The 2017 Shared Task on Extrinsic Parser Evaluation Towards a Reusable Community Infrastructure

Stephan Oepen[♣♠], **Lilja Øvrelid**[♣♠], **Jari Björne**[♡], **Richard Johansson**[♢], **Emanuele Lapponi**[♣], **Filip Ginter**[♡], **and Erik Velldal**[♣]

[♣] University of Oslo, Department of Informatics

[♠] Center for Advanced Study at the Norwegian Academy of Science and Letters

[♡] University of Turku, Department of Information Technology

[♢] Chalmers Technical University and University of Gothenburg, Department of Computer Science and Engineering

epe-organizers@nlpl.eu

Abstract

The 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017) seeks to provide better estimates of the relative utility of different types of dependency representations for a variety of downstream applications that depend centrally on the analysis of grammatical structure. EPE 2017 defines a generalized notion of lexicalized syntactico-semantic dependency representations and provides a common interchange format to three state-of-the-art downstream applications, viz. biomedical event extraction, negation resolution, and fine-grained opinion analysis. As a first step towards building a generic and extensible infrastructure for extrinsic parser evaluation, the downstream applications have been generalized to support a broad range of diverese dependency representations (including divergent sentence and token boundaries) and to allow fully automated re-training and evaluation for a specific collection of parser outputs. Nine teams participated in EPE 2017, submitting 49 distinct runs that encompass many different families of dependency representations, distinct approaches to preprocessing and parsing, and various types and volumes of training data.

1 Introduction & Motivation

Natural language parsing, computing syntactico-semantic structure according to the rules of grammar, is widely considered a prerequisite technique to most forms of language 'understanding'. These very broadly comprise applications of natural language processing (NLP) that require an analysis of (among other things) 'who did what to whom'—as for example diverse types of information extraction or relation and event detection tasks.

Computational parsing of natural language has made great advances over time. For example, the crisp benchmark of replicating parts of the English phrase structure annotations in the venerable Penn Treebank (PTB; Marcus et al., 1993) allows a quantitative comparison spanning more than two decades: Magerman (1995), one of the early parsing accuracy reports against the PTB using the ParsEval measure of Black et al. (1991) recorded a score for constituent labeling precision and recall of 84.2 F_1 points.[1] At 91.0 F_1, the parser of Charniak and Johnson (2005) appeared to mark a PTB parsing plateau for some time, but neural advances in recent years have led to ParsEval F_1 levels of 93.8 (Andor et al., 2016).

Quantitative measures of parsing success in terms of (degrees of) similarity with a gold-standard target representation constitute *intrinsic* parser evaluation and have been a central driving force in much research and engineering on syntactico-semantic parsing. Intrinsic measures, however, cannot predict the contribution of a token parser to a specific NLP application, say relation detection over syntactic analyses. Therefore, quantitative intrinsic benchmarking or historic reflections on parsing progress like the above do not immediately inform us about corresponding advances in natural language 'understanding' capabilities—arguably the ultimate motivation for long-term research interest in natural language parsing.

Another parameter that inhibits comparability

[1] This report is against Section 00 of the PTB, whereas much subsequent work standardized on Section 23 for benchmarking. More importantly, however, Magerman (1995) excluded from the evaluation test sentences above forty words in length, a simplification that has been dropped in more recent PTB parsing research. Under the plausible assumption that longer sentences are, if anything, not easier for the parser to analyze correctly, the score reported by Magerman (1995) probably overestimates the performance level of the time, when compared to current PTB reports.

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 1–16,
Pisa, Italy, September 20, 2017. ©2017 Association for Computational Linguistics

and broader judgment of scientific progress is variability in the target representations for natural language parsing. Dependency-based syntactico-semantic representations have received much attention in parsing research of at least the past decade, in part because they offer a comparatively easy-to-use interface to grammatical structure. Over an even longer period, the formal and linguistic foundations of syntactico-semantic dependency analysis have continously evolved, and there is considerable variation across representations schemes in use today—even within a single language.

For English, for example, variations of the so-called LTH scheme (named after the Faculty of Engineering at Lund University) defined by Johansson and Nugues (2007) were used for the 2007, 2008, and 2009 shared tasks of the Conference on Natural Language Learning (CoNLL). Subsequently, the family of Stanford Dependencies (SD) proposed by de Marneffe and Manning (2008) has enjoyed wide popularity. And more recently, the Universal Dependencies (UD; McDonald et al., 2013; de Marneffe et al., 2014; Nivre et al., 2016) and Semantic Dependency Parsing (SDP; Oepen et al., 2014, 2016) representations further increase diversity—as target representations for the 2017 CoNLL shared task and for parsing tasks at the 2014 and 2015 Semantic Evaluation Exercises (SemEval), respectively.

For each of these representations (and others), detailed intrinsic evaluation reports are available that allow one to estimate parser performance (for example in terms of average dependency accuracy and speed) for different types of input text. These reports, however, are difficult to compare across types of representations (and sometimes different selections of test data), and they fail to provide insights into the actual utility of the various representations for downstream tasks that use grammatical analysis as a preprocessing step.

The purpose of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017) was to shed more light on the *downstream utility* of various representations (at the available levels of accuracy for different parsers), i.e. to seek to contrastively isolate the relative contributions of each type of representation (and corresponding parsing systems) to a selection of state-of-the-art downstream applications—which use different types of text, i.e. exhibit broad domain and genre variation.

2 Syntactico-Semantic Dependencies

Figure 1 shows a representative sample of different dependency representations for the sentence:[2]

> *A similar technique is almost impossible to apply to other crops.*

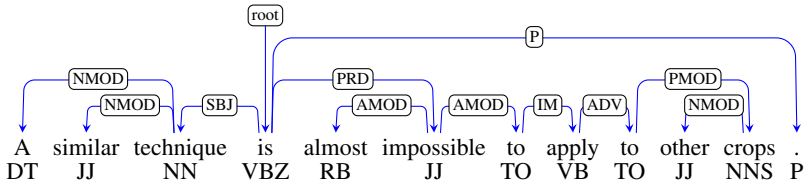
Two 'classic' syntactic dependency trees are presented in 1a and 1b, viz. the target representations from the 2008 CoNLL shared task (Surdeanu et al., 2008) and so-called basic Stanford Dependencies (de Marneffe et al., 2006), respectively. Both are obtained by conversion from the phrase structure annotations in the PTB, combining heuristic head finding rules in the tradition of Collins (1999) with either an interpretation of PTB functional annotations, in the CoNLL case,[3] or with rules targeting specific constructions (e.g. passives or attributive adjectives) in the case of the Stanford Dependencies. While related in spirit, the two analyses differ widely in both their choices of heads vs. arguments and the inventory of dependency types. Where CoNLL tends to view functional words as heads (e.g. the predicative copula or infinitival particle *to*), the Stanford scheme capitalizes more on substantive heads (e.g. the predicative adjective or main verb *apply*).

