EPE 2017

2017 Shared Task on Extrinsic Parser Evaluation

at the Fourth International Conference on Dependency Linguistics and the 15th International Conference on Parsing Technologies

Proceedings of the Shared Task

September 20, 2017
Pisa, Italy

Production and Manufacturing by
Taberg Media Group AB
Box 94, 562 02 Taberg
Sweden

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-945626-74-6

Introduction

The 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017) is a joint initiative of the Fourth International Conference on Dependency Linguistics (DepLing 2017) and the 15th International Conference on Parsing Technologies (IWPT 2017), which are co-located in Pisa (Italy) from September 18 to 22, 2017. EPE 2017 forms part of the joint programme for the two conferences on September 20, 2017, and is expected to give rise to a more informed comparison of dependency representations both empirically and linguistically.

This volume presents the proceedings from the shared task, which are published as a companion to the IWPT 2017 proceedings. The task was organized around a generalized notion of syntacticto-semantic dependency representations, implemented as a uniform interchange format to three state-of-the-art downstream applications, viz. biomedical event extraction, negation resolution, and fine-grained opinion mining. Nine teams submitted parser outputs for end-to-end evaluation against these downstream evaluation.

The developers of each downstream application as well as each of the participating teams were invited to submit a system description for the EPE 2017 proceedings volume. Seven of the teams followed this invitation, where two teams (Paris and Stanford, who had also coordinated preparation of their submissions) opted to prepare a joint system description. All submissions were reviewed by three reviewers, which were recruited from the task co-organizers, members of the participating teams, and a few externals. The final proceedings volume comprises ten papers: a high-level task summary, three application descriptions for the downstream systems, and six system descriptions.

The complete EPE 2017 open-source infrastructure is available to the public; training and evaluation data, the three downstream systems, various conversion and evaluation tools, all parser outputs from participating teams, as well as end-to-end system outputs and scores can be obtained from the following address:

`http://epe.nlpl.eu`

We gratefully acknowledge the funding and technical contributions from the Nordic e-Infrastructure Collaboration (NeIC) through its Nordic Language Processing Laboratory (NLPL), as well as the support we have received from the DepLing and IWPT 2017 chairs, Simonetta Montemagni, Joakim Nivre, Yusuke Miyao, and Kenji Sagae.

Task Co-Organizers:

Jari Björne,
University of Turku

Filip Ginter,
University of Turku

Richard Johansson,
Chalmers Technical University and University of Gothenburg

Emanuele Lapponi,
University of Oslo

Joakim Nivre,
Uppsala University and
Center for Advanced Study at the Norwegian Academy of Science and Letters

Stephan Oepen (chair),
University of Oslo and
Center for Advanced Study at the Norwegian Academy of Science and Letters

Anders Søgaard,
University of Copenhagen

Erik Velldal,
University of Oslo

Lilja Øvrelid,
University of Oslo and
Center for Advanced Study at the Norwegian Academy of Science and Letters

Programme Committee:

Jari Björne
Gerlof Bouma
Jan Buys
Filip Ginter
Richard Johansson
Emanuele Lapponi
Simon Mille
Joakim Nivre
Stephan Oepen
Sebastian Schuster
Djamé Seddah
Weiwei Sun
Anders Søgaard
Erik Velldal
Lilja Øvrelid

Table of Contents

Conference Program

11:30–11:50 *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

11:50–12:05 *EPE 2017: The Biomedical Event Extraction Downstream Application*
Jari Björne, Filip Ginter and Tapio Salakoski

12:05–12:20 *EPE 2017: The Sherlock Negation Resolution Downstream Application*
Emanuele Lapponi, Stephan Oepen and Lilja Øvrelid

12:20–12:35 *EPE 2017: The Trento–Gothenburg Opinion Extraction System*
Richard Johansson

12:35–13:00 *Paris and Stanford at EPE 2017: Downstream Evaluation of Graph-based Dependency Representations*
Sebastian Schuster, Eric De La Clergerie, Marie Candito, Benoît Sagot, Christopher D. Manning and Djamé Seddah

14:30–14:50 *Prague at EPE 2017: The UDPipe System*
Milan Straka, Jana Straková and Jan Hajic

14:50–15:10 *Szeged at EPE 2017: First Experiments in a Generalized Syntactic Parsing Framework*
Zsolt Szántó and Richárd Farkas

15:10–15:30 *UPF at EPE 2017: Transduction-based Deep Analysis*
Simon Mille, Roberto Carlini, Ivan Latorre and Leo Wanner

15:30–16:00 *EPE 2017: Overall Results and Reflections*
Stephan Oepen, Lilja Øvrelid, Filip Ginter and Richard Johansson

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.

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.*

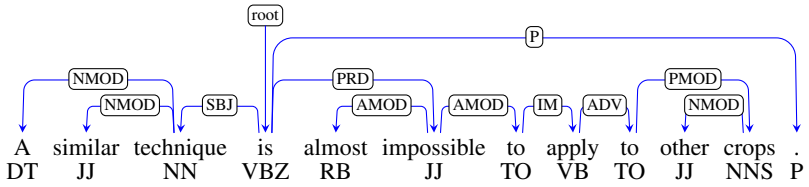
Two 'classic' syntactic dependency trees are presented in 1a and 1b, viz. the target representations from the 2008 CoNLL shared task (Surdeanu et al., 2008) and so-called basic Stanford Dependencies (de Marneffe et al., 2006), respectively. Both are obtained by conversion from the phrase structure annotations 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-

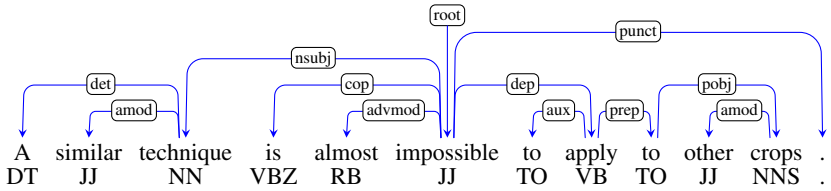
[2] This example is a simplification of a sentence from Section 02 of the PTB and has first been discussed in detail for a broad range of dependency representations by Ivanova et al. (2012) and Oepen et al. (2016).

[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.

