Downstream Evaluation of Graph-based Dependency Representations – The Stanford and Paris Submissions to the EPE 2017 Shared Task

Sebastian Schuster[1,*] **Éric Villemonte de la Clergerie**[3] **Marie Candito**[4]

Benoît Sagot[3] **Christopher D. Manning**[1,2] **Djamé Seddah**[3,5,*]

Department of [1]Linguistics and [2]Computer Science, Stanford University, Stanford, CA
[3]INRIA, Paris, France [4]Université Paris Diderot, Paris, France
[5]Université Paris Sorbonne, Paris, France

{sebschu,manning}@stanford.edu marie.candito@linguist.univ-paris-diderot.fr
{djame.seddah,benoit.sagot,eric.de_la_clergerie}@inria.fr

Abstract

We describe the STANFORD-PARIS and PARIS-STANFORD submissions to the 2017 Extrinsic Parser Evaluation (EPE) Shared Task. The purpose of this shared task was to evaluate dependency graphs on three downstream tasks. Through our submissions, we evaluated the usability of several representations derived from English Universal Dependencies (UD), as well as the Stanford Dependencies (SD), Predicate Argument Structure (PAS), and DM representations. We further compared two parsing strategies: Directly parsing to graph-based dependency representations and a two-stage process of first parsing to surface syntax trees and then applying rule-based augmentations to obtain the final graphs. Overall, our systems performed very well and our submissions ranked first and third. In our analysis, we find that the two-stage parsing process leads to better downstream performance, and that *enhanced UD*, a graph-based representation, consistently outperforms *basic UD*, a strict surface syntax representation, suggesting an advantage of enriched representations for downstream tasks.

1 Introduction

While the main focus of the dependency parsing community still lies on parsing to surface syntax trees, there also has been a growing interest in designing deeper dependency representations that allow for a straightforward extraction of predicate-argument structures, as well as developing methods to parse to these representations.

Notable instances of this line of work are the Prague Dependency Treebank (Böhmová et al., 2003), as well as re-annotated versions of the Penn Treebank (Marcus et al., 1993) with CCGs (Hockenmaier and Steedman, 2007), LFGs (Cahill et al., 2004) or HPSGs (Miyao and Tsujii, 2004).

With the development of the Stanford Dependencies (SD) representation (de Marneffe and Manning, 2008) and its two graph-based flavors, the *collapsed* SD and the *CCprocessed* SD representation, which can all be easily constructed from phrase-structure trees, dependency graph representations started to be used in many downstream systems. More recently, thanks to the two SemEval Shared Tasks on Semantic Dependency Parsing (Oepen et al., 2014, 2015), there has also been a surge in interest in developing parsers that can directly parse to graph-based representations. Further, there have been several initiatives to augment the Universal Dependencies (UD) annotations (Nivre et al., 2016) to recover predicate-argument structures that are missing from strict surface syntax trees. Schuster and Manning (2016) describe an *enhanced* and an *enhanced*++ representation, which both add and augment relations of a surface UD tree. Candito et al. (2017) built upon their work on their own native deep syntax annotation scheme (Candito et al., 2014) and further extended the *enhanced*++ representation by adding dependencies from verbs controlled by nouns and adjectives to their controllers as well as neutralizing several syntactic alternations.

However, while Schuster and Manning (2016) and Candito et al. (2017) discuss automatic methods to obtain these augmented representations from surface syntax UD trees, neither of them evaluated whether these augmentations actually improve the performance of downstream tasks that

*Corresponding Authors.

extract features from dependency graphs.[1]

On top of that, it is still an open research question whether it is better to directly produce these graph-based representations as compared to first parsing to dependency trees and then applying rule-based augmentations to obtain the final graphs. Schluter and Van Genabith (2009) and Çetinoglu et al. (2010) explored direct parsing of LFG f-structures, which usually form a graph. Both report that their rule-based conversions slightly outperformed their direct-parsing approach. More recently, Ribeyre et al. (2016) presented results where a state-of-the-art dependency parser and a rule-based conversion led to similar results as a state-of-the-art graph parser but was vastly outperformed once additional surface syntax-based features were used.

We therefore took the opportunity of this shared task to evaluate the parsability and effectiveness of various Universal Dependencies representations as compared to strict surface syntax trees and other semantic dependency representations. Concretely, we were investigating the following questions.