The Universal Dependencies (UD) in Figure 1c (Nivre et al., 2016) derive from the Stanford Dependencies but generalize beyond the study of English and integrate several parallel initiatives for cross-linguistically valid morphological (Zeman, 2008; Petrov et al., 2012) and syntactic dependency annotation (McDonald et al., 2013; Rosa et al., 2014). UD takes the tendency to select substantive heads one step further, analyzing the prepositional complement *crops* as a head, with the preposition itself as a dependent case marker.[4] This representation was employed in the CoNLL 2017 shared task (Ze-

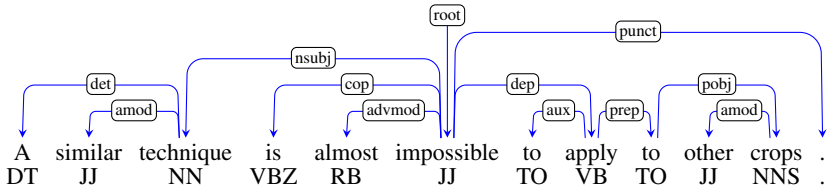
[2] This example is a simplification of a sentence from Section 02 of the PTB and has first been discussed in detail for a broad range of dependency representations by Ivanova et al. (2012) and Oepen et al. (2016).

[3] Johansson and Nugues (2007) discuss the specifics of the conversion for CoNLL 2008, which was implemented as one of several variants in the so-called LTH PennConverter software.

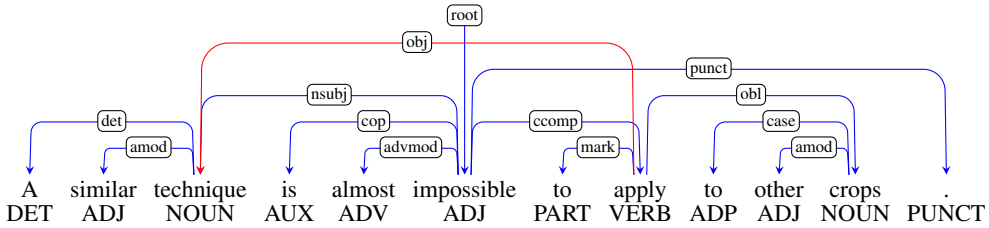
[4] Through its so-called 'enhanced' dependency layer, UD relaxes the constraint that syntactic dependency representations be trees (in the formal sense of connecting each node to the root via a unique directed path): In 1c, for example, *technique* is both a subject dependent of *impossible* and an object dependent of *apply*, marking a reentrancy into this graph node. To date, however, hardly any UD treebanks or parsers support such enhanced dependencies.



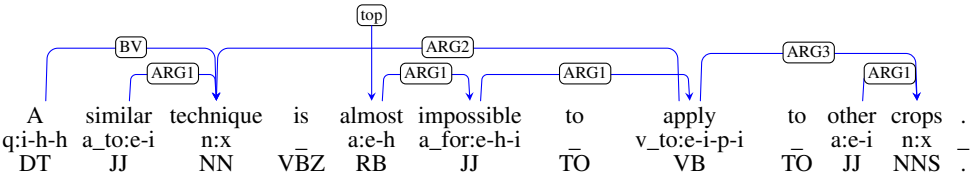
(a) CoNLL: 2008 variant of LTH Dependencies



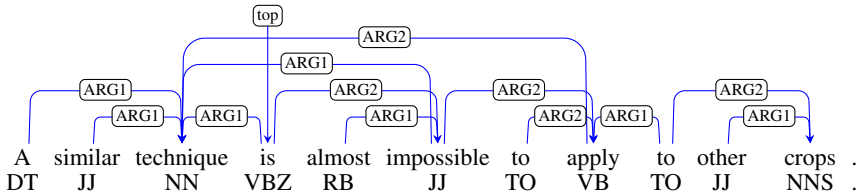
(b) SB: Stanford Basic Dependencies



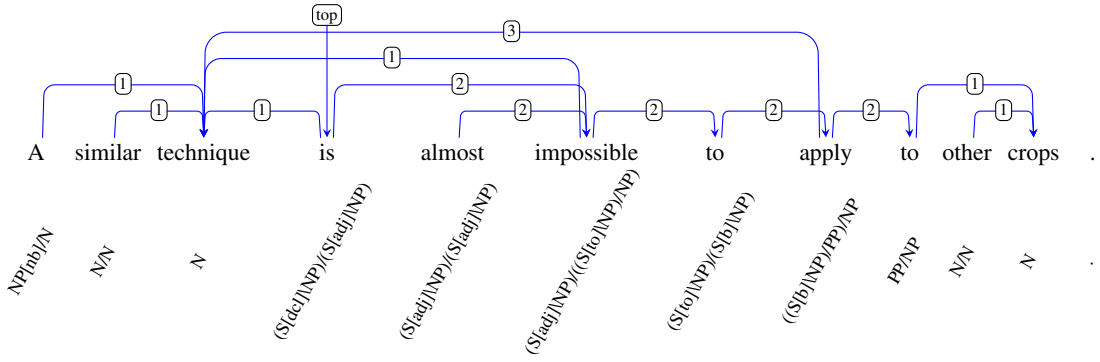
(c) UD: Universal Dependencies (enhanced in red)



(d) DM: DELPH-IN MRS Bi-Lexical Dependencies



(e) PAS: Enju Predicate–Argument Structures



(f) CCD: CCGbank Word–Word Dependencies

Figure 1: Selection of syntactico-semantic dependency representations at EPE 2017.

man et al., 2017), which was devoted to UD parsing from raw text for more than 40 different languages.

Whereas the three first representations are syntactic in nature, there has been some interest in recent years in so-called semantic dependency representations, which necessarily take the form of unrestricted directed graphs. Figures 1d and 1e, for example, show DELPH-IN MRS Bi-Lexical Dependencies (DM; Ivanova et al., 2012) and Enju Predicate–Argument Structures (PAS; Miyao, 2006), respectively. Both are semantic in the sense of using dependency labels that correspond to 'deep' argument positions of the predicates (rather than to surface grammatical functions) and in treating (most) modifiers and determiners as heads—leading to high degrees of graph reentracies, for example at the *technique* node. DM and PAS were among the target representations in a series of parsing tasks at the 2014 and 2015 SemEval conferences. Finally, the CCD word–word dependencies in 1f, which are derived from CCGBank (Hockenmaier and Steedman, 2007; Oepen et al., 2016), arguably range somewhere inbetween the primarily syntactic (CoNLL, SB, and UD) and the more semantic dependency graphs (DM and PAS), as their dependency labels identify argument positions in the CCG lexical categories. Among the selection of dependency representations in Figure 1, DM stands out in (a) leaving semantically vacuous nodes (the copula, infinitival particle, and case-marking preposition) unconnected and (b) its correspondingly lower edge density. Kuhlmann and Oepen (2016) quantitatively contrast these and other dependency representations in terms of a range of formal graph properties.

3 Methodological Challenges

For extrinsic evaluation to provide useful feedback to parsing research, it is necessary to try and tease apart the various contributions to observable end-to-end results. When evaluated as a component of a complex downstream application, the parser proper is but one of many factors that determine extrinsic evaluation performance. Thus, EPE 2017 seeks to capitalize on an initial selection of downstream applications that are understood to *depend centrally on grammatical structure*—in that they all seek to recognize complex and interacting relations where the component pieces often are syntactico-semantic constituents whose interactions are mediated by grammar.

Second, extrinsic evaluation will be most informative when performed at or near *state-of-the-art performance levels*, i.e. reflecting the current best choice of downstream approaches. The 'state of the art' is, of course, both a moving target and inherently correlated with progress in parser engineering. However, at implausibly lower performance levels it could be hard to ascertain, for example, whether failure of a substantive change in the parser to cause an observable end-to-end effect renders the revision irrelevant (to this specific downstream application), or whether the application merely lacks sensitivity as a measurement tool. The EPE 2017 downstream applications all perform at current state-of-the-art levels,[5] and end-to-end results observed in § 8 below compare favorably to published prior art.