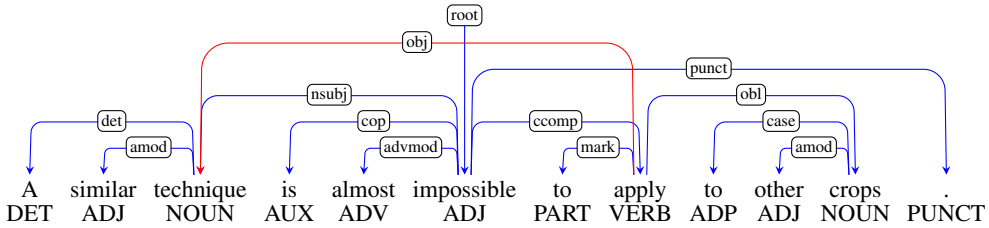
[4] Through its so-called 'enhanced' dependency layer, UD relaxes the constraint that syntactic dependency representations be trees (in the formal sense of connecting each node to the root via a unique directed path): In 1c, for example, *technique* is both a subject dependent of *impossible* and an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



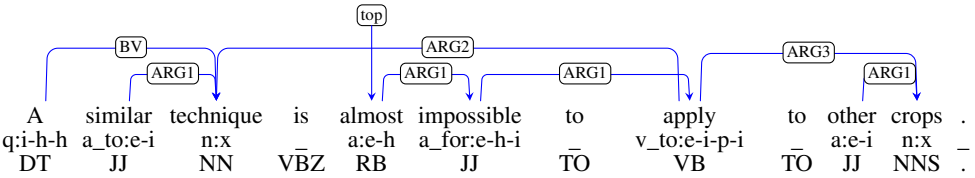
(a) CoNLL: 2008 variant of LTH Dependencies



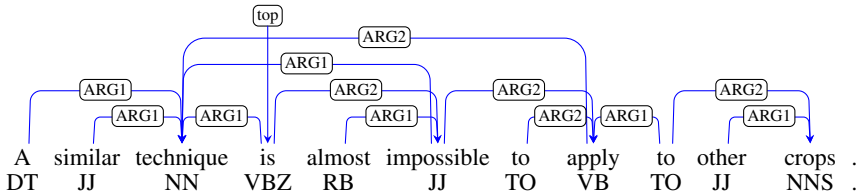
(b) SB: Stanford Basic Dependencies



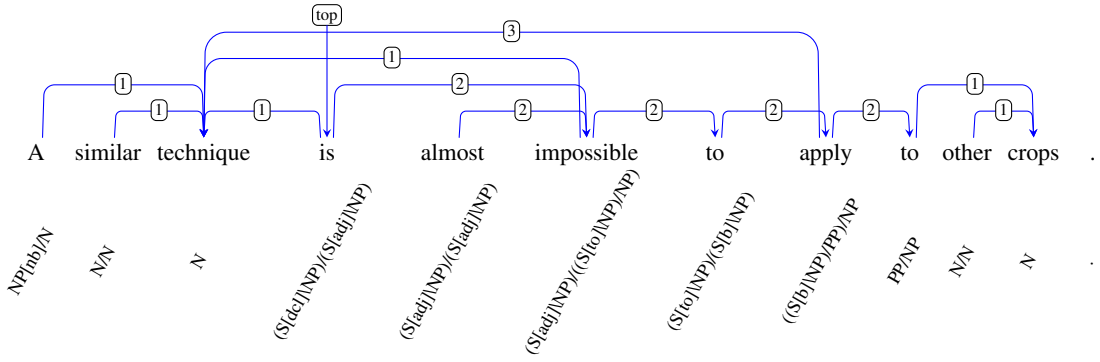
(c) UD: Universal Dependencies (enhanced in red)



(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.

EPE 2017: The Biomedical Event Extraction Downstream Application

Jari Björne, Filip Ginter and Tapio Salakoski
Department of Future Technologies, University of Turku
Turku Centre for Computer Science (TUCS)
Faculty of Mathematics and Natural Sciences, FI-20014, Turku, Finland
firstname.lastname@utu.fi

Abstract

The First Shared Task on Extrinsic Parser Evaluation (EPE 2017) compares different dependency representations by evaluating their impact on downstream applications that utilize these parses for other text mining tasks. In the Biomedical Event Extraction downstream task parses are evaluated by using the Turku Event Extraction System (TEES) with the BioNLP'09 Shared Task as the model challenge. The participants parse the BioNLP'09 dataset, after which the TEES system is run, using the parses as features for predicting events on which the parses are compared. Eight teams submitted a total of 48 runs generated with various parsers, and an additional 13 runs were produced with parsers available via the TEES preprocessing system. Although the TEES system has been developed and optimized using the Stanford Dependencies parsing scheme, among the EPE submissions good performance on the TEES system was achieved also with the Universal Dependencies version 1 scheme.

1 Introduction

The goal of the The First Shared Task on Extrinsic Parser Evaluation (EPE 2017)[1] is to evaluate different dependency representations by comparing their performance on different downstream systems (Oepen et al., 2017). Three downstream applications are used, 1) Biomedical Event Extraction, 2) Fine-Grained Opinion Analysis and 3) Negation Scope Resolution. In this paper we present the results for the Biomedical Event Extraction downstream application and describe the work on upgrading the TEES system to process the varying parse schemes submitted for the task.

Biomedical Event Extraction refers to the process of automatically detecting specific statements of interest from biomedical scientific publications. The biomedical literature is expanding at a rapid pace, with the central PubMed publication database containing as of 2017 over 27 million citations[2]. Text mining is required to search this mass of literature and to extract and summarize the common themes across the millions of publications. Common tasks in biomedical text mining include named entity recognition and normalization (detection of mentions of e.g. genes and mapping them to standardized database ids) as well as interaction extraction (detection of statements of e.g. molecular interactions), where the resulting information can be applied for tasks such as biochemical pathway curation.

In earlier work, biomedical interaction extraction has usually been approached through relation extraction, where all pairs of named entities detected within a span of text (usually a sentence) can either have or not have a stated (sometimes typed and directed) interaction linking them together. On the other hand, events consist of a trigger word (often a verb) and 0–n related arguments, some of which can be other events, allowing complex nested structures. For example, the sentence "Protein A regulates the binding of proteins B and C" can be annotated with a two-event nested structure REGULATION(A, BINDING(B, C)).

2 TEES Overview

The Turku Event Extraction System was originally developed for participation in the BioNLP'09 Shared Task on Biomedical Event Extraction. This task utilized the GENIA corpus, which was the

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

[2] https://www.ncbi.nlm.nih.gov/pubmed/

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 17–24,
Pisa, Italy, September 20, 2017.

first large-scale, annotated resource (+10,000 sentences) for biomedical events (Kim et al., 2008). In total, 24 teams participated in this task, with TEES achieving the first place with 51.95 F-score (Kim et al., 2009). The TEES system has later reached several first places in further shared tasks and has been used as the engine behind several PubMed-scale text mining resources (Björne, 2014).