1. Do enhancements of surface syntax dependency trees that add and augment relations lead to improvements in downstream tasks?

2. Is it more effective to directly produce graph-based representations as compared to first parsing to surface syntax trees and then performing rule-based augmentations to obtain the final graphs?

3. How do the various representations derived from Universal Dependencies compare to other graph-based semantic dependencies representations?

4. Does better parsing performance as measured by intrinsic metrics translate to better performance in downstream tasks?

To answer these questions, we evaluated 8 different annotation schemes (4 of which being extensions of Universal Dependencies trees toward deeper syntactic structures) within two different parsing approaches: (i) direct graph parsing via a neural transition-based graph parser, (ii) a two stage approach of a state-of-the-art surface dependency parser followed by rule-based enrichments. When parsing to the *UD enhanced* representation, our systems ranked first and third in the overall ranking. Our results demonstrate the usefulness of richer graph-based structures and confirm the efficiency of a rule-based enrichment system on top of a state-of-the-art dependency parser.

2 Downstream Tasks

The shared task organizers evaluated the dependency parses on the following three downstream tasks.

Event Extraction The first downstream system is the Turku Event Extraction System (TEES, Björne et al. (2009)), which was the top-performing system in the BioNLP 2009 Shared Task (Kim et al., 2009). For a given sentence and a given set of biological entities, TEES first identifies event triggers, i.e., tokens that describe a certain event. As a second step, TEES then identifies the arguments of each event among all the biological entities in the sentence, as well as the relations between the arguments and the events. Each of the components of TEES are using an SVM classifier with a combination of lexical and dependency path features. This system was originally developed to extract features from the Stanford collapsed dependencies representation (de Marneffe and Manning, 2008). For this shared task, TEES is trained and evaluated on the BioNLP 2009 data set (Kim et al., 2009).

Negation Scope Detection The second downstream system is the negation scope resolution system by Lapponi et al. (2012), which was developed for the 2012 *SEM Shared Task (Morante and Blanco, 2012). This system also consists of two components: a classifier to detect negation cues, i.e., tokens such as *not* or affixes such as *un*, and a sequence labeling model to resolve the negation scope and to identify the event that is being negated. As only the second component uses syntactic features, the present shared task provided gold negation cue predictions and exclusively focused on the second task, resolving the negation scope and identifying the negated events. Lapponi et al. (2012) use a CRF-based sequence labeling model which assigns IOB-style tags to each token, which indicate whether a token is out-of-scope, a cue, in-scope, a negated event, or the end of the scope. The original system was developed to extract features from the Stanford ba-

[1]Note that Michalon et al. (2016) demonstrated the effectiveness of having deep syntactic graphs as input for semantic parsing in the FrameNet framework.

sic dependencies representation (de Marneffe and Manning, 2008). For this shared task, the system was trained and evaluated on the Conan Doyle corpus (Morante and Daelemans, 2012). The evaluation metric considers whether both the scope and the event was correctly output by the system (exact match).

Fine-grained opinion analysis The third downstream system is the fine-grained opinion analysis system by Johansson and Moschitti (2013). For a given sentence, this system first detects different types of subjective and objective expressions and then links them to the opinion holder (if such a holder is explicitly mentioned). The three types of expressions are the ones marked in the MPQA corpus (Wiebe et al., 2005): direct-subjective expressions (DSEs), expressive-subjective elements (ESE), and objective statement expressions (OSE). DSEs are expressions that directly mention emotions and opinions such as *hate* or *approve*; ESEs are expressions that do not explicitly mention an emotion but the choice of words in context conveys and attitude; OSEs are statements and speech events that do not convey an opinion. For the identified DSEs and ESEs, the system further predicts the polarity of the expression, i.e., whether the expression conveys a positive or negative sentiment. To detect DSEs, ESEs, and OSEs, Johansson and Moschitti implement a sequence labeling model which generates multiple labeled candidates. They further use an SVM classifiers that makes use of lexical and syntactic features to identify potential opinion holders, and they use another SVM classifier to predict the polarity of each subjective expression. As the sequence-labeling model can take only local context features into account, they finally rerank the set of candidates using a model that extracts additional features from a dependency tree and semantic role labels. The original system used the dependency representation of the CoNLL 2008 Shared Task (Mel'čuk, 1988; Surdeanu et al., 2008). For the shared task, the organizers removed all SRL features from the reranking system and the opinion holder classifier such that the structural information is exclusively coming from the dependency representation.

