UPF at EPE 2017: Transduction-based Deep Analysis

Simon Mille [1], **Roberto Carlini** [1], **Ivan Latorre** [1], **Leo Wanner** [1,2]
[1]Universitat Pompeu Fabra, Roc Boronat 138, 08018 Barcelona, Spain
[2]Institució Catalana de Recerca i Estudis Avançats (ICREA),
Lluis Companys 23, 08010 Barcelona, Spain
firstname.lastname@upf.edu

Abstract

This paper describes the runs submitted to EPE 2017 by Universitat Pompeu Fabra. The three outputs correspond to three different levels of linguistic abstraction: (i) a surface-syntactic tree, (ii) a deep-syntactic tree, and (iii) a predicate-argument graph. The surface-syntactic tree is obtained with an off-the-shelf parser trained on the CoNLL'09 Penn Treebank, and the deeper representations by running a sequence of graph transduction grammars on the output of the parser.

1 Credits

The work described in this paper has been partially funded by the European Commission under the contract numbers H2020-645012-RIA, H2020-700024-RIA, and H2020-700475-RIA.

2 Introduction

The Extrinsic Parser Evaluation shared task (Oepen et al., 2017)[1] aims at evaluating different dependency representations from the perspective of three downstream applications: biological event extraction, negation scope resolution, and fine-grained opinion analysis; see (Björne et al., 2017; Lapponi et al., 2017; Johansson, 2017) respectively for the descriptions. The NLP group at UPF (UPF-TALN) submitted three different system outputs ("runs") to be used as input to the selected applications; each of the outputs corresponds to a different level of abstraction of the linguistic description:

- **SSynt**: surface-syntactic structures (SSyntSs), i.e., syntactic trees with fine-grained relations over all the words of a sentence;
- **DSynt**: deep-syntactic structures (DSyntSs), i.e., syntactic trees with coarse-grained relations over the meaning-bearing units of a sentence;
- **PredArg**: predicate-argument structures (PerdArgSs), i.e., directed acyclic graphs with predicate-argument relations over the meaning-bearing units of a sentence.

[1]http://epe.nlpl.eu

This stratified view largely follows the model of the Meaning-Text Theory (MTT); see, e.g., (Mel'čuk, 1988) for more details on the definition of the different types of structures. The MTT model supports fine-grained annotation at the three main levels of the linguistic description of written language: semantics, syntax and morphology, while facilitating a coherent transition between them via intermediate levels of deep-syntax and deep-morphology. At each level, a clearly defined type of linguistic phenomena is described in terms of distinct dependency structures.

The first representation is obtained with a statistical dependency parser, on top of which rule-based graph transduction grammars are applied, similarly to, e.g., the conversions in (Ribeyre et al., 2012) and (Schuster and Manning, 2016).

The idea behind submitting three very different types of outputs is to see to what extent the downstream applications chosen by the organizers of the shared task are sensitive to the variations in the linguistic representation. In what follows, we describe the targeted dependency structures and the respective systems used to obtain them. We then discuss briefly the results.

3 Run 1: Surface-syntactic trees

3.1 Targeted dependency representation

For the surface-syntactic (SSynt) annotation, many annotation schemes and parsers are avail-

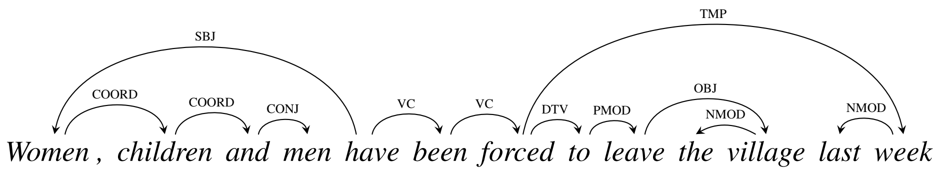

SBJ
COORD
COORD
CONJ
VC
VC
TMP
DTV
PMOD
OBJ
NMOD
NMOD
Women , children and men have been forced to leave the village last week

Figure 1: SSyntS for *Women, children and men have been forced to leave the village last week.*

able. We chose to use the representation followed in the CoNLL'09 shared task on dependency parsing (Hajič et al., 2009), because we believe that it is one of the most syntactically sound representations that are available; in particular:

(i) Its dependency tagset is fine-grained enough to take into account the most basic syntactic properties of English (37 different labels, without counting composed and gapped relations).

