan, Straňák, Pavel, Surdeanu, Mihai, Xue, Nianwen, & Zhang, Yi. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. *Pages 1–18 of: Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*.

Henderson, J., P.Merlo, I.Titov, & G.Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational linguistics*, **39**(4).

Ivanova, Angelina, Oepen, Stephan, Øvrelid, Lilja, & Flickinger, Dan. 2012. Who Did What to Whom? A Contrastive Study of Syntactico-Semantic Dependencies. *Pages 2–11 of: Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea: Association for Computational Linguistics.

Johansson, R., & Nugues, P. 2007. Extended Constituent-to-Dependency Conversion for English. *Pages 105–112 of:* Nivre, J., Kaalep, H.-J., Muischnek, K., & Koit, M. (eds), *Proceedings of NODALIDA 2007*.

Johansson, Richard, & Nugues, Pierre. 2008a. Dependency-Based Semantic Role Labeling of PropBank. *Pages 69–78 of: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing.*

Johansson, Richard, & Nugues, Pierre. 2008b. Dependency-based syntactic–semantic analysis with PropBank and NomBank. *Pages 183–187 of: Conll 2008: Proceedings of the twelfth conference on natural language learning.*

Kamp, H., & Reyle, U. 1993. *From Discourse to Logic.* Dordrecht, NL: Kluwer Academic Publishers.

Kanerva, Jenna, Luotolahti, Juhani, & Ginter, Filip. 2014. Turku: Broad-coverage semantic parsing with rich features. *Semeval 2014*, 678.

Kasper, R., & Hovy, E.H. 1990. Performing Integrated Syntactic and Semantic Parsing Using Classification. *Pages 54–59 of: Proceedings of the Workshop on Speech and Natural Language at HLT '90.* Stroudsburg, PA, USA: Association for Computational Linguistics.

Kasper, R.T., & Rounds, W.C. 1986. A logical semantics for feature structures. *Pages 257–266 of: Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics.*

Kingsbury, P., & Palmer, M. 2002. From Treebank to PropBank. *In: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC).*

Klimeš, Václav. 2006. Transformation-based tectogrammatical analysis of czech. *Pages 135–142 of: Text, speech and dialogue.* Springer.

Klimeš, Václav. 2007. Transformation-based tectogrammatical dependency analysis of english. *Pages 15–22 of: Text, speech and dialogue.* Springer.

Kübler, Sandra, McDonald, Ryan, & Nivre, Joakim. 2009. *Dependency parsing.* Morgan and Claypool.

Lluís, Xavier, Carreras, Xavier, & Màrquez, Lluís. 2013. Joint Arc-Factored Parsing of Syntactic and Semantic Dependencies. *Transactions of the Association for Computational Linguistics*, 219–230.

Marcus, Mitchell P., Santorini, Beatrice, Marcinkiewicz, Mary Ann, MacIntyre, Robert, Bies, Ann, Ferguson, Mark, Katz, Karen, & Schasberger, Britta. 1994. The Penn Treebank: Annotating predicate-argument structure. *Pages 114–119 of: Human language technologies.*

Martins, André FT, & Almeida, Mariana SC. 2014. Priberam: A turbo semantic parser with second order features. *Semeval 2014*, 471.

McDonald, Ryan, Pereira, Fernando, Ribarov, Kiril, & Hajič, Jan. 2005. Non-Projective Dependency Parsing Using Spanning Tree Algorithms. *Pages 523–530 of: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing.* HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics.

Mel'čuk, Igor. 1988. *Dependency Syntax: Theory and Practice.* State University of New York Press.

Mel'čuk, I. 2013. *Semantics: From meaning to text, Volume 2.* Amsterdam: Benjamins Academic Publishers.

Mel'čuk, I., & Wanner, L. 2006. Syntactic mismatches in maschine translation. *Machine translation*, **20**, 81–138.