3 Experimental Protocol

Our experimental setup is aimed at allowing for all the comparisons that we mentioned in the introduction while at the same time, keeping the total number of runs to a minimum. Our submissions varied along three dimensions: the dependency representation, the parsing method, and the training data composition.

3.1 Representations

In total, we evaluated eight different representations.

DM The Minimal Recursion Semantics-derived dependencies (DM) is a graph-based representation that can be derived automatically from the DeepBank HPSG annotations (Flickinger et al., 2012) using the MRS conversion framework of Oepen and Lønning (2006). Most of the dependency labels indicate the index of the argument, e.g., `ARG1` and `ARG2`, but there also exist some special relations for several semantic phenomena, including coordination and bound variables.

PAS Predicate-argument structure (PAS) is another graph-based dependency representation that has been derived from the automatic HPSG-style annotation the Penn-Treebank (Miyao and Tsujii, 2004). PAS encodes the index of the arguments of each predicates by also using general dependency labels such as `ARG1` and `ARG2` but prefixed with the head's generic part-of-speech (eg. `det`, `verb`, etc.).
Note that the PAS and DM representations are structurally different and are derived through different methods. One being treebank-based and the other grammar-based, their underlying syntactic backbones differ (e.g., in the choice of heads and roots and the treatment of copula).

SD The Stanford Dependencies representation (de Marneffe and Manning, 2008) is a typed dependency representation that encodes the grammatical roles of arguments and modifiers, using approximately 50 different relation labels. We evaluate only the *basic* SD representation, which is always guaranteed to be a strict surface-syntax tree.

UD basic The Universal Dependencies (UD) representation (Nivre et al., 2016) is the result of a global initiative to adapt the Stanford Dependencies representation to a large variety of languages, including morphologically rich ones. The UD relations also encode the grammatical roles of arguments and modifiers. Apart from a slightly different repertoire of relations, UD primarily differs from SD in that it has a stronger tendency to treat content words as heads. The *basic* UD representation is again guaranteed to be a strict surface-syntax tree. We evaluate version 1 of this representation.

UD enhanced The enhanced representation (Schuster and Manning, 2016) is based on the *basic* UD representation and includes additional dependencies for subjects of controlled verbs and for shared arguments or modifiers. Further, this representation uses augmented relation labels to disambiguate the type of coordination or modifier.

UD enhanced++ The enhanced++ representation includes all the additional edges of the *enhanced* representation, and provides special treatment of partitives and multi-word prepositions. We evaluate English *enhanced*++ UD graphs as described by Schuster and Manning (2016) with one exception: We do not add copy nodes for conjoined prepositions or prepositional phrases as some of the downstream systems do not support tokens that are not overtly present in the sentence.

UD diathesis The UD diathesis representation (Candito et al., 2017) builds upon the *enhanced*++ representation. It contains all the relations of the *enhanced*++ representation but additionally, It adds subjects of infinitival verbs controlled not only by verbs but also by nouns or adjectives. Arguments of non-finite verbs (past-participles, gerunds) have also been added. Further, it neutralizes syntactic alternations (e.g., passives) such that demoted agents are turned into subjects and passive subjects into objects.

UD diathesis-- This representation is identical to the UD diathesis representation except that this representation does not contain relation labels augmented with function words (e.g., the relation `nmod:with` is replaced with `nmod`), which drastically reduces the dependency label space.

3.2 Parsers

We evaluated two different parsing scenarios: Directly parsing to dependency graphs and parsing to a surface syntax dependency tree and then applying rule-based conversions to obtain the dependency graphs.