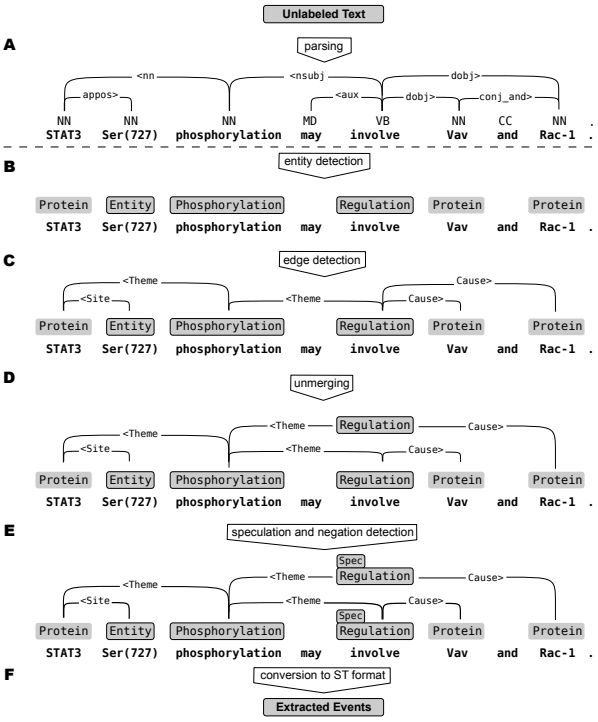


Figure 1: Event extraction. Before event extraction, (A) the text is split into sentences and parsed to produce dependency parse graphs. Relying on this information (B) keywords of interest (entities) are detected in the parsed sentences, after which (C) interaction edges can be detected between the entities. The graph is then (D) unmerged into individual events for which (E) various modifiers can be detected. Finally, the generated events can be (F) exported into the BioNLP Shared Task format. Figure adapted from (Björne et al., 2012).

The TEES system was built around the approach of modelling events as a graph. With relation extraction, graph-based approaches, such as the graph-kernel, have demonstrated good performance (Airola et al., 2008). If the word tokens of the sentence are thought of as the nodes of a graph, the relations can then be seen as edges. This formalism can be extended for events by considering the trigger word as the root node of the entire event, and all arguments as the outgoing edges of this node. Arising from this model of events as graphs, the TEES system is defined as a pipeline of multiclass-classification steps (using SVM^{multiclass}; Tsochantaridis et al., 2005), the first of which detects word entities of interest (nodes), the second the arguments between these nodes (edges) and finally an unmerging step duplicates certain nodes to separate overlapping events. Optionally, a modifier detection step can be used to detect event labels such as negation and speculation (See Figure 1).

Dependency parses can similarly be modelled as graphs, with tokens as nodes and the dependencies as the edges. By starting from an automated dependency analysis, the TEES event extraction process can be seen as converting the syntactic dependency parse graph into the semantic event graph, the two graphs being linked via the shared token nodes. However, while the tokens form directly the nodes of the dependency graph, the entity and trigger nodes of the event graph may not follow the same tokenization, and can often cover multiple tokens (such as "human protein actin"). To align the graphs, TEES uses a heuristic to map each entity to the head token of the span of text covered by that entity.

In the context of the EPE task, there are three main attributes that determine the performance of a parse when used with TEES. First, the tokenization will determine at how fine-grained a level entities get mapped to tokens. Second, the dependencies determine the *shortest path* between any two tokens, the primary source for features such as the n-grams used by TEES to detect event argument edges. Finally, the labeling of the dependency graph affects the features generated: TEES uses the dependency types for edge detection and the token POS tags for entity detection. For the token POS tags, EPE participants could either use a generic POS attribute, or alternatively define both XPOS and UPOS attributes. If both XPOS and UPOS were used, the TEES system was run separately for both tag types and the higher performing result was used as the final result for that submission.

3 Adapting TEES for EPE

A number of improvements were developed for the TEES system in order to more easily apply it for the EPE task. In order to utilize the partici-

pants' submissions, the system was updated to import the JSON EPE file format.

3.1 Importing the EPE parses

The EPE JSON format is the common interchange file format for the EPE task, allowing the participating parses to be used with the different downstream applications. As with the Interaction XML format used in TEES, dependency parses are modelled as graphs in the EPE format, with word tokens becoming the nodes and the dependency relations the edges of the graph. Each node must also be bound unambiguously to the source text with character offsets. While importing the EPE format is quite straightforward, challenges arise from the variation in the valid ways in which parses can be stored in this format.

For example, several of the participating systems produce parses where only tokens of interest have defined nodes. Previously, all parses used by TEES had annotated each word token in the sentence, leading to several mechanisms that relied on this implicit assumption to be true. In order to handle the parses which provide only partial tokenization, the TEES parse importer was updated to whitespace-tokenize and generate dummy tokens for the spans of text not part of partially tokenized parses.

TEES detects events by aligning the dependency parse graph with the event graph, using the parse tokens as shared nodes. However, the original nodes of the event graph are the annotated entities, which may consist of spans of text longer than single tokens. In order to align the graphs, each entity is mapped to a single *head token* using a heuristic for detecting the syntactic head of the parse subgraph contained within the entity. This heuristic assigns each token initial scores (0 for tokens with no dependencies, 1 for tokens connected by dependencies and -1 for special character tokens that should never be the head). After this, the score of each governor token of a dependency is increased to be higher than the score of the dependent token, until scoring no longer changes or a loop cutoff count is reached.

The variation in the parse schemes used by the EPE task participants means that the downstream applications can no longer rely on any parse specific information, such as conventions in POS tag or dependency type naming. In earlier TEES versions the likelihood of loops happening in head token detection was reduced by only considering a subset of primary dependency types for the iterative scoring. Since these dependency types were specific for the Stanford collapsed dependencies scheme they could no longer be used with the various dependencies submitted for the EPE task, so the limitation on dependency types was removed, with only the loop count cutoff now terminating loops if they happen. In practice, the impact on performance was minimal and the new system no longer depends on the specific naming of Stanford collapsed dependency types.

3.2 Parse Format Conversion

In addition to importing the EPE format parses TEES was updated to also export them. The purpose of this update was to utilize the TEES preprocessor for converting various parser output formats into the common EPE interchange format, so the number of supported import formats was also extended. By using the Interaction XML format as the intermediate representation, the TEES parse converter can now reasonably well convert back and forth between the EPE, Penn TreeBank, Stanford Dependencies, CoNLL, CoNLL-X, CoNLL-U and CoreNLP formats. However, reliable conversion from non-text bound formats (which lack token character offsets) into text bound ones is limited by difficulties in aligning some parser outputs with the original text, due to text modifications in the parser output files.