(ii) One lexeme corresponds to one and only one node in the tree. For instance, in a relative clause, the relative pronoun is viewed from the perspective of its function in the relative clause and not from the perspective of its conjunctive properties.

(iii) The subject is a dependent of the inflected top verb, not of the non-finite verb, which might also occur in the sentence. This accounts for the syntactic agreement that holds between the auxiliary and the subject; the relation between the non-finite verb and the subject is more of a "semantic" one, and thus made explicit at a higher level of abstraction. The finite verb in an auxiliary construction is a dependent of the closest auxiliary.

(iv) Subordinating and coordinating conjunctions depend on the governor of the first group, and govern the one of the second group. This hierarchical approach accounts for the linking properties of conjunctions. The only exception to this are the relative pronouns, as mentioned above.

Another advantage of the SSynt target representation is that it facilitates the mapping to the abstract structures used in Runs 2 and 3.

3.2 Implementation

The surface syntactic (SSynt) analysis is performed in three steps, including two preprocessing steps and the proper parsing. First, the raw text needs to be broken down into sentences, and the sentences into tokens, as the surface syntactic parser runs at sentence level and takes a one-word-per-line format as input. For this task, we use the Stanford Core NLP sentence splitter and tokenizer.[2]

Then, in order to match the training data of the syntactic parser, we replace some punctuation marks that cannot be found in the training set with equivalents that are present. For example, left and right single quotation marks are replaced by one single straight quotation mark; double quotation marks are replaced by two single straight quotation marks; the different types of dashes are all replaced by a classic dash; square brackets are replaced by round brackets; etc. If these substitutions do not take place, the parser tends to assign proper noun tags to all unknown symbols, which affects negatively the quality of the resulting structure.

UAS	LAS	PoS
93.67	92.68	97.42

Table 1: Reported accuracy scores for Bohnet and Nivre's system (Unlabeled and Labeled Attachment Scores and PoS tagging accuracy).

Module	Toolkit used
Sentence splitting	Stanford Core NLP
Tokenization	Stanford Core NLP
Character normalization	In-house Script
Joint tagging and parsing	(Bohnet and Nivre, 2012)
Speed	≈ 65 ms/sentence
Memory used	≈ 4GB

Table 2: Steps for surface-syntactic parsing

Finally, for lemmatizing, tagging and parsing, we use the joint tagger and parser described in (Bohnet and Nivre, 2012)[3], which was trained on the CoNLL'09 dataset (Hajič et al., 2009), and

[2]https://nlp.stanford.edu/software/
[3]https://code.google.com/archive/p/mate-tools/downloads

evaluated on Section 23 of the WSJ; see Table 1. Table 2 summarizes the different steps followed for this run.

4 Run 2: Deep-syntactic trees

4.1 Targeted dependency representation

Deep syntactic (DSynt) structures are dependency structures that capture the argumentative, attributive and coordinative relations between full words (*lexemes*) of a sentence. Compared to SSynt structures, in DSynt structures, functional prepositions and conjunctions, auxiliaries, modals, and determiners are removed. Each lexeme is associated with attribute/value pairs that encode such information as part of speech, verbal finiteness, modality, aspect, tense, nominal definiteness, etc. The nodes are labeled with lemmas; in addition, they are aligned with the surface nodes through attribute/value pairs (each DSynt node points to one or more SSynt node, using the surface IDs). All nodes have a PoS feature, which is copied from the SSynt output. The resulting English annotation is the same as found in the AnCora-UPF treebank of Spanish (Mille et al., 2013).

The abstraction degree of the DSynt structures is in between the output of a syntactic dependency parser as in Run 1 and the output of a semantic role labeler as in Run 3: on the one hand, they maintain the information about the syntactic structure and relations, but, on the other hand, dependency labels are oriented towards predicate-argument relations, and the dependencies directly connect meaning-bearing units, that is, meaning-void functional elements are not available anymore. Predicate-argument relations include **I**, **II**, **III**, **IV**, **V**, **VI**; modifier relations include **ATTR** and **APPEND** (the latter is used for modifiers that generally correspond to peripheral adjuncts); the other two relations are **COORD** (for coordinations) and **NAME** (connecting parts of proper nouns).