Third, the ultimate focus of the EPE 2017 initiative is to enable *comparison across different syntactico-semantic dependency representations*. In principle, all our downstream systems are based on machine learning and are automatically retrained for each different submission of parser outputs—i.e. customized to the specifics of each distinct representation and parser. In preparing the systems for use in the EPE 2017 context, feature templates or heuristics that were specialized to one specific scheme of syntactico-semantic analysis (e.g. targeting individual PoS tags or dependency types) have been generalized or removed.[6] However, as all three systems were originally developed against one specific type of dependency representation, there is remaining room for hidden 'bias' there: Original feature design and selection (based on experimental results) have quite possibly been influenced by the linguistic properties of the specific variant of dependency representation targeted at the time. Short of manual tuning, error analysis, and optimization for at least each distinct type of syntactico-semantic dependency representation

[5] For the Sherlock negation resolution system of Lapponi et al. (2012, 2017), the subsequent studies by Packard et al. (2014) and Fancellu et al. (2016) suggest slight performance advances. However, these systems would not be immediately suitable for extrinsic parser evaluation across frameworks because they are highly specialized to one type of semantic representation, on the one hand, or very constrained in their utilization of syntactico-semantic analyses, on the other hand. Moreover, the top-performing submissions to EPE 2017 appear to again advance the state of the art moderately.

[6] For two of the downstream systems, we have confirmed that 'pseudonymizaton' of parser outputs by systematic renaming of tags and edge labels has no or only neglible effects on end-to-end results. In future use of the EPE infrastructure, we plan to make pseudonymizaton an automated part of the extrinsic evaluation pipeline.

submitted to the task, there is no practical way of fully eliminating such bias. In generalizing their sytems for EPE 2017, the application developers have sought to reduce such affinity to individual dependency schemes, but end-to-end results (see § 8) suggest that more work will be required. By making all three systems (and all submitted parser outputs, together with end-to-end results) publicly available, we hope that parser developers will be enabled to apply more in-depth error analysis and, ideally, also to adapt and extend the downstream systems accordingly.

Finally, among the participating parsers there are *multiple dimensions of variation* at play, besides differences in their choices of syntactico-semantic output representation. One such dimension is the parser itself, i.e. whether it internally targets tree- or graph-shaped target representations; whether it parses directly into bi-lexical dependencies or into, say, constituent trees that are converted to dependencies; whether it employs 'classic' machine learning or neural techniques; whether there is a layer of (typically heuristic) post-processing and 'enhancement' of dependencies after parsing proper; and of course its overall 'maturity' and level of output accuracy. Submissions to the task also differ widely in the amount of training data used in constructing the parser, ranging from a few hundred thousand to almost two million tokens of annotated text. Last but not least, the EPE 2017 task opts to break with a long tradition of CoNLL and SemEval parsing competitions that start from preprocessed inputs—by assuming parser inputs where segmentation into sentences and tokens (and sometimes also PoS tagging and lemmatization) have been applied beforehand. In contrast, for a more 'realistic' interpretation of the parsing problem, the EPE 2017 task starts from original documents of 'raw' running text, such that participating systems will also differ in how they prepare these inputs prior to parsing.[7]

4 Related Work

Even though the bulk of work on parser evaluation focuses on intrinsic output quality metrics, there have been a few previous studies devoted to extrinsic parser evaluation. Several studies compare different parsers using the same syntactic representation in downstream tasks such as machine translation (Popel et al., 2011) and sentiment analysis (Gómez-Rodríguez et al., 2017), but in the following we will focus on studies devoted to the comparison of different types of syntactico-semantic representations in downstream evaluation.

Miyao et al. (2008) compare the performance of constituent-based, dependency-based, and deep linguistic parsers on the task of identifying protein–protein interactions (PPI) in biomedical text. The dependency parsers assign CoNLL-style analyses and are compared to PTB-style constituent parsers and to the HPSG-based Enju parser, where the authors find comparable results for all three repsentations while emphasizing the importance of domain adaptation for all parsers.

Johansson and Nugues (2008) also contrast constituent-based PTB and dependency-based CoNLLrepresentations in the downstream task of semantic role labeling. They find that the dependency-based systems perform slightly better in the sub-problem of argument classification, whereas the constituent-based parsers achieve slightly higher results in argument identification.

Buyko and Hahn (2010) compare the 2007 and 2008 CoNLL schemes and Stanford Basic Dependencies for the task of event extraction from biomedical text. They find that the more functionally oriented CoNLL representations largely outperform the content-oriented Stanford scheme for this task.

In a SemEval 2010 shared task on Parser Evaluation Using Textual Entailments (Yuret et al., 2010), widely different syntactic frameworks—PTB constituent trees, CCG analyses, and dependency representations—are compared in the downstream task of textual entailment recognition. A small dataset was constructed containing entailments that rely on syntactic information (such as active vs. passive sentences). The participants were then required to create their own entailment recognition system, a step which the parser developers solved with varying degrees of success, where the two top-performing systems for this task both employed a CCG parser.

The previous work that perhaps is most similar to EPE 2017 is that of Elming et al. (2013), where the focus is on comparison of different types of dependency representations and their contributions over several different downstream tasks: nega-

[7] To enable participation by teams who might not have a pipeline for English sentence splitting and tokenization readily available, the task co-organizers also provided a preprocessed secondary variant of all parser inputs, which about a third of all submissions used as a matter of convenience.

tion resolution, semantic role labeling, statistical machine translation, sentence compression, and perspective classification. They contrast the performance of the same parser trained on various dependency conversions of the Penn Treebank: the Yamada–Matsumoto scheme, the CoNLL 2007 and 2008 target representations,[8] and the annotation scheme used in the English Web Treebank (an extension of basic Stanford Dependencies). Elming et al. (2013) find that the choice of dependency representation has clear effects on the downstream results and furthermore that these effects vary depending on the task. For negation resolution for instance, the Yamada–Matsumoto scheme performs best, whereas the Stanford and LTH schemes lead to superior SRL performance.

5 Shared Task Set-Up

The EPE 2017 task was sponsored jointly by the Fourth International Conference on Dependency Linguistics (DepLing) and the 15th International Conference on Parsing Technologies (IWPT). Parser inputs were released in mid-March 2017, system submissions due in mid-June, and results presented on September 20, 2017, as part of the overlapping programme for DepLing and IWPT. Further details on the task schedule, technical infrastructure, and results are available from the task web site at:

`http://epe.nlpl.eu`

The following sections briefly discuss aspects of the task set-up that are of broader methodological interest.

Dependency Representations The term (bi-lexical) dependency representation in the context of EPE 2017 is interpreted as a graph whose nodes are anchored in surface lexical units, and whose edges represent labeled directed relations between two nodes. Each node corresponds to a sub-string of the underlying linguistic signal (input string), identified by character stand-off pointers. Node labels can comprise a non-recursive attribute–value matrix (or 'feature structure'), for example to encode lemma and part of speech information. Each graph can optionally designate one or more 'top' nodes, broadly interpreted as the root-level head or highest-scoping predicate (Kuhlmann and Oepen, 2016). This generalized notion of dependency graphs encompasses both 'classic' syntactic dependency trees as well as structures that relax one or more of the 'treeness' assumptions made in much syntactic dependency parsing work, as is the case, for example, in various types of semantic dependency graphs (see § 2 above).