Most parsers do not provide the character offsets in the original text for the generated tokens, and this becomes a problem when such parsers also modify the input text. Common modifications include file format related escapings such as converting left and right parentheses into *-LRB-* and *-RRB-* tags. Such modifications can be detected and reversed relatively easily, but unfortunately many parsers perform also more unpredictable modifications, such as replacing British spellings with American ones (such as "labour" becoming "labor").

Detecting the full list of such modifications would be an open ended problem, so in order to have a decent chance of aligning most modified tokens with the input text the TEES preprocessor now uses the Needleman-Wunsch global alignment algorithm (Needleman and Wunsch, 1970). A fast shortcut for aligning unmodified tokenizations that differ from the original text only in

whitespace is tried at first, with reversals of various common parser modifications then being tried consecutively, with the alignment with the least mismatches being the chosen one. In this manner, the updated TEES preprocessor can convert most parse formats into text-bound ones, although parses with extensive modifications of the original text can still be difficult to align fully.

3.3 Updating the Preprocessor

In the TEES system, importing parses, parsing and other preliminary tasks such as named entity recognition are performed with the TEES preprocessor tool. In previous versions, the preprocessor was implemented as a fixed pipeline of steps. These steps were, in order, conversion of plain text or the BioNLP Shared Task format to the Interaction XML format (the file format used internally by TEES), sentence splitting (with the GENIA sentence splitter; Sætre et al., 2007), named entity recognition (with BANNER; Leaman and Gonzalez, 2008), constituency parsing (with the BLLIP parser; Charniak and Johnson, 2005), dependency conversion (with the Stanford Tools; de Marneffe and Manning, 2008), named entity token splitting, entity syntactic head detection and division into train, devel and test sets. Individual steps in the pipeline could be turned off or modified with parameters, but the pipeline itself could not be redefined.

For the EPE task, the TEES preprocessor was updated into a fully configurable pipeline. The user can now define with the command line interface any list of consecutive steps, with the only limitations imposed by the input and output formats of these steps. Most steps take as both input and output an Interaction XML structure, but e.g. beginning steps can convert an input in the form of a directory of txt files into an Interaction XML structure, and final export steps can likewise convert an Interaction XML structure into a number of output formats.

In earlier TEES versions, parameters could be passed for the individual steps with a separate parameter option, using the step name (e.g. *--steps A,B,C --parameters A.parameter=value*). In the fully customizable pipeline the same step may appear multiple times, so the command line interface was updated to use regular Python syntax, evaluated at run-time, to configure both the steps and their parameters (e.g. *--steps A(parameter=value),B,C*). This approach allows not only using the same step multiple times in the pipeline, but also a consistent way of passing arbitrarily complex Python data structures as parameters for any preprocessing step.

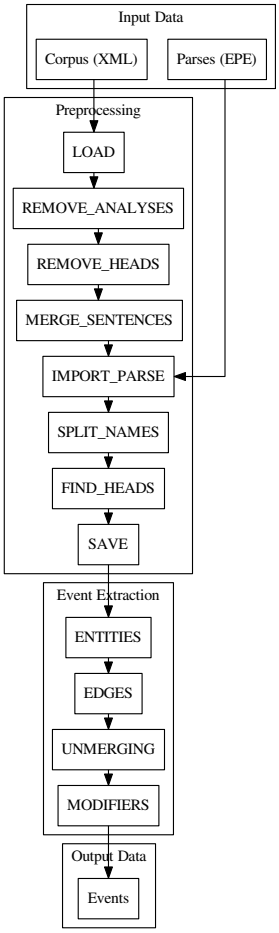


Figure 2: The TEES Preprocessor. The TEES Preprocessor is a fully configurable pipeline, which in the example shown in this figure is used to import an EPE task parse submission from the EPE interchange format and insert it into the BioNLP'09 Shared Task corpus from which existing parse information is removed. Once the parse is inserted, other preprocessing steps such as entity head token detection and entity token splitting are performed to prepare the parsed corpus for event extraction. The Interaction XML output from the Preprocessor is used as the input for Event Extraction, which finally generates the predicted events on which the different parses are evaluated in this EPE downstream task.

The updated TEES Preprocessor can now be used to perform a larger number of supporting tasks useful for event extraction. For example,

the default parses for the BioNLP'09 Shared Task GENIA corpus (installed with TEES) can be exported to the EPE JSON format with the pipeline *[LOAD(corpusName='GE09'), EXPORT(formats='epe')]*. The original TEES preprocessing pipeline was designed to prepare an unparsed corpus for event extraction by parsing it, and this same preprocessing can now be performed with the customized pipeline *[LOAD, GENIA_SPLITTER, BLLIP_BIO, STANFORD_CONVERT, SPLIT_NAMES, FIND_HEADS, SAVE]*. These steps correspond to the fixed preprocessing pipeline of earlier TEES versions.

In evaluating the EPE task, pregenerated parses provided by the participants need to be inserted instead of running a parser (See Figure 2). This can be achieved by changing a few of the preprocessing steps, resulting in the pipeline *[LOAD, REMOVE_ANALYSES, REMOVE_HEADS, MERGE_SENTENCES, IMPORT_PARSE(parseDir='x'), SPLIT_NAMES, FIND_HEADS, SAVE]*. Here, existing parse information and the sentence division based on the parse are removed from the loaded corpus using the *REMOVE_ANALYSES*, *REMOVE_HEADS* and *MERGE_SENTENCES* steps. A new parse is loaded (for example from a directory containing files in the EPE JSON format) using the *IMPORT_PARSE* step, and the rest of the pipeline performs the named entity token splitting, syntactic head detection and saving steps used in the normal parsing pipeline. In this manner, preprocessing steps can be freely combined to quickly define new experimental setups.

4 EPE 2017 Results and Discussion

In total, 48 runs were submitted by eight teams for the event downstream task. In addition, 13 "baseline" runs were generated by the organizers using the various parsers available via the TEES preprocessing pipeline, leading to a total of 61 individual parses for the biomedical event downstream task (See Table 1, Appendix A).

4.1 Baselines

The baseline runs were generated by parsing the BioNLP'09 corpus with parsers available via the TEES preprocessing pipeline, then converting the output to the EPE format, and finally running it through the same EPE evaluation pipeline as the participants' submissions. Therefore, these baselines are not to be seen necessarily as baselines in terms of performance, but as additional points of comparison that could easily be generated by using publicly available parsing tools.