The degree of "semanticity" of DSynt structures can be directly compared to Prague's tectogrammatical structures (PDT-tecto (Hajič et al., 2006), from which the *PSD* runs by some EPE participants stem from), which contain *autosemantic* words only. Thanks to the distinction between argumental and non-argumental edges, tectogrammatical structures are also trees, thus they maintain the syntactic structure of the sentence. The main differences between the two representations 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.

Although the annotations are not really of the same nature, DSynt structures can be also contrasted to the *Collapsed Stanford Dependencies (SD)* (de Marneffe and Manning, 2008). Collapsed SDs differ from DSynt structures in that: (i) in the same fashion as in the Prague Dependency Treebank, they collapse only (but all) prepositions, conjunctions and possessive clitics, whereas DSynt structures omit functional nodes; (ii) they do not involve any removal of (syntactic) information since the meaning of the preposition remains encoded in the label of the collapsed dependency, while DSynt structures omit or generalize the purely functional elements; (iii) they do not add predicate-argument information compared to the surface annotation. That is, Collapsed SDs keep the surface-syntactic information, representing it in a different format, while DSynt structures keep only deep-syntactic information.

4.2 Implementation

In order to obtain DSynt structures, we run a sequence of rule-based graph transducers on the output of the SSynt parser. Our graph-transduction grammars are thus rules that apply to a subgraph of the input structure and produce a part of the output structure. During the application of the rules, both the input structure (covered by the *leftside* of the rule) and the current state of the output structure at the moment of application of a rule (i.e., the *rightside* of the rule) are available as context. The output structure in one transduction is built incrementally: the rules are all evaluated, the ones that match a part of the input graph are applied, and a first piece of the output graph is built; then the rules are evaluated again, this time with the rightside context as well, and another part of the output graph is built; and so on; cf. (Bohnet and Wanner, 2010). The transduction is over when no rule is left that matches the combination of the leftside and the rightside.

The SSynt-DSynt mapping is based on the notion of *hypernode*. A hypernode, known as *syntagm* in linguistics, is any surface-syntactic configuration with a cardinality ≥ 1 that corresponds

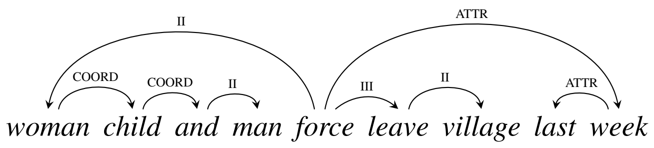


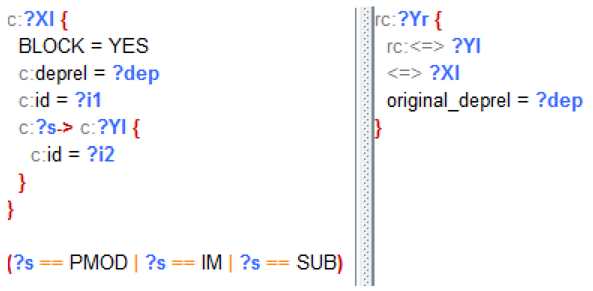
Figure 2: DSyntS for *Women, children and men have been forced to leave the village last week.*

Grammars	#rul.	Description
ALL	165	
Pre-Proc. 1	15	Assign default PB/NB IDs. Mark passive, genitive, possessive constructions.
Pre-Proc. 2	17	Mark hypernodes.
SSynt-DSynt	55	Wrap hypernodes. Assign DSynt dependencies. Transfer aspect/modality as attr. Mark duplicate relations. Mark relative clauses.
Post-Proc.	78	Relabel duplicate relations. Reestablish gapped elements. Mark coord. constructions.
Speed	≈ 25 ms/sentence	
Memory used	≈ 300MB	

Table 3: Rules for SSynt-DSynt mapping

to a single deep-syntactic node. For example, *to leave* or *the village* constitute hypernodes that correspond to the DSynt nodes *leave* and *village* respectively (see Figures 1 and 2). Hypernodes can also contain more than two nodes, as in the case of more complex analytical verb forms, e.g., *have been forced*, which corresponds to the node *force* in the DSyntS of Figure 2. 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 subtasks: (i) hypernode identification, (ii) DSynt tree reconstruction, and (iii) DSynt arc labeling.[4]