Dependency graph parser We used a version of the transition-based graph parser of Ribeyre et al. (2015) DYALOG-SRNN which was extended in De La Clergerie et al. (2017) with a neural network component implemented in Dynet (Neubig et al., 2017). The key idea is that the neural component can provide the best parser action or, if asked, a ranking of all possible actions. This information is then used as extra features for our parsing model to ultimately make a decision. We kept the same set of transitions as Ribeyre et al. (2015) for the SemEval Semantic Dependency Parsing Shared Task (Oepen et al., 2014), which enables the construction of dependency graphs with multiple governors and orphan nodes (nodes without governors). Essentially, it relies on `pop{0,1}` transitions to discard stack element 0 or 1 from the stack, and an `attach` transition that adds a dependency edge between the two topmost stack elements without removing the dependent element from the stack. We also included features related to the governors of the 3 topmost stack elements and some of their descendants.

The submissions using this parser are all labeled PARIS-STANFORD.

Dependency tree parser and rule-based augmentation Apart from directly parsing to graphs, we also evaluated parsing to surface dependency trees and then applying rule-based conversions to the parser output to obtain the dependency graph representations. Such conversions are only available for the UD representations, and therefore we could not parse to PAS or DM using this method. For parsing to *basic* UD and *basic* SD, we used the dependency parser by Dozat and Manning (2017), which was the best-performing parser in the CoNLL 2017 Shared Task (Zeman et al., 2017). This parser is a graph-based[2] neural dependency parser which represents each token as the output of a multi-layer perceptron on top of a bidirectional LSTM model. It uses these token representations to score each possible dependency relation resulting in a directed weighted graph, and then constructs a maximum spanning tree from these scored edges. Once it constructed the unlabeled dependency tree, it adds labels to the edges with another classifier that again uses the token representations as input.

We convert *basic UD* trees to *enhanced* and

[2]Note that while the parser by Dozat and Manning builds a complete graph during inference, it only uses this graph to find the highest scoring tree. Thus it always outputs a strict surface syntax tree and cannot be used for directly parsing to a graph-based representation such as enhanced UD or PAS.

	DM	PAS	UD basic	UD enhanced	UD enhanced++	UD diathesis	UD diathesis --
				Train Set			
Edges	559975	723445	770873	794299	794572	798185	798185
% empty nodes	21.63	4.30	-	-	-	-	-
				Development Set			
Edges	27779	35573	38054	39236	39246	39418	39418
% empty nodes	21.58	4.25	-	-	-	-	-
Unique labels	52	43	40	279	325	323	40
% additional edges	-27.36	-0.06	-	3.04	3.07	3.54	3.54

Table 1: Data set properties. *Additional edges are calculated relative to UD basic*.

enhanced++ UD graphs with the converter by Schuster and Manning (2016) as implemented in Stanford CoreNLP version 3.8, and we convert *enhanced++ UD* graphs to the two *UD diathesis* representations with a custom converter, modeled after the converter for French by Candito et al. (2017), which uses a graph-rewriting system (Ribeyre et al., 2012).

The submissions using this parser are all labeled STANFORD-PARIS.

3.3 Data

We trained our parsers with two different data sets: the DM SPLIT data set and the FULL data set. The DM SPLIT data contains all the sentences of sections 00-21 of the PTB WSJ, with sections 00-19 being the training data and section 20 the development data, which corresponds to the default split for the PAS and DM treebanks.

The FULL data set consists of sections 02-21 of the PTB WSJ, the first 8 of every 10 sentences of the Brown corpus, and the training split of the GENIA treebank. We used section 22 of the PTB WSJ, the ninth out of every 10 sentences of the Brown corpus, and the development split of the GENIA treebank as a development set. A large portion of this dataset is not annotated with the PAS and DM schemes and therefore, we were only able to train models for the SD and the various UD representations on this data set. While this prevents us from comparing SD and UD to PAS and DM in this setting, it allows us to investigate the effect of adding more in-domain[3] training data.

For DM and PAS, we used the official data sets from the SemEval 2015 SDP Shared Task

[3] The event extraction task requires parsing of biomedical texts similar to the ones that appear in the GENIA treebank, and the negation scope resolution task requires parsing of fiction, which is one the genres of the Brown corpus.

Annotation scheme	**Complete**	**15k**
UD v1 basic	89.70	88.99
UD v1 enhanced	87.44	86.90
UD v1 enhanced++	87.61	87.08
UD v1 enhanced++ diathesis	85.47	84.71
UD v1 enhanced++ diathesis–	86.41	85.68

Table 2: Impact of the training set size (complete training data vs. random sampling of 15k sentences) on the performance of the PARIS-STANFORD parser. All the results are LAS on the FULL development set.