The first baseline in Table 1, the *BioNLP'09 Analyses*, uses the official, BioNLP'09 organizer provided syntactic analyses from 2009. The analyses used were the PTB trees produced with the BLLIP parser using David McClosky's biomedical parsing model and the dependency parses generated with the Stanford Converter (Kim et al., 2009). The TEES preprocessor was used to insert these supporting resources into the BioNLP'09 corpus, followed by exporting them to the EPE format for evaluation as with the other baselines. Running the event extraction system using these official parses resulted in an F-score of 47.87, somewhat lower than the official UTurku result of 51.95 from 2009. The difference is most likely explained by the fact that the 2009 system included e.g. two parallel entity detection systems combined into an ensemble system and used a more corpus-specific, rule-based unmerging system. Therefore, the official BioNLP'09 UTurku result is not directly comparable with the baselines and the EPE submissions, all of which are evaluated using the current version of TEES.

The other baselines use various combinations of the BLLIP parser (commit 558adf6, Jan 9, 2016; Charniak and Johnson, 2005), the Stanford Converter (version 2012-03-09; de Marneffe and Manning, 2008 and the SyntaxNet parser (using the Parsey McParseface and Parsey Universal models; Andor et al., 2016). The BLLIP parser is used with either the standard English model, or David McClosky's biomedical parsing model (McClosky, 2009). The highest performance of 50.48 is achieved with the TEES default parsing settings, using BLLIP with the McClosky biomodel, followed by Stanford conversion using the CCProcessed output format. The BLLIP biomodel parses outperform every other baseline parse regardless of the type of Stanford conversion used, showing a consistent gain from domain adapted parsing. At 47.52 F-score, the SyntaxNet Parsey McParseface model has the best performance out of all the non-biomodel baseline parses, but the Parsey Universal (Universal Dependencies) model is behind all other baselines by several percentage points at 42.79 F-score.

4.2 Submissions

A wide variety of submissions were provided by the task participants, using several different parsers and parsing approaches (See Table 1, Appendix A). The event extraction performance when using the various parses ranged from 42.70 to 50.26 F-score. The highest performance of 50.26 F-score was reached by the Stanford and Paris run 0, using the Stanford Basic scheme with the *XPOS* POS tag type. This parse, like nine out of the ten best submissions (48.99–50.26) was adapted for the biomedical domain by training also on the GENIA corpus.

However, Paris and Stanford run 2, which was trained only on the WSJ corpus, reached a performance of 49.55, less than a percentage point below the best performing domain adapted results. Although too many conclusions cannot be drawn from this single result, it is encouraging to see improved performance for general English parsers on this biomedical text mining task, perhaps indicating less need in the future for time-consuming and resource-dependent domain adaptation.

For the 14 submissions that did not mention using a Universal Dependencies (Nivre et al., 2016, 2017) model, performance was in the range 42.84–50.29. The 21 UD v1.x parses resulted in performances in the range 43.22–50.23, and the nine UD v2.0 submissions were in the range 42.70–45.54. At least on the basis of these results, the UD v1.x scheme can achieve very good performance with an event extraction system such as TEES which was originally developed on the Stanford collapsed dependencies scheme. The UD v2.0 demonstrates overall lower performance, perhaps partially explained by this newer scheme diverging further from the underlying dependency parsing paradigms on which TEES still relies.

5 Conclusions

The First Shared Task on Extrinsic Parser Evaluation (EPE 2017) explored the feasibility of evaluating different parsers in light of the performance of downstream applications that use these parses as supporting analyses for some other text mining task. For the TEES downstream task, 48 participant submissions and 13 internally generated parses were compared.

While the best performance was achieved with the Stanford and Paris run 0, using the Stanford Basic dependencies scheme, also the UD v1.x scheme proved to work effectively with the existing event extraction framework. However, more work is still needed to make use of the UD v2.0 parses for event extraction. In evaluating the results for the EPE biomedical event extraction downstream challenge, it is important to remember that the TEES system has been developed and optimized since 2009 using Stanford collapsed dependencies parses.

Even if TEES is not bound to any single parse scheme, the iterative development and optimization using Stanford collapsed dependencies has no doubt biased the system to some degree towards parsing schemes similar to those collapsed dependencies. Such "overfitting" is of course an issue for any downstream task developed originally using a specific parser, but in any case means that these results must not be assumed to be completely objective evaluations of parser performance or suitability for biomedical event extraction.

The work on adapting the TEES system to not only use the EPE file format, but to work efficiently with the varying schemes of the different parsers, is published as part of the TEES open source project[3]. The improvements to the preprocessing system and the increased robustness in handling different parse schemes should make the system increasingly suitable for more varied text mining tasks. The organizers' work in adapting the downstream applications to use the EPE format, as well as the participants' efforts to export their parses in this common interchange format, build a strong foundation for continued shared evaluation of parsers using downstream text mining applications.

Acknowledgments

We thank Dr. Matthew Shardlow and Professor Sophia Ananiadou of the National Centre for Text Mining (Manchester Institute of Biotechnology, University of Manchester)[4], OpenMinTeD project[5], for assistance with using the BioNLP'09 test set for the purpose of evaluating the EPE task submissions. We also thank CSC – IT Center for Science Ltd for computational resources.

[3] http://jbjorne.github.io/TEES/
[4] http://www.nactem.ac.uk/
[5] http://openminted.eu

References

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics* 9(11):S2.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. *CoRR* abs/1603.06042. http://arxiv.org/abs/1603.06042.

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

Jari Björne, Filip Ginter, and Tapio Salakoski. 2012. University of Turku in the BioNLP'11 shared task. *BMC bioinformatics* 13(11):S4.

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

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*. Association for Computational Linguistics, Stroudsburg, PA, USA, BioNLP '09, pages 1–9. http://dl.acm.org/citation.cfm?id=1572340.1572342.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics* 9(1):10. https://doi.org/10.1186/1471-2105-9-10.

R. Leaman and G. Gonzalez. 2008. BANNER: an executable survey of advances in biomedical named entity recognition. *Pacific Symposium on Biocomputing* pages 652–663.

David McClosky. 2009. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University.

Marie-Catherine de Marneffe and Christopher Manning. 2008. The Stanford typed dependencies representation. In *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48(3):443–453.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1983.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC*.

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

R. Sætre, K. Yoshida, A. Yakushiji, Y. Miyao, Y. Matsubyashi, and T. Ohta. 2007. AKANE system: protein-protein interaction pairs in BioCreAtIvE2 challenge, PPI-IPS subtask. In *Proceedings of the Second BioCreative Challenge Workshop*. Citeseer, pages 209–212.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)* 6(Sep):1453–1484.