Table 3 shows the different steps of the SSynt–DSynt mapping. During a two-step pre-processing, specific constructions and hypernodes are marked. Auxiliaries, meaning-void conjunctions and determiners are easy to identify, but to know which prepositions belong to the valency pattern (subcategorization frame) of their governor, we need to consult a lexicon extracted from PropBank (Palmer et al., 2005), and NomBank (Meyers et al., 2004).[5] The output of these pre-processing steps is still a SSynt structure. The third transduction is the core of this module: it "wraps" the hypernodes into a single node and manages the labeling of the edges, again looking at the PropBank-based lexicon (i.e., at the valency pattern of the predicates), together with the surface dependencies. For instance, a subject of a passive verb is mapped to a first argument (*I*), while the subject of a passive verb is mapped to a second argument (*II*). An object introduced by the functional preposition *to* is mapped to second argument in the case of the predicate *want*, but to the third in the case of *give*, etc. Consider, for illustration, a sample rule from the SSynt-DSynt mapping in Figure 3. This rule, in which we can see the *leftside* and the *rightside* fields, collapses the functional prepositions (*?Xl* identified during the pre-processing stage with the *BLOCK=YES* attribute/value pair) with their dependent (*?Yl*).



```
c:?Xl {                               rc:?Yr {
  BLOCK = YES                           rc:<=> ?Yl
  c:deprel = ?dep                       <=> ?Xl
  c:id = ?i1                            original_deprel = ?dep
  c:?s-> c:?Yl {                      }
    c:id = ?i2
  }
}

(?s == PMOD | ?s == IM | ?s == SUB)
```

Figure 3: A sample graph-transduction rule . *?* indicates a variable; *?Xl*{} is a node, *?r*⟶ is a relation, *a*=*b* is an attribute/value pair.

The SSynt-DSynt mapping inevitably produces duplications of argumental relations, which need to be fixed. The post-processing grammar evaluates the different argument duplications and modifies some edge labels in order to get closer to a correct structure.[6]

For indicative purposes, a former evaluation

[4]For more details about the SSynt-DSynt correspondences, see (Ballesteros et al., 2015).

[5]See (Mille and Wanner, 2015).

[6]58 rules in the post-processing grammar are dedicated to mark coordinations for the representation on the next level; they are duplicates of other rules with other values and are thus not counted in order not to distort the numbers. In general, about 30% of the total number of rules (90/313) are dedicated to simply copy attribute/value pairs on the nodes; these rules are not counted either in the totals shown in Table 3.

performed on 300 manually annotated DSynt structures from Section 23 of the WSJ (6,979 SSynt and 4,976 DSynt tokens) is presented in Table 4.[7] On the EPE data, due to the current state of the rule-based system, the output contains 18 cases of duplicated arguments labels and 89 disconnected structures (out of approximately 40,000 sentences).

Hypernode Identification

Precision	Recall
97.00	99.96

Attachment and labeling

UAP	UAR	LAP	LAR
93.88	96.74	88.54	91.24

Table 4: Reported accuracy scores for our SSynt-DSynt graph transducer; see (Ballesteros et al., 2015).

5 Run 3: Predicate-argument graphs

5.1 Targeted dependency representation

For this run, we target predicate-argument (PredArg) structures with abstract semantic role labels which also capture the underlying argument structure of predicative elements (which is not made explicit in syntax). Lexical units are tagged according to several existing lexico-semantic resources, namely PropBank, NomBank, VerbNet (Schuler, 2005) and FrameNet (Fillmore et al., 2002). The presented system is currently limited to choose the first meaning for each word.[8] During this transition, we also aim at removing support verbs. For the time being, this is restricted to light *be*-constructions, that is, constructions in which the second argument of *be* in the DSyntS is a predicate P that can have a first argument and that does not have a first argument in the structure. In this case, the first argument of the light *be* becomes the first argument of P in the PredArg representation.

$$\begin{tikzcd} X \arrow[r, bend left, "\text{Time}"] & Y & = & X & \text{time} \arrow[l, bend right, "\text{A1}"'] \arrow[r, bend left, "\text{A2}"] & Y \end{tikzcd}$$

Figure 4: Correspondence between a non-core relation and a binary predicate

[7]'UAP'\'UAR' stands for"unlabeled attachment precision\recall"; 'LAP'\'LAR' for "labeled attachment precision\recall".