Defining nodes in terms of (in principle arbitrary) sub-strings of the surface signal makes the EPE 2017 view on dependency representations independent of notions of 'token' or 'word' (which can receive divergent interpretations in different types of dependency representations). Furthermore, the above definition does not exclude overlapping or 'empty' (i.e. zero-span) node sub-strings, as might characterize more weakly lexicalized dependency graphs like Elementary Dependency Structures (EDS; Oepen and Lønning, 2006) or even Abstract Meaning Representation (AMR; Banarescu et al., 2013), if aligned to surface sub-strings. However, current EPE 2017 downstream systems only have limited (if any) support for overlapping or empty dependency nodes and, hence, may not immediately be able to take full advantage of these more weakly lexicalized types of semantic (dependency) graphs.

EPE 2017 is (regrettably) limited to parsing English text. For each downstream application, separate training, development, and evaluation data has been provided as 'running' clean text (i.e. without information about sentence and token boundaries). There are no limitations on which parsing approaches and resources can be put to use, as long as the output of the parsing system is a dependency representation in the above sense (and the parser is wholly independent of the evaluation data).

Interchange Format To generalize over a broad variety of different dependency representations and to provide a uniform interface to the various downstream applications, EPE 2017 defines its own interchange format for morpho-syntactico-semantic dependency graphs. Unlike a venerable string of tabular-separated (CoNLL-like) file formats, the EPE serialization of dependency representations is tokenization-agnostic (nodes can correspond to arbitrary and potentially overlapping or empty sub-strings of the underlying document), has no hard-wired assumptions about the range of admissible annotations on nodes, naturally lends itself to graphs transcending rooted trees (including dif-

[8] Specifically, the output from the LTH converter of Johansson and Nugues (2007), using its `-conll07` and `-oldLTH` options, respectively.

ferent notions of 'roots' or top-level 'heads'), and straightforwardly allows framework-specific extensions.

The EPE interchange format serializes a sequence of dependency graphs as a stream of JSON objects, using the newline-separated so-called JSON Lines convention. Each dependency graph has the top-level properties id (an integer) and nodes, with the latter being an (ordered) array of node objects. Each node, in turn, bears its own (unique) id (an integer), form (a string, the surface form), and start and end character ranges (integers); all but the id property are optional (e.g. to be able to represent 'empty' or elided nodes). Furthermore, nodes can have properties and edges, where the former is a JSON object representing an (in principle) arbitrary attribute–value matrix, for example containing properties like pos, lemma, or more specific morpho-syntactic features.

The encoding of graph structure in the EPE interchange format is by virtue of the edges property on nodes, whose value is an array of edge objects, each with at least the following properties: label (a string, the dependency type) and target (an integer, the target node). Thus, edges in the EPE encoding are directed from the head (or predicate) to the dependent (or argument). Unlike for nodes, there is no meaningful ordering information among edges, i.e. the value of the edges property is interpreted as a multi-set. Conversely, encoding each edge as its own JSON object makes possible framework-specific extensions; for example, a future UD parser could output an additional boolean property, to distinguish so-called 'basic and 'enhanced' dependencies.

Finally, adopting the terminology of Kuhlmann and Oepen (2016), the EPE interchange format supports the optional designation of one or more 'top' nodes. In classic syntactic dependency trees, these would correspond to a (unique and obligatory) root, while in the SDP semantic dependencies, for example, top nodes correspond to a semantic head or highest-scoping predicate and can have incoming edges. In the JSON encoding, nodes can bear a boolean top property (where absence of the property is considered equivalent to a false value).

Software Support To lower the barrier to entry, the EPE infrastructure makes available a software utility to (a) convert common file formats for dependency representations into the EPE interchange format and (b) preprocess the 'raw' parser inputs into sentence and token units with PoS tagging and lemmatization applied.

Format conversion supports the file formats from the 2007, 2008, 2009, and 2017 CoNLL shared tasks, from the 2014 and 2015 SDP parsing tasks at SemEval, as well as from a couple more specialized parser output format (as specified by participating teams). Most pre-existing formats fail to record sub-string character offsets, but these are required for the generalized interface to EPE 2017 downstream applications. Thus, the converter builds on the robust alignment tool developed by Dridan and Oepen (2013), essentially recovering token-level character offsets by post-hoc anchoring against the original 'raw' document.

For optional preprocessing of running text into pre-segmented parser inputs, the EPE utility implements a 'baseline' stack of simple, yet state-of-the-art preprocessing tools for sentence splitting, tokenization, part of speech tagging, and lemmatization—essentially the same integration of off-the-shelf components described by Velldal et al. (2012). Starting with a re-release of the parser inputs in mid-April 2017, a readily preprocessed variant of the EPE 2017 document collection has been available to prospective participants, to further lower the barrier to entry in the task, say for teams who do not readily have the preprocessing tools for English available.

6 Downstream Applications

For the EPE 2017 task, an initial set of three state-of-the-art downstream applications is supported.

6.1 Biological Event Extraction

Event extraction refers to the detection of complex semantic relations. It differs from pairwise relation extraction in that events (a) have a defined trigger word (usually a verb), (b) can have 1 to n arguments, and (c) can act as arguments of other events, leading to complex nested structures.

The Turku Event Extraction System (TEES) is a machine learning tool developed for the detection of events in biomedical texts (Björne, 2014). In the EPE context, the event dataset used for training and evaluation is the GENIA corpus from the BioNLP 2009 Shared Task, for which TEES was originally built (Kim et al., 2009). This corpus defines nine types of biochemical events annotated for over ten thousand sentences. A typical GENIA annotation could for example take the form of a nested two-

event structure REGULATION$(A,$ BINDING$(B, C))$ for a sentence like:

> *Protein A regulates the binding of proteins B and C*

Similarly to dependency parses, events can also be seen as graphs, with triggers and other entities as the nodes, and event arguments as the edges. The trigger entity acts as the root node of the subgraph that is a single event, and as the child node for argument edges of any nesting events. TEES is built around the event graph concept, treating event extraction as a graph prediction task implemented with consecutive SVM classification steps.

TEES event prediction proceeds in three main steps. First, *entities* are detected by classifying each surface token into one of the entity classes, or as a negative. Second, event argument *edges* are predicted for each valid pair of detected entities. In the resulting graph there can be only one entity per word token, but multiple events can be annotated for a single word. Therefore, the final step consists of *unmerging* predicted, overlapping events to produce the final event graph. As an optional fourth step, binary *modifiers* (such as negation or speculation) can be predicted for each event.

TEES relies heavily on dependency parses for machine learning example generation. The dependency parse graphs and the event annotation graphs are aligned at the level of word tokens, after which the prediction of an event graph for a sentence can be thought of as converting the syntactic dependency parse into the semantic event graph. In entity detection, features include PoS tags, information about nearby tokens in the linear order, but also token and dependency n-grams built for all dependency paths within a limited distance, originating from the candidate entity token. In edge detection, the primary features are built from n-grams constructed from the *shortest path of dependencies*.

Annotated event entities may not correlate exactly with the syntactic tokenization, so entities are aligned with the parses by using a heuristic to find a single head token for each entity. This means that in addition to the dependency graph, and PoS and dependency type labeling, the granularity of the tokenization can influence TEES performance.

6.2 Opinion Analysis

The opinion analysis system by Johansson and Moschitti (2013) marks up expressions of opinion and emotion in running text. It uses the annotation model and the annotated corpus developed in the MPQA project (Wiebe et al., 2005). The main component in this annotation scheme is the *opinion expression*, which can be realized linguistically in different ways. Examples of opinion expressions are *enjoy*, *criticize*, *wonderful*, or *threat to humanity*. Each opinion expression is connected to an *opinion holder*, a lingustic expression referring to the person expressing the opinion or experiencing the emotion. In some cases, this entity is not explicitly mentioned in the text, for instance if it is the author of the text. Furthermore, every non-objective opinion expression is tagged with a *polarity*: positive, negative, or neutral.