A Biomedical Event Extraction Results

Team	Run	Representation	Training	POS	Development Set			Evaluation Set		
					P	R	F	P	R	F
UTurku in BioNLP'09					55.62	51.54	53.50	58.48	46.73	51.95
BioNLP'09 Analyses	BL				61.83	45.33	52.31	60.06	39.79	47.87
BLLIP-Bio, SF-Conv	BL	CCprocessed			60.70	52.04	56.04	58.80	44.22	**50.48**
BLLIP-Bio, SF-Conv	BL	basic			59.69	49.30	54.00	57.18	44.78	50.23
BLLIP-Bio, SF-Conv	BL	collapsed			59.84	51.70	55.47	58.10	43.75	49.91
BLLIP-Bio, SF-Conv	BL	collapsedTree			61.33	50.59	55.44	59.06	43.21	49.91
BLLIP-Bio, SF-Conv	BL	nonCollapsed			58.73	49.13	53.50	56.87	43.81	49.49
BLLIP, SF-Conv	BL	CCprocessed			56.20	45.16	50.08	55.18	41.04	47.07
BLLIP, SF-Conv	BL	basic			54.52	47.96	51.03	52.68	42.93	47.31
BLLIP, SF-Conv	BL	collapsed			64.51	42.93	51.55	58.23	37.15	45.36
BLLIP, SF-Conv	BL	collapsedTree			58.60	45.00	50.91	55.46	39.75	46.31
BLLIP, SF-Conv	BL	nonCollapsed			55.15	48.52	51.62	53.36	41.99	47.00
SyntaxNet	BL				57.41	46.62	51.46	55.73	41.42	47.52
SyntaxNet	BL	UD			49.33	47.07	48.17	46.99	39.28	42.79
ECNU	0	UD v2.0	English 2.0	upos	54.12	45.61	49.50	49.48	39.00	43.62
	1	UD v2.0	English 2.0	upos	54.43	43.66	48.45	50.72	38.97	44.08
	2	UD v2.0	English 2.0	upos	52.91	45.28	48.80	52.24	40.23	**45.46**
	3	UD v2.0	English 2.0	upos	57.85	41.87	48.58	54.53	35.58	43.06
	4	UD v2.0	English 2.0	upos	62.90	43.66	51.54	60.69	35.76	45.00
Paris and Stanford	0	DM	WSJ 00–20 (SDP Sub-Set)		58.26	43.43	49.76	59.11	37.71	46.04
	1	PAS	WSJ 00–20 (SDP Sub-Set)		51.29	46.73	48.90	52.39	40.98	45.99
	2	UD v1 basic	WSJ 00–20 (SDP Sub-Set)		54.71	48.13	51.21	55.79	44.56	**49.55**
	3	UD v1 enh	WSJ 00–20 (SDP Sub-Set)		56.45	47.23	51.43	57.48	41.64	48.29
	4	UD v1 enh++	WSJ 00–20 (SDP Sub-Set)		61.53	46.00	52.64	58.55	39.50	47.17
	5	UD v1 enh++ dia	WSJ 00–20 (SDP Sub-Set)		54.84	49.13	51.83	55.58	43.37	48.72
	6	UD v1 enh++ dia--	WSJ 00–20 (SDP Sub-Set)		57.98	43.94	49.99	58.11	39.19	46.81
	7	UD v1 basic	WSJ, Brown, GENIA		61.05	49.02	54.38	57.69	42.80	49.14
	8	UD v1 enh	WSJ, Brown, GENIA		55.62	49.86	52.58	54.90	44.75	49.31
	9	UD v1 enh++	WSJ, Brown, GENIA		58.68	49.58	53.75	58.03	43.02	49.41
	10	UD v1 enh++ dia	WSJ, Brown, GENIA		60.04	46.84	52.62	59.88	40.19	48.10
	11	UD v1 enh++ dia--	WSJ, Brown, GENIA		61.63	44.77	51.86	58.92	40.07	47.70
Peking	0	DM	SDP 2015		54.46	41.31	46.98	59.28	34.22	43.39
	1	CCD	SDP 2016		56.15	45.72	50.40	58.26	40.07	47.48
	2	DM	SDP 2015		55.06	48.69	51.68			
	3	CCD	SDP 2016		55.78	47.79	51.48			
	4	DM	SDP 2015		52.81	47.79	50.17	55.42	40.95	47.10
	5	CCD	SDP 2016		55.39	48.91	51.95	54.73	42.17	**47.64**
Prague	0	UD v2.0	English 2.0	xpos	52.86	38.85	44.78	53.84	36.61	43.58
	1	UD v2.0	English 2.0	xpos	55.70	42.09	47.95	56.35	38.21	**45.54**
	2	UD v2.0	English, LinES, ParTUT 2.0	xpos	52.69	42.15	46.83	53.22	37.87	44.25
	3	UD v2.0	English 2.0	xpos	53.44	39.58	45.48	51.91	36.27	42.70
	4	UD v1.2	English 2.0	xpos	52.96	41.53	46.55	51.71	37.12	43.22
Stanford and Paris	0	Stanford Basic	WSJ, Brown, GENIA	xpos	55.75	49.92	52.67	56.93	45.03	**50.29**
	1	UD v1 basic	WSJ 00–20 (SDP Sub-Set)	xpos	60.73	46.73	52.82	57.59	40.76	47.73
	2	UD v1 enh	WSJ 00–20 (SDP Sub-Set)	xpos	61.40	47.46	53.54	57.24	40.98	47.76
	3	UD v1 enh++	WSJ 00–20 (SDP Sub-Set)	xpos	58.88	48.97	53.47	56.76	42.74	48.76
	4	UD v1 enh++ dia	WSJ 00–20 (SDP Sub-Set)	xpos	63.62	46.00	53.39	58.86	40.51	47.99
	5	UD v1 basic	WSJ, Brown, GENIA	xpos	58.10	49.19	53.28	58.75	42.21	49.13
	6	UD v1 enh	WSJ, Brown, GENIA	xpos	59.51	50.42	54.59	58.36	44.09	50.23
	7	UD v1 enh++	WSJ, Brown, GENIA	xpos	63.45	46.73	53.82	62.30	41.55	49.85
	8	UD v1 enh++ dia	WSJ, Brown, GENIA	xpos	59.19	51.37	55.00	57.47	44.47	50.14
	9	UD v1 enh++ dia--	WSJ 00–20 (SDP Sub-Set)	xpos	58.15	49.92	53.72	55.29	43.21	48.51
	10	UD v1 enh++ dia--	WSJ, Brown, GENIA	xpos	56.87	50.25	53.36	57.22	42.83	48.99
Szeged	0				59.33	45.89	51.75	60.20	39.69	**47.84**
	1				58.11	45.11	50.79	59.09	39.53	47.37
	2				57.28	46.23	51.17	57.93	39.13	46.71
	3				54.85	45.89	49.97	55.14	40.48	46.69
	4				56.14	44.77	49.81	55.12	39.41	45.96
UPF	0	SSyntS	WSJ 02–21		53.91	46.28	49.80	53.21	41.36	**46.54**
	1	DSyntS	WSJ 02–21		53.56	44.61	48.68	54.06	39.94	45.94
	2	PredArg	WSJ 02–21		55.38	44.38	49.27	56.37	39.63	**46.54**
UW	0	DM	SDP 2015		58.34	45.22	50.95	54.86	35.14	**42.84**

Table 1: EPE Biomedical Event Extraction downstream task results. *SF-Conv* refers to the Stanford Dependencies Converter and *BL* to the baseline parses generated via the TEES preprocessor. In *representation*, 'enh' refers to 'enhanced' and 'dia' to 'diathesis'. If no POS tag is defined, the generic EPE format POS attribute was used. The primary metric of evaluation is the evaluation (test) set F-score, with the best F-score for each team shown in bold. All results are for the BioNLP'09 Task 1 Approximate Span & Recursive Mode, measured using the official BioNLP'09 Shared Task evaluation software.