[8]The selected downstream applications do not use any sense information; only the lemma and PoS features are taken into account.

The predicate-argument relations are sorted in two subtypes: on the one hand, the "core" relations: **Argument1**, **Argument2**, **Argument3**, **Argument4**, **Argument5**, **Argument6**; and, on the other hand, the "non-core" relations: **Benefactive**, **Direction**, **Extent**, **Location**, **Manner**, **Purpose**, **Time**, **NonCore** (which is the only underspecified relation). The non-core labels come mainly from the corresponding labels in the Penn Treebank, that is, they are provided by the surface-syntactic parser. Our system also uses the presence of certain prepositions in order to derive these labels (e.g., *for* often indicates a *purpose*, so non-argumental *for* dependents are simplistically labeled as *purpose* by default). The non-core relations allow for avoiding the introduction of new nodes without a counterpart in the original sentences, which at the same time simplifies the representation. These relations are actually a compact representation of binary predicates, as illustrated in Figure 4. The other available relations are **NAME** (between parts of proper nouns), **Set** (between a coordinating predicate and each of its conjuncts), and **Elaboration** (which connects elements with no argumental relation). PredArg nodes are the same as the DSynt nodes, that is, they are lemmas that can correspond to more than one surface node (hypernodes).[9] The PoS feature set at this level is slightly different from the other two levels in that all morphological information is removed from the tags; that is, all common nouns are tagged *NN* while all verbs are tagged *VB*.

The predicate-argument graphs show some similarities with PropBank structures, with three main differences, namely: (i) PropBank representations capture existing dependencies governed by nominal and verbal elements only; (ii) PropBank representations are forests of trees defined over individual lexemes or phrasal chunks; and (iii) PropBank representations do not differentiate between functional prepositions and meaning-bearing ones.

Predicate-argument structures are also comparable to the target structures of the SemEval 2014 shared task on Broad-Coverage Semantic Dependency Parsing (Oepen et al., 2014). For instance, the DELPH-IN annotation, which is a rough conversion of the Minimal Recursion Semantics treebank (Oepen and Lønning, 2006) into bi-lexical dependencies, also captures the lexical argument

[9]Only in a very limited number of cases nodes can be added in the graph, for example, a coordination conjunction is added in the case of a conjunctionless coordination.

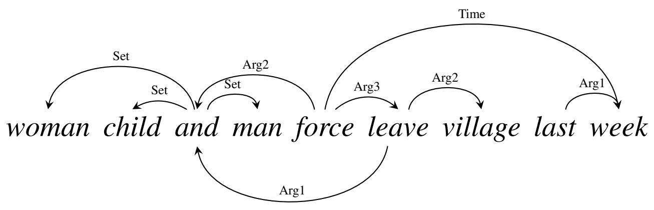


Figure 5: PredArgS for *Women, children and men have been forced to leave the village last week.*

(or valency) structure and ignores some functional elements (such as the *be* copula and governed prepositions) in the graph. DELPH-IN corresponds to the DM run of some EPE participants; see, e.g. (Chen et al., 2017; Schuster et al., 2017). The main differences are in the direction of some edges, the labels used, and the fact that all the words of the original sentence are in the representation, although not always connected with a dependency (in the same fashion as PropBank). Another example is the Enju annotation (Miyao, 2006), which is a pure predicate-argument graph over all 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.[10]

5.2 Implementation

In order to obtain the PredArg structures, we run another sequence of graph-transducers on the output of the DSynt parser (see Section 4.2 for a general description of the grammars); that is, this module takes as input the output provided by Run 2.

The first grammar in this module creates a pure predicate-argument graph, with the mapping of DSynt relations onto PredArg relations according to PropBank/NomBank, and the introduction of new predicates, as *time* on the right part of Figure 4.[11] Coordinating conjunctions are linking elements in the Penn Treebank and DSynt representations; in a predicate-argument graph, they are represented as predicates, which have all the conjuncts as arguments and which receive all incoming edges to the coordinated group; cf. Figure 5. Lexical units are assigned a VerbNet class. Once this is done, a few post-processing grammars are applied; they recover the shared arguments in coordinated constructions, remove light verbs, remove the distinction between external and non-external arguments (i.e., for all predicates that have an *A0*, we push all the arguments one rank up: *A0* becomes *A1*, *A1* becomes *A2*, etc.), assign FrameNet frames and introduce the non-core dependencies – that is, turn the right part of Figure 4 into the left part.