(Oepen et al., 2015). For the SD and UD schemas, we converted phrase-structure trees to dependency graphs using the converter in CoreNLP version 3.8 (for *SD*, *UD basic*, *UD enhanced*, and *UD enhanced++*) as well as a custom converter (for the two *UD* diathesis representations)

We replaced the gold part-of-speech tags in our training data with predicted Universal POS and PTB POS tags.

Given time constraints, for training the PARIS dependency graph parser, we randomly sampled 15k sentences during each of the up to 20 epochs. As the post-hoc results in Table 2 show, this only led to a small loss in parsing performance (0.65 percentage points on average) while reducing our training time by a factor of 10. Because of this sampling, there was less WSJ data in our FULL runs as compared to the DM SPLIT runs but given our good performances in the extrinsic evaluation tasks, we believe that this made the parser less sensitive to domain variation.

3.3.1 Statistics

Table 1 presents some interesting properties of our data sets. In contrast to the UD-based treebanks, the DM and PAS treebanks contain empty nodes. All of the UD-based data sets (except for the *UD diathesis*– one) contain a large set of labels. This does not seem to cause much issues for our parsers, apart from considerably increased parsing and training times. As expected, the augmented UD treebanks contain between 3% and 3.5% more edges than their *UD basic* source.

4 Parsing pipeline

We use a standard parsing pipeline consisting of separate tokenization, sentence splitting, part-of-speech tagging, and parsing steps. We tokenize and sentence-split the shared task data using the English tokenizer in Stanford CoreNLP version 3.8 (Manning et al., 2014) with default parameters. We then jointly predict Universal and PTB POS tags with the tagger by Dozat, Peng and Manning (2017), which we trained on the FULL training data, and then run the parser on the tokenized and tagged input.

In addition, for all our runs and parsers, we used the word2vec word embeddings provided by the CoNLL 2017 Shared Task organizers (Zeman et al., 2017). For the DM SPLIT PARIS-STANFORD run, we also used Brown clusters extended with morphological features, which we extracted from an Americanized version of the British National Corpus following Seddah et al. (2012). For the FULL PARIS-STANFORD run, we extracted the same kind of clusters from the same corpus and the Medline biomedical abstract corpus.

5 Results and Discussion

The results of our submissions on the downstream tasks are shown in Table 3 (DM SPLIT) and Table 4 (FULL). Overall, our submissions performed very well in the shared task: the STANFORD-PARIS submissions ranked first, and the PARIS-STANFORD submissions ranked third according to the official ranking, which considers only the best submission of each team. While part of this success can certainly be attributed to our high-performing parsers and having an expressive relation, we also want to emphasize that to the best of our knowledge, we did use more training data than any of the other teams did, and therefore, the results are not fully comparable.

Effect of augmenting UD trees As shown in Table 4, the overall best-performing run was obtained with the FULL data set parsed to *UD enhanced*, followed by the run that parsed to *UD diathesis*. Further, in all our parser-data combinations, the *UD enhanced* representation led to better downstream results than the *UD basic* representation. All of this suggests that there is value in adding edges to surface syntax trees in order to make the relations between content words more explicit. This is also in line with the results by Silveira (2016) who reports that the *UD enhanced* and *UD enhanced*++ representations led to better performance than *UD basic* in a biomedical event extraction task. However, based on the present results, it is hard to make more specific claims about the four graph-based UD representations that we evaluated as we observe some unexpected inconsistencies, which require further investigation. For example, as described above, the *UD diathesis* representation extends the *UD enhanced*++ representation and in the STANFORD-PARIS FULL runs, we obtained better results with *UD diathesis* than with *UD enhanced*++. However, in some of the other runs, we observed the opposite and it is unclear at this point why the effect of these enrichments varies so much.