EPE 2017:
The Sherlock Negation Resolution Downstream Application

Emanuele Lapponi[♣], Stephan Oepen [♣♠], and Lilja Øvrelid[♣♠]

[♣] University of Oslo, Department of Informatics

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

{emanuel|oe|liljao}@ifi.uio.no

Abstract

This paper describes Sherlock, a generalized update to one of the top-performing systems in the *SEM 2012 shared task on Negation Resolution. The system and the original negation annotations have been adapted to work across different segmentation and morpho-syntactic analysis schemes, making Sherlock suitable to study the downstream effects of different approaches to pre-processing and grammatical analysis on negation resolution.

1 Introduction & Motivation

Negation Resolution (NR) is the task of determining, for a given sentence, which part of the linguistic signal is affected by a negation cue. The 2012 shared task at the First Joint Conference on Lexical and Computational Semantics (*SEM) is a notable effort in NR research (Morante and Blanco, 2012), providing the field with a sizable human-annotated corpus for negation (the first outside the biomedical domain), a standardized set of evaluation metrics, as well as empirical NR results from eight competing teams. Our NR system, Sherlock (Lapponi et al., 2012b), ranked first and second in the open and closed tracks, respectively. It has later been used as a pre-processor for Sentiment Analysis (Lapponi et al., 2012a) and, due to its reliance on dependency-based features, as a means of evaluating different dependency representations extrinsically (Elming et al., 2013; Ivanova et al., 2013).

These latter efforts served as an inspiration for the 2017 shared task on Extrinsic Parser Evaluation (EPE 2017; Oepen et al., 2017). Here, participants are invited to provide fully pre-processed and syntactically parsed inputs to three dowstream systems addressing different tasks: biological event extraction (Björne et al., 2017) and fine-grained opinion analysis (Johansson, 2017), in addition to NR. Although Sherlock and the *SEM 2012 negation data have already been used for extrinsic dependency parsing evaluation, the novelty of the current work lies in the fact that the aforementioned earlier work assumed dependency graphs obtained over uniform, gold-standard sentence and token boundaries, as defined by the original token-level annotations of Morante and Daelemans (2012). In contrast, for use of Sherlock in conjunction with a diverse range of parsers that each start from 'raw', unsegmented text, the NR set-up had to be generalized to allow 'projection' of the original, token-level annotations to variable segmentations, both during training and evaluation. In the remainder of this paper we will provide an overview of the task of NR as defined by the annotations in the *SEM 2012 negation data, describe the process of generalizing the gold-standard negation annotations to arbitrary character spans, summarize the generalized Sherlock pipeline, and discuss the EPE 2017 end-to-end results for negation resolution.

2 The Conan Doyle Data

The *SEM 2012 negation data annotate a collection of fiction works by Sir Arthur Conan Doyle (Morante and Daelemans, 2012), henceforth CD. The CD data is comprised of the following annotated stories: a training set of 3644 sentences drawn from The Hound of the Baskervilles, a development set of 787 sentences taken from Wisteria Lodge, and a held-out evaluation set of 1089 sentences from The Cardboard Box and The Red Circle.

The negation annotations in these sets are comprised of so-called negation *cues* (linguistic signals of negation), which can be either full tokens (e.g. *not* or *without*) or sub-tokens (*un* in *unfortunate* or *n't* in contracted negations like *can't*); for each cue,

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 25–30,
Pisa, Italy, September 20, 2017.

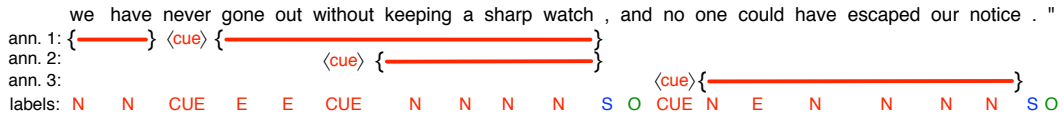


Figure 1: An example of how overlapping CD scope annotations are converted to flat sequences of labels. In this example, an in-scope token is labeled with N, a cue with CUE, a negated event with E, a negation stop with S, and an out-of-scope token with O.

the annotations further comprise its *scope*, i.e. the (sub-)tokens that are affected by the negation. 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. Consider the two following examples from the data, where cues are shown in angle brackets, in-scope tokens in braces, and negated events are underlined:

(1) Since {we have been so} ⟨un⟩{fortunate as to miss him} [...]

(2) If {he was} in the hospital and yet ⟨not⟩ {on the staff} he could only have been a house-surgeon or a house-physician: little more than a senior student.

Notice that negation scopes extend to full propositions (the prefix *un* in example (1) negates that *we have been so fortunate as to miss him*), and that example (2) annotates no negated event, since the sentence is non-factual. Scopes may further be discontinuous, as in example (2). Oftentimes there can be multiple instances of negation within one sentence, and their respective scopes may overlap or nest within each other.

3 Annotation Projection

One generalization that had to be made to Sherlock for use in the EPE 2017 shared task is related to segmentation into 'sentences' and tokens. The original *SEM 2012 negation data is annotated in a token-oriented format, inspired by a series of shared tasks at the conferences for Computational Natural Language Learning (CoNLL), where basic units of annotation are tokens—one per line, in a plain text file, with annotations separated from surface tokens by tabulator characters. Conversely, the EPE 2017 task design starts from 'raw' running text, i.e. participating parsers are expected to apply their own sentence splitting and tokenization. Thus, no specific segmentation conventions are imposed on parser outputs.