Mel'čuk, I., & Wanner, L. 2008. Morphological mismatches in maschine translation. *Machine translation*, **22**, 101–152.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., & Grishman, R. 2004. The NomBank Project: An Interim Report. *In: Proceedings of the Workshop on Frontiers in Corpus Annotation, held in connection with the Annual Conference of the North American Chapter of the Association for Computational Linguistics.*

Mille, Simon, & Wanner, Leo. 2015. Towards large-coverage detailed lexical resources for data-to-text generation. *In: Proceedings of the First International Workshop on Data-to-Text Generation.*

Mille, Simon, Burga, Alicia, Ferraro, Gabriela, & Wanner, Leo. 2012. How Does the Granularity of an Annotation Scheme Influence Dependency Parsing Performance? *In: Proceedings of the International Conference on Computational Linguistics (COLING).*

Mille, Simon, Burga, Alicia, & Wanner, Leo. 2013. AnCora-UPF: A Multi-Level Annotation of Spanish . *In: Proceedings of the Second International Conference on Dependency Linguistics (DEPLING 2013).*

Miyao, Yusuke. 2006. *From linguistic theory to syntactic analysis: Corpus-oriented grammar development and feature forest model.* Ph.D. thesis, The University of Tokyo.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. 2007a. The CoNLL 2007 Shared Task on Dependency Parsing. *Pages 915–932 of: Proceedings of the CoNLL-ST-07.*

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., & Marsi, E. 2007b. Maltparser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, **13**, 95–135.

Oepen, Stephan, & Lønning, Jan Tore. 2006. Discriminant-Based MRS Banking. *In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006).*

Oepen, Stephan, Kuhlmann, Marco, Miyao, Yusuke, Zeman, Daniel, Flickinger, Dan, Hajic, Jan, Ivanova, Angelina, & Zhang, Yi. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. *Pages 63–72 of: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).*

Palmer, Martha. 2009. Semlink: Linking Propbank, VerbNet and FrameNet. *In: Proceedings of the Generative Lexicon Conference (GenLex-09).*

Palmer, Martha, Gildea, Daniel, & Kingsbury, Paul. 2005. The proposition bank. *Computational linguistics*, **31**, 71–106.

Rambow, O., & Joshi, A. 1997. A formal look at dependency grammar and phrase structure grammars, with special consideration of word-order phenomena. *Pages 167–190 of:* Wanner, L. (ed), *Recent Trends in Meaning-Text Theory.* Amsterdam: Benjamins Academic Publishers.

Ribeyre, Corentin, De La Clergerie, Éric Villemonte, & Seddah, Djamé. 2014a. Alpage: Transition-based semantic graph parsing with syntactic features. *In: Proceedings of the International Workshop on Semantic Evaluation.*

Ribeyre, Corentin, Candito, Marie, & Seddah, Djamé. 2014b. Semi-automatic deep syntactic annotations of the french treebank. *Pages 184–197 of: Treebanks and linguistic theories.*

Sagae, Kenji, & Tsujii, Jun'ichi. 2008. Shift-reduce dependency dag parsing. *Pages 753–760 of: Proceedings of the 22nd international conference on computational linguistics-volume 1.* Association for Computational Linguistics.

Schuler, Karin Kipper. 2005. *Verbnet: A broad-coverage, comprehensive verb lexicon.* Ph.D. thesis, University of Pennsylvania.

Soler-Company, Juan, Ballesteros, Miguel, Bohnet, Bernd, Mille, Simon, & Wanner, Leo. 2015. Visualizing deep-syntactic parser output. *In: Proceedings of the Annual Meeting of the North-American Chapter of the ACL – Human Language Technologies (NAACL - HLT 2015), Demonstrations Track.*

Steedman, Mark. 2000. *The syntactic process.* MIT Press.

Surdeanu, Mihai, Johansson, Richard, Meyers, Adam, Màrquez, Lluís, & Nivre, Joakim. 2008. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. *Pages 159–177 of: Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL).*

Taulé, M., Martí, M. Antònia, & Recasens, Marta. 2008. Ancora: Multilevel Annotated Corpora for Catalan and Spanish. *In: Proceedings of the Sixth International Language Resources and Evaluation (LREC '08)*. Marrakech, Morocco: European Language Resources Association (ELRA).

Xue, Naiwen, Xia, Fei, Chiou, Fu-Dong, & Palmer, Martha. 2004. The Penn Chinese Treebank: Phase Structure Annotation of a Large Corpus. *Natural Language Engineering*, **11**, 207–238.

Zhang, Yue, & Clark, Stephen. 2008. Joint Word Segmentation and POS Tagging Using a Single Perceptron. *Pages 888–896 of: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Columbus, Ohio: Association for Computational Linguistics.

Zhao, Hai, Chen, Wenliang, Kity, Chunyu, & Zhou, Guodong. 2009. Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing. *Pages 55–60 of: Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task*. Boulder, Colorado: Association for Computational Linguistics.