Parsing methods Our results consistently indicate that it is more efficient to first parse to a dependency tree and then augmenting the output using rule-based converters to obtain dependency graphs as compared to directly parsing to graphs. The results in Table 3 and Table 4 show that the STANFORD-PARIS systems, which produced dependency graphs using the combination of a surface parser and rule-based converters, consistently outperformed the PARIS-STANFORD systems, which directly parsed to a graph. Interestingly, Ribeyre et al. (2016) found the opposite to be true when they evaluated their parser on the Deep French Treebank (Candito et al., 2014). However, as they note, it might be possible to improve the two-stage parser by improving the conversion scripts such that they can better deal with parsing errors. The converter in CoreNLP contains some of these heuristics, which – in combination with the superior performance of the parser by Dozat and Manning (2017) – might explain why our two-stage parsing approach works better.

Comparison to other representations Compared to the UD family, the DM and PAS representations abstract further away from the surface syntax and arguably provide a better way to extract the arguments of a predicate independent of their

Team	Run	Representation	Event Extraction	Negation Scope	Opinion Analysis	Average
PARIS-STANFORD	0	DM	46.04	59.78	57.37	54.40
PARIS-STANFORD	1	PAS	45.99	58.29	58.54	54.27
PARIS-STANFORD	2	UD v1 basic	49.55	55.98	63.50	56.34
PARIS-STANFORD	3	UD v1 enhanced	48.29	56.75	64.27	56.44
PARIS-STANFORD	4	UD v1 enhanced++	47.17	55.59	63.38	55.38
PARIS-STANFORD	5	UD v1 enhanced++ diathesis	48.72	55.59	64.24	56.18
PARIS-STANFORD	6	UD v1 enhanced++ diathesis–	46.81	56.75	62.18	55.25
STANFORD-PARIS	1	UD v1 basic	47.73	63.05	64.24	58.34
STANFORD-PARIS	2	UD v1 enhanced	47.76	63.75	64.18	58.57
STANFORD-PARIS	3	UD v1 enhanced++	48.76	**64.10**	**64.37**	**59.08**
STANFORD-PARIS	4	UD v1 enhanced++ diathesis	47.99	63.05	64.23	58.42
STANFORD-PARIS	9	UD v1 enhanced++ diathesis--	48.51	61.62	63.94	58.02

Table 3: Results on the downstream tasks (F1) for our systems trained on the DM SPLIT data set. Numbers in **bold** indicate the best overall result; numbers in **blue** indicate the best results of the PARIS-STANFORD parser.

Team	Run	Representation	Event Extraction	Negation Scope	Opinion Analysis	Average
PARIS-STANFORD	7	UD v1 basic	49.14	56.36	63.28	56.26
PARIS-STANFORD	8	**UD v1 enhanced (#3)**	49.31	**57.14**	**63.97**	**56.81**
PARIS-STANFORD	9	UD v1 enhanced++	**49.41**	55.98	63.78	56.39
PARIS-STANFORD	10	UD v1 enhanced++ diathesis	48.10	53.18	63.29	54.86
PARIS-STANFORD	11	UD v1 enhanced++ diathesis–	47.70	56.75	62.65	55.70
STANFORD-PARIS	0	Stanford basic	**50.29**	65.13	63.72	59.71
STANFORD-PARIS	5	UD v1 basic	49.13	64.80	64.70	59.54
STANFORD-PARIS	6	**UD v1 enhanced (#1)**	50.23	**66.16**	65.14	**60.51**
STANFORD-PARIS	7	UD v1 enhanced++	49.85	63.75	**65.20**	59.60
STANFORD-PARIS	8	UD v1 enhanced++ diathesis	50.14	64.45	65.01	59.87
STANFORD-PARIS	10	UD v1 enhanced++ diathesis--	48.99	65.13	64.55	59.56

Table 4: Results on the downstream tasks (F1) for our systems trained on the FULL (WSJ, Genia, and Brown) data set. Numbers in **bold** indicate the best overall result; numbers in **blue** indicate the best results of the PARIS-STANFORD parser.

syntactic realization. However, at the same time, UD, and especially the enhanced versions of UD, provide a more fine grained label set than DM and PAS do. Our results suggest that the latter is more important for at least two of the three downstream tasks in this shared task. If we only consider the PARIS-STANFORD runs that were trained on the DM SPLIT data set, which all used the same parser trained on the same sentences, then we observe that all the UD-based representations performed better than the DM and PAS representations on the event extraction and opinion mining tasks. Interestingly, for the negation scope resolution task, DM and PAS performed better than UD if one only considers the PARIS-STANFORD runs. However, it is also noteworthy that for some reason, the PARIS-STANFORD UD runs led to much lower downstream performance than the STANFORD-PARIS UD runs on this task, which was not the case for the other two tasks. Overall, it seems to be the case that under equal conditions, there is no representation that works best for all three downstream tasks, but if one considers the average performance, then the UD-based representations – and in particular the graph-based ones – seem to give better results, especially for semantic downstream tasks.