To exemplify, in the sentence

> *"The report is full of absurdities," Xirao-Nima said.*

the expressions *full of absurdities* and *said* are opinion expressions with a negative polarity, and *Xirao-Nima* the opinion holder of these two expressions.

The system by Johansson and Moschitti (2013) required a number of modifications in order to make it more robust to variation in the structure of the input representation. The original implementation made strong assumptions that the input conforms to the linguistic model of the 2008 CoNLL shared task (Surdeanu et al., 2008), which represents sentences using two separate dependency graphs (syntactic and semantic). For this reason, feature extraction functions needed to be reengineered so that they do not assume a particular set of dependency edge labels or part-of-speech tags, or that the dependency graph has any particular structure. Most importantly, this relaxation has an impact on features that represent syntactic relations via paths in the dependency graph: Since the graph is not necessarily a tree, the revised model represents a set of shortest paths instead of a single unique path.

Evaluation Metrics In the EPE task, we evaluated submissions in three different sub-problems, corresponding to the metrics of Johansson and Moschitti (2013):

- marking up opinion expressions in the text, and determining their linguistic subtype; for instance, in the example the expression *full of absurdities*) is an *expressive-subjective element* (ESE) and *said* a *direct-subjective expression* (DSE);

- determining the opinion holder for every extracted opinion expression; for instance, that *Xirao-Nima* is the holder of the two expressions in the example; and
- determining the polarity of each extracted subjective expression, for instance that the two expressions in the examples are both negative.

For each of the above, precision and recall measures were computed. As explained by Wiebe et al. (2005), the boundaries of opinion expressions can be hard to define rigorously, which motivates the use of a 'softer' method for computing the precision and recall: For instance, if a system proposes just *absurdities* instead of the correct *full of absurdities*, this is counted as partially correct.

Furthermore, for the detailed analysis we evaluated the opinion holder extractor separately, using gold-standard opinion expressions. We refer to this task as *in vitro holder extraction*. The reason for investigating holder extraction separately is that this task is highly dependent on the design of the dependency representation, and as we will see in the empirical results this is also the sub-problem where we see most of the variation in performance. In vitro holder extraction scores were used for the overall ranking of submissions when averaging F_1 across the three downstream applications.

6.3 Negation Resolution

The negation resolution system (Sherlock; Lapponi et al., 2012, 2017) determines, for a given sentence, the scope of negation cues. The system is built on the annotations of the Conan Doyle negation corpus (CD; Morante and Daelemans, 2012), where *cues* can be either full tokens (e.g. *not*) or subtokens (*un* in *unfortunate*) and their *scopes*, i.e. the (sub-)tokens they affect. Additionally, in-scope tokens are marked as *negated events* or *states*, provided that the sentence in question is factual and the events in question did not take place. In the example

> *Since* {*we have been so*} ⟨*un*⟩{*fortunate as to miss him*} *[...]*

the prefix cue (in angle brackets) negates the proposition *we have been so fortunate as to miss him* (i.e. its scope, in braces), and *fortunate* (underlined) is its negated event.

Sherlock looks at negation resolution as a classical sequence labeling problem, using a Conditional Random Field (CRF) classifier. The token-wise annotations in CD contain multiple layers of information. Tokens may or may not be negation cues and they can be either in or out of scope; in-scope tokens may or may not be negated events, and are associated with each of the cues they are negated by. Moreover, scopes may be (partially or fully) overlapping, with cues affecting other cues and their scopes. Before presenting the CRF with the annotations, Sherlock flattens the scopes, converting the CD representation internally by assigning one of six labels to each token: out-of-scope, cue, substring cue, in-scope, event, and negation stop (defined as the first out-of-scope token after a sequence of in-scope tokens), respectively.

The feature set of the classifier includes different combinations of token-level observations, such as surface forms, part-of-speech tags, lemmas, and dependency labels. In addition, we extract both token and dependency distance to the nearest cue, together with the full shortest dependency path. After classification, the hierarchical (overlapping) negation structures are reconstructed using a set of post-processing heuristics. It is important to note that one of these heuristics in previous Sherlock versions targeted a specific morpho-syntactic property directly, to help with factuality detection: When a token classified with as a negated event appeared within a certain range of a token tagged as a modal (the *MD* tag), its label was changed from negated event to in-scope. In order to accommodate arbitrary PoS tag sets, this step was removed.

Standard evaluation measures for Sherlock include scope tokens (ST), scope match (SM), event tokens (ET), and full negation (FN) F_1 scores. ST and ET are token-level scores for in-scope and negated event tokens, respectively, where a true positive is a correctly retrieved token instance of the relevant class. The remaining measures are stricter, counting true positives as perfectly retrieved full scopes, either including (FN) or excluding negated events (SM).

7 Participating Teams

Of the nine participating teams, eight submitted complete, well-formed entries. In the following, we list the teams in the order of their overall rank and briefly characterize their different entries.

The collaborating Paris and Stanford teams (Schuster et al., 2017) test two different parsing strategies, treated as separate submissions to the task: The **Stanford–Paris** entry is a text-to-tree

parser followed by rule-based augmentation resulting in a graph representation, whereas the **Paris–Stanford** entry is a direct text-to-graph parser. The team experiments with eight different representations, of which six are derived from Stanford and Universal Dependencies, and the remaining two are the semantically-oriented DM and PAS representations. The **Szeged** team (Szántó and Farkas, 2017) which ranked between the two entries from Paris and Stanford, tests three different representations. The first representation builds a graph from top-k parse trees weighting each edge according to its frequency. The second representation is a combination of dependency and constituency analyses, and the third and final representation collapses dependency labels that are not useful for the downstream tasks. The Universitat Pompeu Fabra (**UPF**) team (Mille et al., 2017) submitted three entries whose representations range in depth from a surface syntactic tree to a predicate–argument graph. The surface-syntactic tree is obtained with an off-the-shelf transition-based parser, while the latter representations are produced using a series of graph transductions of the surface syntactic tree. The team from the East China Normal University (**ECNU**) (Ji et al., 2017) use a neural network–based parser trained on the Universal Dependencies English treebank. The team tested five versions of their pipeline, varying the tagging component in the pipeline as well as the use of pre-trained embeddings. The **Peking** team (Chen et al., 2017) experimented with three architectures: tree approximation, transition-based, and maximum subgraph parsing.[9] The **Prague** team (Straka et al., 2017) participated with the UDPipe neural transition-based parser trained on several different versions of the Universal Dependencies English data. Finally, the University of Washington (**UW**) team submitted a single run in the DM representation, produced using a neural network–based parser (Peng et al., 2017).

Among them, the eight teams submitted 48 distinct 'runs' (parser outputs for one specific configuration), whose results we summarize in the following section.

[9]Owing to a technical error in the submission from Peking which was only detected late, the official scores do not include evaluation results for the transition-based parser. End-to-end scores on the development segments are, however, available for all downstream applications, suggesting that the Peking transition-based parser performs comparably to their other two parsers.