PropBank, NomBank, VerbNet, and FrameNet classes are assigned through a simple dictionary lookup. For this purpose, we built dictionaries that can be consulted by the graph-transduction environment and that contain the classes and their members, together with the mappings between them, using the information from SemLink (Palmer, 2009).

Table 5 summarizes the different steps of this module.[12]

Grammars	#rul.	Description
ALL	154	
DSynt-Sem	59	Assign core dependencies. Recover shared arguments. Establish coord. conj. as predicates. Assign VerbNet classes.
Post-Proc. 1	11	Recover shared arguments in coordinated constructions. Mark light verbs.
Post-Proc. 2	23	Remove light verbs. Assign frames (FrameNet).
Post-Proc. 3	30	Normalize argument numberings.
Post-Proc. 4	31	Introduce non-core dependencies
Speed	≈ 55 ms/sentence	
Memory used	≈ 300MB	

Table 5: Rules for DSynt-PredArg mapping

[10]See (Ivanova et al., 2012) for a more complete overview of Enju and DELPH-IN, and (Oepen et al., 2014) for a parallel illustration of these and tectogrammatical structures.

[11]This kind of representation is useful for some applications such as paraphrasing, but having doubts about their relevance for the EPE tasks, we did not submit a run based on them.

[12]As for the deep-syntactic analysis module, we take out of the count 160 rules that are dedicated to transfer attribute/value pairs only.

Predicate-argument structures are supposed to be connected acyclic graphs, such that each single node can occupy more than one argumental position. Due to the current limitations of the rule-based system, 15 cases of double dependencies between nodes and 150 disconnected structures were produced (out of approximately 40,000 sentences in the EPE data). There is no systematic intrinsic evaluation for this module available as yet.

6 Results and discussion

In order to have an idea of the performance level of the whole pipeline, we carried out an informal evaluation of the whole pipeline on 30 manually annotated sentences from three general domain press articles (about 520 words in total). Due to time restrictions we only evaluated the unlabeled precision and recall, for which the system obtained 74.40 and 71.02 points respectively.

The three selected downstream applications require surface syntactic structures as input: Event Extraction and Negation Scope Detection in the fashion of Stanford (de Marneffe and Manning, 2008), and Opinion Analysis in the CoNLL'08 style (Surdeanu et al., 2008). Thus, it is not surprising that surface-syntactic schemes generally perform better that abstract ones across the different approaches. This is reflected in the extrinsic evaluations of Negation Scope Detection and Opinion Analysis (see Table 6), for which the accuracy of our pipeline seems to drop with the degree of abstraction. However, this is not true for the Event Extraction application, for which the results of SSynt and PredArg are exactly the same, whereas the DSynt exhibits only a light drop.

Run	Event	Negation	Opinion	Avg.
SSynt	46.54	59.78	63.62	56.65
DSynt	45.94	33.34	60.42	46.57
PredArg	46.54	30.67	55.86	44.36

Table 6: Results (F1) obtained for the Event Extraction, Negation Scope Resolution, Opinion Analysis downstream applications, and the average scores for the three representations (starting from raw text)

It is possible that there is a correlation between the scores and the presence of all the nodes of the sentence in the representation. Indeed, the three downstream applications use all the words of the sentence, thus, it is possible that the fact that we remove a lot of words in the DSynt and PredArg structures had a negative impact on the results. This is true for Negation and Opinion, while Event Extraction would be rather insensitive to the change.

Our DSynt and PredArg representations are similar to the DM used by several other participants, but do not seem to trigger the same results: DSynt seems to perform on average slightly better for Event Extraction and Opinion Analysis, but much worse for Negation Scope Resolution. PredArg achieves an even higher score for Event Extraction, but lower scores for the other two applications. Across participants, it seems like maintaining a tree structure helps for Opinion Analysis (PredArg, DM and PSD are all graphs). On the contrary, for Event Extraction, graphs seem to be able to perform as well as trees.

7 Future work

We presented three system outputs to the shared task: (i) a classic syntactic tree, (ii) a deep-syntactic tree with functional words removed and generalized edge labels, and (iii) a predicate-argument graph that shows implicit and explicit argumental relations. These three runs correspond to three different levels of abstraction in the linguistic analysis.