One potential confound is that two of the three systems (event extraction and negation scope resolution) were originally developed to extract features from SD, which is very similar to UD, and that the feature extraction therefore might be tailored towards an SD-like representation. While this is a valid criticism of the setup of this shared task, the fact that DM and PAS scored higher than UD in the negation scope resolution task indicates that at least one of the downstream systems seems to be able to effectively make use of features from various dependency representations.

Comparison of UD to SD Achieving cross-linguistic consistency was one of the primary goals in developing the UD representation (Nivre et al., 2016). On the other hand, one of the main design criteria in developing the SD representation was its usability in downstream tasks (de Marneffe and Manning, 2008). Considering these two potentially competing goals, we wanted to compare the basic UD and basic SD representations. Table 4 shows that on average, there is very little difference in downstream performance with SD performing slightly better than UD (59.71 vs. 59.54). In our experiments, UD performed worse on event extraction and slightly worse on negation resolution but better on the opinion analysis task. However, overall, the runs with both representations performed very well and these results suggest that despite the different primary goal of UD, UD is as useful or at least almost as useful as SD in downstream tasks.

Intrinsic evaluation The upper part of Table 5 shows the intrinsic performance on the FULL development data as well as the average downstream performance of the PARIS-STANFORD and STANFORD-PARIS parsers. Both of these parsers were trained on the FULL test data, so these numbers are comparable. While two data points are clearly not sufficient to conclude that there is a meaningful correlation, these results provide anecdotal evidence that better parsing performance also translates to better downstream performance.

The lower part of Table 5 shows the results of the PARIS-STANFORD parser on the DM development data, which we included to allow for comparisons with shared task submissions from other teams that also used this representation and data set.

Domain sensitivity and training data size When we train on more out-of-domain data using the FULL data set, the STANFORD-PARIS parser shows a very systematic improvement of 1-1.5 points as compared to training on the DM SPLIT. The PARIS-STANFORD results, on the other hand, are more stable. Part of the reason for this stability potentially comes from the random sampling training process as well as the use of clusters that were extracted from the combination of a very balanced corpus and a very specific one such as the BNC (170M tokens) and the Medline corpus (22M).

As mentioned before, to the best of our knowledge, the FULL data set is larger and more diverse than the data used by other participants in this shared task. However, this is not true for the DM SPLIT, and it is noteworthy that even if we only consider the systems trained on the DM SPLIT, we would have still ranked first with our *UD enhanced++* run.

UD basic (FULL)	LAS	UAS	F1
PARIS-STANFORD	88.99	90.43	56.26
STANFORD-PARIS	91.13	93.26	59.54
DM	LF	UF	F1
PARIS-STANFORD	85.25	86.95	54.40

Table 5: Intrinsic and extrinsic performance of three of our runs. The top part shows the parsing performance (LAS and UAS) of the two parsers on the FULL development set as well as the downstream performance (F1) of these two systems. The lower part shows the parsing performance (LF and UF) of the PARIS parser on the DM development set and its downstream performance (F1).

A note on our runs with the DM annotation scheme If we compare our parser trained on the DM treebank to the parsers of other participants who trained their parsers on the same annotation scheme, we observe that they all exhibit the same levels of performance on average but differ considerably on each task (see Table 6). Our parser performed better on the Event Detection task than the parsers by the Peking and UW teams, but it performed considerably worse on the opinion mining task. One explanation might be that we used the 2014 data set whereas the other two teams presumably used the 2015 data sets but further investigations are needed on this point.

Team	Event	Neg.	Op.	Av.
PARIS-STANFORD	**46.04**	59.78	57.37	54.40
PEKING	43.39	**60.89**	59.03	54.44
UW	42.84	56.75	**60.18**	53.26

Table 6: Results on the downstream tasks (F1) of parsers trained on the DM scheme.