8 Experimental Results

Table 1 shows a summary of the experimental results, for each downstream task as well as overall average across the three tasks along with rank, broken down by participating team and individual runs. Table 1 further includes information on the type of dependency representation used in the various runs for each team, along with information on training data used to train the parsers and its input data: raw text ('txt') or the supplied segmented and tokenized version of the data ('tt'). The system with the overall best result was the Stanford–Paris system with an overall score of 60.51, followed by the Szeged (58.57) and Paris–Stanford (56.81) teams. The Stanford–Paris system also has the best results for the event extraction and negation resolution subtasks, whereas the Szeged system is the top performer in the opinion analysis subtask.

Dependency Schemes As we can see from Table 1, the participating systems employ a variety of different dependency representations. We observe both syntactic dependency representations (CoNLL, SSyntS, Stanford, UD) and more abstract, semantic (to various degrees) dependency representations (CCD, DM, DSyntS, PAS, PredArg). The overall best performing team (Stanford–Paris) experiment with both basic Stanford Dependencies, UD version 1 (basic and enhanced) dependencies, as well as with various modifications of the latter. Their overall best result is obtained using UD enhanced dependencies. This also gives the overall best result for negation resolution, while for event extraction the run employing Stanford Basic Dependencies works marginally better. In comparing the two main UD representation (basic versus enhanced), it is clear that the enhanced representation actually fares better across all three downstream applications for this system. Note, however, that this generalization is dependent on the use of a large training set (WSJ, Brown, and Genia).

As mentioned previously, the syntactic dependency representations can often be subdivided into function-oriented versus content-oriented representations, where CoNLL is an example of the former and Stanford and UD are examples of the latter. In the shared task, only the Szeged system employed the CoNLL representation. This system is the top performing system for opinion analysis, a result that might in principle indicate a remaining bias in this downstream system, as it was originally

					Event Extraction			Negation Resolution			Opinion Analysis				
Team	Run	Dependencies	Train	In	P	R	F	P	R	F	P	R	F	Avg	#
ECNU	0	UD2	UD2	tt	49.48	39.00	43.62	99.17	45.45	62.33	60.27	57.42	58.81	54.92	
	1	UD2	UD2	tt	50.72	38.97	44.08	99.17	45.45	62.33	62.86	60.04	61.42	55.94	
	2	UD2	UD2	tt	52.24	40.23	**45.46**	99.17	45.45	62.33	62.15	59.75	60.93	**56.24**	5
	3	UD2	UD2	tt	54.53	35.58	43.06	99.18	45.83	**62.69**	62.11	58.17	60.08	55.28	
	4	UD2	UD2	tt	60.69	35.76	45.00	99.15	43.94	60.89	63.32	61.07	**62.17**	56.02	
Paris and Stanford	0	DM	WSJ SDP	txt	59.11	37.71	46.04	99.12	42.80	**59.78**	65.04	51.32	57.37	54.40	
	1	PAS	WSJ SDP	txt	52.39	40.98	45.99	99.09	41.29	58.29	65.80	52.73	58.54	54.27	
	2	UD1B	WSJ SDP	txt	55.79	44.56	**49.55**	99.04	39.02	55.98	65.87	61.30	63.50	56.34	
	3	UD1E	WSJ SDP	txt	57.48	41.64	48.29	99.06	39.77	56.75	66.22	62.43	**64.27**	56.44	
	4	UD1Ep	WSJ SDP	txt	58.55	39.50	47.17	99.03	38.64	55.59	65.10	61.75	63.38	55.38	
	5	UD1EpD	WSJ SDP	txt	55.58	43.37	48.72	99.03	38.64	55.59	66.62	62.03	64.24	56.18	
	6	UD1EpDm	WSJ SDP	txt	58.11	39.19	46.81	99.06	39.77	56.75	64.21	60.27	62.18	55.25	
	7	UD1B	WSJ, B, G	txt	57.69	42.80	49.14	99.05	39.39	56.36	65.78	60.96	63.28	56.26	
	8	UD1E	WSJ, B, G	txt	54.90	44.75	49.31	99.07	40.15	57.14	65.59	62.42	63.97	**56.81**	3
	9	UD1Ep	WSJ, B, G	txt	58.03	43.02	49.41	99.04	39.02	55.98	66.77	61.04	63.78	56.39	
	10	UD1EpD	WSJ, B, G	txt	59.88	40.19	48.10	98.97	36.36	53.18	65.86	60.92	63.29	54.86	
	11	UD1EpDm	WSJ, B, G	txt	58.92	40.07	47.70	99.06	39.77	56.75	64.90	60.56	62.65	55.70	
Peking	0	DM	WSJ SDP	tt	59.28	34.22	43.39	99.15	43.94	60.89	65.63	53.64	59.03	54.44	
	1	CCD	WSJ SDP	tt	58.26	40.07	**47.48**	99.15	44.32	**61.26**	66.57	54.55	59.96	**56.23**	6
	2	DM	WSJ SDP	tt											
	3	CCD	WSJ SDP	tt											
	4	DM	WSJ SDP	tt	55.42	40.95	47.10	99.10	41.67	58.67	65.74	53.66	59.09	54.95	
	5	CCD	WSJ SDP	tt	54.73	42.17	47.64	99.12	42.42	59.41	66.97	54.84	**60.30**	55.78	
Prague	0	UD2	UD2	txt	53.84	36.61	43.58	99.10	41.83	58.83	62.61	57.21	59.79	54.07	
	1	UD2	UD2	tt	56.35	38.21	**45.54**	99.16	44.70	**61.62**	62.31	59.74	**61.00**	**56.05**	7
	2	UD2	UD2, L, P	txt	53.22	37.87	44.25	99.12	42.97	59.95	63.45	54.63	58.71	54.30	
	3	UD2	UD2	txt	51.91	36.27	42.70	99.12	42.97	59.95	61.26	56.72	58.90	53.85	
	4	UD1	UD2	txt	51.71	37.12	43.22	98.90	34.22	50.85	61.00	56.25	58.53	50.86	
Stanford and Paris	0	SB	WSJ, B, G	txt	56.93	45.03	***50.29***	99.22	48.48	65.13	67.26	60.54	63.72	59.71	
	1	UD1B	WSJ SDP	txt	57.59	40.76	47.73	99.19	46.21	63.05	67.47	61.30	64.24	58.34	
	2	UD1E	WSJ SDP	txt	57.24	40.98	47.76	99.20	46.97	63.75	67.69	61.02	64.18	58.57	
	3	UD1Ep	WSJ SDP	txt	56.76	42.74	48.76	99.21	47.35	64.10	67.43	61.58	64.37	59.08	
	4	UD1EpD	WSJ SDP	txt	58.86	40.51	47.99	99.19	46.21	63.05	66.68	61.95	64.23	58.42	
	5	UD1B	WSJ, B, G	txt	58.75	42.21	49.13	99.22	48.11	64.80	68.18	61.56	64.70	59.54	
	6	UD1E	WSJ, B, G	txt	58.36	44.09	50.23	99.24	49.62	***66.16***	68.86	61.81	65.14	***60.51***	***1***
	7	UD1Ep	WSJ, B, G	txt	62.30	41.55	49.85	99.20	46.97	63.75	68.44	62.25	**65.20**	59.60	
	8	UD1EpD	WSJ, B, G	txt	57.47	44.47	50.14	99.21	47.73	64.45	67.64	62.57	65.01	59.87	
	9	UD1EpDm	WSJ SDP	txt	55.29	43.21	48.51	99.16	44.70	61.62	66.68	61.42	63.94	58.02	
	10	UD1EpDm	WSJ, B, G	txt	57.22	42.83	48.99	99.22	48.48	65.13	67.30	62.01	64.55	59.56	
Szeged	0	CoNLL	WSJ 02–21	tt	60.20	39.69	**47.84**	99.17	45.08	**61.98**	66.73	65.04	65.87	**58.57**	2
	1	CoNLL^{++}	WSJ 02–21	tt	59.09	39.53	47.37	99.14	43.56	60.53	67.04	65.63	***66.33***	58.07	
	2	CoNLL^{--}	WSJ 02–21	tt	57.93	39.13	46.71	99.15	44.32	61.26	66.05	60.45	63.13	57.03	
	3	CoNLL^{++}	WSJ 02–21	tt	55.14	40.48	46.69	99.12	42.80	59.78	65.35	61.28	63.25	56.57	
	4	CoNLL^{++}	WSJ 02–21	tt	55.12	39.41	45.96	99.11	42.05	59.05	63.37	61.66	62.50	55.84	
UPF	0	SSyntS	WSJ 02–21	txt	53.21	41.36	**46.54**	99.12	42.80	**59.78**	66.25	61.19	**63.62**	**56.65**	4
	1	DSyntS	WSJ 02–21	txt	54.06	39.94	45.94	98.15	20.08	33.34	64.65	56.71	60.42	46.57	
	2	PredArg	WSJ 02–21	txt	56.37	39.63	**46.54**	97.96	18.18	30.67	61.03	51.50	55.86	44.36	
UW	0	DM	WSJ SDP	tt	54.86	35.14	42.84	99.06	39.77	56.75	67.31	54.41	60.18	53.26	8