Two interesting conclusions can be drawn from the results: first, an application designed on syntactic trees can work equally well on a semantic graph (Event Extraction); and second, similar types of predicate-argument graphs can lead to very different results. It would be interesting to investigate the impact of missing nodes, of the number of dependencies, and of the type of PoS used in the structure in order to try to explain the different behaviors.

In the future, the current implementation will be improved according to the following aspects: (i) integration of a word sense disambiguation component; (ii) removal of more support verbs in the predicate-argument structures, in particular through the identification of lexical functions (Mel'čuk, 1996). Furthermore, experiments will be carried out on the effect of collapsing of all prepositions (not only the functional ones) in another downstream application, namely, abstractive summarization.

Acknowledgments

We would like to thank warmly the task organizers for their availability and support, and the three anonymous reviewers for their (more than) insightful comments.

References

Miguel Ballesteros, Bernd Bohnet, Simon Mille, and Leo Wanner. 2015. Data-driven deep-syntactic dependency parsing. *Natural Language Engineering* pages 1–36.

Jari Björne, Filip Ginter, and Tapio Salakoski. 2017. EPE 2017: The Biomedical event extraction downstream application. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 13 – 20.

B. Bohnet and J. Nivre. 2012. A transition-based system for joint Part-of-Speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Jeju Island, Korea, pages 1455–1465.

Bernd Bohnet and Leo Wanner. 2010. Open source graph transducer interpreter and grammar development environment. In *Proceedings of the International Conference on Linguistic Resources and Evaluation (LREC)*. Valletta, Malta.

Yufei Chen, Junjie Cao, Weiwei Sun, and Xiaojun Wan. 2017. Peking at EPE 2017: A comparison of tree approximation, transition-based, and maximum subgraph models for semantic dependency analysis. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 56 – 60.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation, 22nd International Conference on Computational Linguistics (COLING)*. Manchester, UK, pages 1–8.

Charles J. Fillmore, Collin F. Baker, and Hiroaki Sato. 2002. The FrameNet database and software tools. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*. Las Palmas, Canary Islands, Spain, pages 1157–1160.

J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL): Shared Task*. Boulder, CO, USA, pages 1–18.

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

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the 6th Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Richard Johansson. 2017. EPE 2017: The Trento–Gothenburg opinion extraction system. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 27 – 35.

Emanuele Lapponi, Stephan Oepen, and Lilja Øvrelid. 2017. EPE 2017: The Sherlock negation resolution downstream application. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 21 – 26.

I.A Mel'čuk. 1996. Lexical functions: A tool for the description of lexical relations in the lexicon. In L. Wanner, editor, *Lexical Functions in Lexicography and Natural Language Processing*, Benjamins Academic Publishers, Amsterdam, pages 37–102.

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

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An interim report. In *Proceedings of the Workshop on Frontiers in Corpus Annotation, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*. Boston, MA, USA, pages 24–31.

Simon Mille, Alicia Burga, and Leo Wanner. 2013. AnCora-UPF: A multi-level annotation of Spanish. In *Proceedings of the 2nd International Conference on Dependency Linguistics (DepLing)*. Prague, Czech Republic, pages 217–226.

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

Yusuke Miyao. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. thesis, The University of Tokyo.

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

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based mrs banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*. Genoa, Italy.

Stephan Oepen, Lilja Øvrelid, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter, and Erik Velldal. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation. Towards a reusable community infrastructure. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 1 – 12.

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

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31:71–105.

Corentin Ribeyre, Djamé Seddah, and Éric Villemonte De La Clergerie. 2012. A Linguistically-motivated 2-stage Tree to Graph Transformation. In Chung-Hye Han and Giorgio Satta, editors, *TAG+11 - The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms - 2012*. INRIA, Paris, France. https://hal.inria.fr/hal-00765422.

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

Sebastian Schuster, Eric De La Clergerie, Marie Candito, Benot Sagot, Christopher D. Manning, and Djam Seddah. 2017. Paris and Stanford at EPE 2017: Downstream evaluation of graph-based dependency representations. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies*. Pisa, Italy, page 43 – 55.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluis Márquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*. Manchester, UK, pages 159–177.