6 Conclusion

As our discussion hopefully conveyed, it is very challenging to exactly pinpoint the factors that contribute to a representation being useful for downstream tasks, and the results are not conclusive enough to make a specific recommendation which representation should be used when building a downstream system. However, we found that the *enhanced UD* graph representation consistently outperformed the *basic UD* surface syntax representation, which suggests that the additional edges and augmented labels provide an advantage in downstream tasks. The results for the other graph-based UD representations were less consistent and more work is needed to determine whether they help in downstream tasks.

That all being said, based on the results of this shared task, we do believe that the various representations derived from UD are well suited for downstream tasks as they constitute expressive representations for which sophisticated data conversion tools and high-performing parsers exist.

References

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting contextualized complex biological events with rich graph-based feature sets. In *Proceedings of the Workshop on BioNLP: Shared Task*. pages 10–18.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The prague dependency treebank. In *Treebanks*, Springer, pages 103–127.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Cahill, Aoife and Burke, Michael and O'Donovan, Ruth and van Genabith, Josef and Way, Andy (2004) Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In: ACL 2004 - 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July 2004, Barcelona, Spain.*. Barcelona, Spain.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. Deep syntax annotation of the Sequoia French treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*. pages 2298–2305.

Marie Candito, Guy Perrier, Bruno Guillaume, and Djamé Seddah. 2017. Enhanced UD dependencies with neutralized diathesis alternation. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*.

Özlem Çetinoglu, Jennifer Foster, Joakim Nivre, Deirdre Hogan, Aoife Cahill, and Josef van Genabith. 2010. Lfg without c-structures. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*.

Eric De La Clergerie, Benoît Sagot, and Djamé Seddah. 2017. The ParisNLP entry at the CoNLL UD shared task 2017: A tale of a #parsingtragedy. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 243–252.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. pages 1–8.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 20–30.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Richard Johansson and Alessandro Moschitti. 2013. Relational features in fine-grained opinion analysis. *Computational Linguistics* 39(3):473–509.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp 2009 shared task on event extraction. In *Proceedings of the Workshop on BioNLP: Shared Task*. pages 1–9.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. Uio$_2$: Sequence-labeling negation using dependency features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*. pages 319–327.

Christopher D Manning, John Bauer, Jenny Finkel, Steven J Bethard, Mihai Surdeanu, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 55–60.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

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

Olivier Michalon, Corentin Ribeyre, Marie Candito, and Alexis Nasr. 2016. Deeper syntax for better semantic parsing. In *Coling 2016*.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of the 18th International Conference on Computational Linguistics*. Geneva, Switzerland, pages 1392–1397.

R Morante and E Blanco. 2012. *SEM 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*.

Roser Morante and Walter Daelemans. 2012. ConanDoyle-neg: Annotation of negation in Conan Doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*. pages 1563–1568.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 915–926.

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 *Proc. of the 8th International Workshop on Semantic Evaluation*. pages 63–72.

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

Corentin Ribeyre, Djamé Seddah, and Éric Villemonte De La Clergerie. 2012. A linguistically-motivated 2-stage tree to graph transformation. In *TAG+ 11-The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms-2012*.

Corentin Ribeyre, Éric Villemonte De La Clergerie, and Djamé Seddah. 2015. Because syntax does matter: Improving predicate-argument structures parsing using syntactic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Corentin Ribeyre, Éric Villemonte de La Clergerie, and Djamé Seddah. 2016. Accurate deep syntactic parsing of graphs: The case of French. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 3563–3568.

Natalie Schluter and Josef Van Genabith. 2009. Dependency parsing resources for French: Converting acquired lexical functional grammar f-structure annotations and parsing f-structures directly. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA 2009)*.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 2371–2378.

Djamé Seddah, Benoît Sagot, and Marie Candito. 2012. The Alpage architecture at the SANCL 2012 shared task: Robust preprocessing and lexical bridging for user-generated content parsing. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*. Montréal, Canada.

Natalia Silveira. 2016. *Designing Syntactic Representations for NLP: An Empirical Investigation*. Ph.D. thesis, Stanford University.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 08)*. pages 159–177.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation* 39(2/3):165–210.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökrmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Hěctor Martínez Alonso, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadova, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 1–19.