Table 1: Summary of results. The columns show, left to right: team name, run number enumerating multiple team submissions, type of dependency representation, training data used for the parser, input mode (tokenized or running text), precicion, recall, and F_1 across the three downstream applications, average F_1 across applications, and finally the overall rank of the best run for each team. The representation type is indicated by the following codes: UD1 and UD2 (UD version 1 or 2, respectively), B (basic), E (enhanced), Ep (enhanced plus-plus), D (diathesis), Dm (diathesis minus-minus), SB (Stanford Basic), DM (DELPH-IN MRS Dependencies), PAS (Enju Predicate–Argument structure). The training data is indicated using the following codes: UD2 (English Universal Dependency treebank 2.0), B (Brown), G (Genia), WSJ sections of the PTB, SDP (SDP subset of WSJ sections 00–20), L (LinES), and P (ParTUT). The best F_1 scores for each team for each task are indicated in bold, while the globally best scores are indicated with bold and italics.

developed towards this representation. It is not possible, however, to perform a fair comparison of function-oriented and content-oriented representations, since no systems contrast these in their individual runs.

The shared task also features parsers that produce semantic dependency representations (e.g. DM, PAS, and CCD), more specifically the Paris–Stanford, Peking, and UW systems and even though the semantic representations do not lead to top results in any of the downstream tasks, there are still some interesting observations to be gleaned from the results. The Peking system contrasts the DM and CCD representations and the Paris–Stanford system submitted runs both using semantic dependencies (DM and PAS), as well as syntactic dependencies (various UD representations). The UW system submitted only one run of their system (DM), which ranked eighth overall. The Paris–Stanford system thus enables comparison of (one type of) syntactic versus semantic dependency representations. Here we observe a clear difference in the three downstream applications: For the arguably semantic subtask of negation resolution, the run producing DM dependencies actually performs better than the other (syntactic and semantic) variants, whereas the UD basic and UD enhanced representations give superior results for event extraction and opinion analysis, respectively. For the negation task, we also observe that the DM representation outperforms the other semantic representation produced by the Paris–Stanford parser. For the Peking parser, conversely, we find that the CCD representations perform slightly better across all three subtasks compared to DM.

Finally, the Szeged submissions fully embrace the generalized EPE 2017 interface format and present a range of 'hybrid' dependency representations for end-to-end evaluation, e.g. merging graphs from multiple parsers and presenting k-best lists of analyses in one graph. In general, the resulting graphs are likely denser and one could plausibly hope to see positive downstream effects, for example increased recall while maintaining comparable precision levels. Among the current set of Szeged runs, this expectation is not quite met: Their off-the-shelf baseline system (using CoNLL-style dependencies and the comparatively simple parser of Bohnet, 2010) achieves the best Szeged results, averaged across the three subtasks, and ranks second in the overall competition.

Preprocessing Systems also differ in their choice of preprocessing. Whereas the Stanford–Paris, Paris–Stanford, Prague, and UPF systems make use of their own preprocessors, the rest of the teams rely on the segmented and tokenized versions of the data supplied by the task organizers. Only the Prague runs contrast the two different types of preprocessing. From their results (comparing runs 0 and 1), we observe a clear performance difference in all three downstream tasks by varying the preprocessing strategy, where the parser applied to the supplied preprocessed data ('tt') outperforms the parser that uses the Prague in-house preprocessing scheme on raw text ('txt'). We find that the effect of preprocessing is even stronger than the addition of more training data (run 2) for this parser.[10]

Training Data As we see in Table 1, the systems also make use of different training data for their parsers. The training data vary along several dimensions, most notably size and domain. The choice of training data is to a certain extent governed by the availability of data for a certain type of dependency representation. The parsers producing semantic dependencies invariably employ the data sets released with the SemEval tasks on semantic dependency parsing, which comprise sections 00–20 of the Wall Street Journal (Oepen et al., 2014, 2016) and around 800,000 tokens. In comparison, the parsers that rely only on the English UD treebanks (ECNU and Prague) train their systems on a little more than 200,000 tokens. In order to assess the influence of training data on results, we focus here on the systems that systematically vary the data sets used for the training of their parsers (Stanford–Paris, Paris–Stanford and Prague). For both the Stanford–Paris and Prague parsers, a larger training set has a clear positive effect on results. The Paris and Stanford systems employ the largest training set out of all participating systems: a concatenation of the Wall Street Journal, Brown, and GENIA data sets, which in total comprises 1,692,030 tokens. They contrast the use of this large data set with the use of WSJ data in isolation, and find that the best performance across all three subtasks is obtained with the larger data set. Regarding the influence of domain, we can not draw any firm conclusions: The GENIA dataset is

[10]Note however, that the added training data only comprises the additional English UD treebanks LinES and ParTUT, for an additional 87,630 tokens. The additional data sets employed by e.g. the Stanford–Paris team are considerably larger.

taken from the biomedical domain, hence could be seen to provide an element of domain adaptation for the event extraction subtask. However, even though the Stanford–Paris team does achieve the best result for this subtask with the large, aforementioned training data set, it is not possible to isolate the effect of the domain from the size of the data set based on the submitted runs for this parser.

Reflections In general, it is difficult to compare results across different teams due to the fact that these vary along several dimensions: the parser (and its output quality), the representation, input preprocessing, and the amount and domain of training data. The top-ranking system clearly has the advantage of having one of the currently best performing parsers for English, in terms of intrinsic evaluation (Dozat et al., 2017), in addition to a very large training set. It is not always straightforward, however, to correlate published intrinsic evaluation scores with the EPE 2017 end-to-end results, often due to divergent experimental settings along the above dimensions of variation. We see a few cases where intrinsic performance appears to pattern with extrinsic, end-to-end results. Both the Paris–Stanford and Peking parsers (albeit possibly in earlier variants) participated in the 2014 SDP task (Oepen et al., 2014), where Peking scored midly better for the DM target representation—which appears reflected in higher extrinsic scores, in particular for the negation resolution and opinion analysis subtasks. Conversely, the UW parser for the DM target representation currently defines the intrinsic state of the art (Peng et al., 2017), but its performance in the EPE 2017 context is not competetive. UW only submitted one parsing run and did not provide a system description for the task; seeing as they worked from the preprocessed EPE 2017 inputs, we conjecture that there may well be a technical mismatch with what their parser assumes of its input, for example regarding lemmatization conventions.

9 Conclusion & Outlook

In our view, the EPE 2017 task marks a successful first step towards a flexible and freely available infrastructure for extrinsic parser evaluation. We provide all software, data, submissions, and results for public download, in the hope of continued community-driven work in this direction. For example, the wealth of empirical results available from the 2017 task calls for additional error analysis, for example a contrastive, quantitative study of which downstream items are particularly 'hard' or 'easy' to all or sub-sets of participating parsers. In a similar spirit, in-depth qualitative error analysis of individual runs will likely help identify remaining bias in downstream systems for specific types of dependency representations, e.g. in the form of suggesting revisions or additions of features for the various machine learning components. Finally, it would likely be instructive to quantitatively constrast formal graph properties across submissions, e.g. various indicators of 'treeness' and graph 'densitity' (Kuhlmann and Oepen, 2016).

Follow-up experimentation should seek to isolate some of the interacting factors that make interpretation of EPE 2017 results across teams challenging, for example by constructing additional run series like those of the Paris and Stanford teams, or by contrasting these parsers with additional baselines—which could include 'empty' or mechanically produced, nearly content-free dependency graphs as well as parsers that intrinsically have fallen behind the state of the art. Pushing in a different direction, we hope to start experimentation with more abstract dependency representations (e.g. concept graphs like EDS or AMR), where graph nodes need not correspond (one-to-one) to surface tokens.

Looking ahead, inclusion of additional downstream systems would immediately strengthen the EPE infrastructure, of course, and it would naturally drive development towards further automation of the extrinsic evaluation workflow, ideally maybe through a self-help portal that transparently submits user-contributed parser outputs for end-to-end evaluation on a suitable HPC system. The task coorganizers will jointly continue to try and engage a larger community of parser developers and push the EPE infrastructure towards an actively used and community supported extrinsic benchmark.

Acknowledgments

We are grateful to Emily M. Bender, Gosse Bouma, Dan Flickinger, and in particular Joakim Nivre for in-depth discussions of the task design and interpretation of results. The EPE 2017 shared task was in part funded by the Nordic e-Infrastructure Collaboration (NeIC) through their support to the Nordic Language Processing Laboratory (NLPL; http://www.nlpl.eu); Anders Søgaard has been instrumental in making extrinsic parser eval-

uation a core work package in NLPL. The first two authors were supported by the Center for Advanced Study (CAS) at the Norwegian Academy of Science and Letters. Richard Johansson was supported by the Swedish Research Council under grant 2013–4944. We are grateful to our NLPL and CAS colleagues and to the Nordic tax payers.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*. Berlin, Germany, page 2442 – 2452.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria, page 178 – 186.

Jari Björne. 2014. *Biomedical Event Extraction with Machine Learning*. Ph.D. thesis, University of Turku.

Ezra Black, Steve Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Phil Harrison, Don Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, S. Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*. Pacific Grove, USA, page 306 – 311.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Beijing, China, page 89 – 97.

Ekaterina Buyko and Udo Hahn. 2010. Evaluating the impact of alternative dependency graph encodings on solving event extraction tasks. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA, USA, page 982 – 992.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*. Ann Arbor, MI, USA, page 173 – 180.

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 60 – 64.

Michael Collins. 1999. *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania, Philadelphia.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies. A cross-linguistic typology. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, page 4585 – 4592.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*. Genoa, Italy, page 449 – 454.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Manchester, UK, page 1 – 8.

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 2017 CoNLL Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada, page 20 – 30.

Rebecca Dridan and Stephan Oepen. 2013. Document parsing. Towards realistic syntactic analysis. In *Proceedings of the 13th International Conference on Parsing Technologies*. Nara, Japan.

Jacob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez, and Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proceedings of Human Language Technologies: The 2013 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Atlanta, GA, USA, page 617 – 626.

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*. Berlin, Germany, page 495 – 504.

Carlos Gómez-Rodríguez, Iago Alonso-Alonso, and David Vilares. 2017. How important is syntactic parsing accuracy? an empirical evaluation on sentiment analysis. *CoRR* abs/1706.02141. http://arxiv.org/abs/1706.02141.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank. A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33:355 – 396.

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 Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, page 2 – 11.

Tao Ji, Yuekun Yao, Qi Zheng, Yuanbin Wu, and Man Lan. 2017. ECNU at EPE 2017: Universal dependencies representations parser. 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 40 – 46.

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

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*. Tartu, Estonia, page 105 – 112.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics*. Manchester, UK, page 393 – 400.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Boulder, CO, USA, page 1 – 9.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* In press.

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 25 – 30.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. UiO2. Sequence-labeling negation using dependency features. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*. Montréal, Canada, page 319 – 327.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33th Meeting of the Association for Computational Linguistics*. Cambridge, MA, USA, page 276 – 283.

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

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, and Oscar Täckström. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of Association for Computational Linguistics (ACL)*. pages 92–97.

Simon Mille, Roberto Carlini, Ivan Latorre, and Leo Wanner. 2017. UPF at EPE 2017: Transduction-based deep 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 76 – 84.

Yusuke Miyao. 2006. *From Linguistic Theory to Syntactic Analysis. Corpus-Oriented Grammar Development and Feature Forest Model*. Doctoral dissertation, University of Tokyo, Tokyo, Japan.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics*. Columbus, OH, USA, page 46 – 54.

Roser Morante and Walter Daelemans. 2012. ConanDoyle-neg. Annotation of negation in Conan Doyle stories. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey.

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 10th International Conference on Language Resources and Evaluation*. Portorož, Slovenia.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*. Portorož, Slovenia, page 3991 – 3995.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, 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*. Dublin, Ireland, page 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. Genoa, Italy, page 1250 – 1255.

Woodley Packard, Emily M. Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Meeting of the Association for Computational Linguistics*. Baltimore, MD, USA, page 69 – 78.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Meeting of the Association for Computational Linguistics*. Vancouver, Canada, page 2037 – 2048.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey, page 2089 – 2096.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtský. 2011. Influence of parser choice on dependency-based mt. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Stroudsburg, PA, USA, WMT '11, pages 433–439.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtskỳ. 2014. HamleDT 2.0. Thirty dependency treebanks Stanfordized. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, page 2334 – 2341.

Sebastian Schuster, Eric De La Clergerie, Marie Candito, Benoît Sagot, Christopher D. Manning, and Djamé 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 47 – 59.

Milan Straka, Jana Straková, and Jan Hajič. 2017. Prague at EPE 2017: The UDPipe 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 65 – 74.

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 12th Conference on Natural Language Learning*. Manchester, UK, page 159 – 177.

Zsolt Szántó and Richárd Farkas. 2017. Szeged at EPE 2017: First experiments in a generalized syntactic parsing framework. 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 75 – 79.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational Linguistics* 38(2):369 – 410.

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.

Deniz Yuret, Aydin Han, and Zehra Turgut. 2010. Semeval-2010 task 12: Parser evaluation using textual entailments. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Stroudsburg, PA, USA, SemEval '10, page 51 – 56.

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. Marrakech, Morocco, page 213 – 218.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, 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 dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, 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 Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, 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*. Association for Computational Linguistics, Vancouver, Canada, pages 1–19.