In order to use the *SEM 2012 negation data over arbitrary and diverse base segmentations, we developed a separate 'projection' step that (a) converts the gold-standard negation annotations into character-level (stand-off) spans, (b) projects these spans onto a dependency graph provided by a participating parser, and (c) serializes the enriched graph in the token-oriented *SEM 2012 file format, for Sherlock training and evaluation. In other words, annotation projection creates a 'personalized' version of the negation annotations for each individual segmentation, i.e. each distinct parser output. Annotation projection crucially depends on accurate character-level stand-off pointers into the underlying 'raw' document. As these were not available for the original *SEM 2012 negation data, we adapted the alignment tool of Dridan and Oepen (2013) to determine the correspondences from surface tokens in the annotations to sub-strings of the original documents.[1]

Conceptually, annotation projection is fairly straightforward: The *SEM 2012 negation annotations include both sub-token and multi-token negation cues and scopes, for example the prefix *un* or the multi-word *by no means*. Projection of negation annotations onto a different segmentation (with fewer, additional, or just different sentence and token boundaries) may thus move some negation instances into or out of the sub-token and multi-token categories, but both types are treated transparently in Sherlock as well as in the official *SEM 2012 scorer. In principle, we could evaluate final negation predictions (by Sherlock, for a specific parser) against the gold-standard segmentation, by applying a 'reverse' projection from the enriched dependency graph. However, for practical simplicity we opt to evaluate on the 'native' segmentation of the parser directly, i.e. invoke the *SEM 2012 negation scorer on the projected, 'personalized'

[1]The alignment tool applies dynamic programming to compute the globally optimal solution, using the Needleman–Wunsch algorithm, taking into account common normalizations applied during tokenization, e.g. conversion from multi-character ASCII sequences for different-length dashes or various quote marks to corresponding Unicode glyphs.

Features	bigram	trigram	+token	+lemma
token	•	•		
lemma				
pos-tag	•	•	•	
first-order dependency pos-tag				
second-order dependency pos-tag				
dependency relation				
right token distance from cue				
left token distance from cue				
dependency distance from cue				
dependency path from cue				•

Table 1: Features used to train the conditional random field models (on the left), combined with token/lemma, bigram, and trigram features as indicated by the dots. Both bigram and trigram features include backward (e.g. $w_i \wedge w_{i-1}$) and forward variants ($w_i \wedge w_{i+1}$).

gold standard and the actual system output. To ensure that results are comparable across different parsers, we have confirmed that the counts of negation instances remain unaffected, so as to guard against the theoretical possibility of spurious sentence boundaries separating (parts of) a negation cue from (parts of) its arguments.

4 System Description

The task of Negation Resolution, in the context of the CD annotations, is comprised of three subtasks: negation cue identification, scope resolution, and negated event resolution. Sherlock tackles the two latter tasks (assuming that cue identification is either provided by a separate module or accepting gold-standard cues in its input), and basically looks at NR as a classical sequence labeling problem. The main component in the Sherlock pipeline, hence, is Wapiti (Lavergne et al., 2010), an open-source implementation of a Conditional Random Field (CRF) classifier, a discriminative model for sequence labeling.

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) overlapping, as in Figure 1, where the scope of *without* is contained within the scope of *never*.

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), as shown in the final row of Figure 1. Using a fine-grained set of labels (rather than a minimal one, with only out-of-scope, in-scope and event labels) has been shown to yield better performance in this task (Lapponi, 2012). The models for events and in-scope tokens are trained separately; in the event model all N-labeled tokens in Figure 1 have an O label, and all E-labeled tokens in the scope model have an N label.

The features used in the CRF model are listed in Table 1.[2] By default, Sherlock utilizes the same feature set used by Lapponi et al. (2012b) (albeit without the constituents available in the original data), and runs Wapiti with default settings. Sherlock was originally developed to deal with fully connected, single headed dependency trees, and it was updated to be robust to the wider range of dependency graphs submitted to the EPE shared task. The *dependency relation* feature now records the full set of relations for a token (so if token *x* is both *y*'s *a* and *z*'s *b*, its dependency relation feature would be *a,b*). The *dependency distance* and *path from cue* features now assume graphs with (possible) re-entrancies and unconnected nodes, and only record one of possibly several equally shortest paths. If a path from a token to a cue is not found, we simply record a -1 feature.

[2] Wapiti comes with a built-in pattern-based feature expansion system. The patterns used for EPE Sherlock runs are available at https://github.com/ltgoslo/sherlock/tree/master/patterns

	UiO$_2$	Elming et al.	Stanford–Paris #6	Szeged #0	Paris–Stanford #7
ST	85.75	—	**88.57**	86.64	88.19
SM	80.00	**81.27**	80.43	78.42	80.14
ET	**80.55**	76.19	76.55	75.47	71.77
FN	66.41	**67.94**	65.37	62.15	60.48

Table 2: Results of the top-three performers at EPE 2017 (across all tasks), compared to the original UiO$_2$ submission to *SEM 2012 and the best-performing configuration of Elming et al. (2013).

After classification, the full (overlapping) annotations are reconstructed using a set of post-processing heuristics. It is important to note that one of these heuristics in previous Sherlock builds took advantage of the original annotations directly to help with factuality detection; when a token classified 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. This post-processing step has been removed in order to accommodate arbitrary tag sets. The remaining post-processing steps remain unchanged from (Lapponi et al., 2012b). In short, we (1) scan negation cues from left to right; (2) if b is found to the left of a within a fixed-size window, with no punctuation or S-labeled tokens in between, mark it as negated by a; (3) assign all N-negated tokens to the closest cue (again, breaking at punctuation and S-labels); (4) if cue a negates b, assign all of its N-labeled tokens to a as well. The current, EPE-ready release of Sherlock is open source and available for public download.[3]

5 EPE Shared Task Results in Context

Sherlock runs for the EPE shared tasks are evaluated on a subset of the original *SEM evaluation metrics: 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 matched full scopes (SM), and requiring both a perfect scope and event match in the strictest 'full negation' (FN) metric. For the purpose of ranking participating submissions, the EPE 2017 shared task considered the FN metric as primary.

One important difference between previously published Sherlock results is that EPE runs on the held-out data set rely on gold-standard rather than predicted cues, making it hard to relate evaluation results directly. Table 2 shows development set F_1 results from the original *SEM shared task runs (here called UiO$_2$, the name of the original system), the best configuration from Elming et al. (2013), and the top three overall EPE submissions; Table 3 shows the full batch of F_1 scores for all teams and runs, for both the CD development and evaluation sets.

Unlike the EPE runs, UiO$_2$ and Elming et al. (2013) in Table 2 share the same set of pre-processors, and differ only in terms of dependency graphs. The former parses the data using the default MaltParser English model (Nivre et al., 2007), while the latter uses the Mate parser (Bohnet, 2010) converting the resulting phrase-structure trees into dependencies using the Yamada-Matsumoto conversion scheme. Both parsers are trained on Sections 2–21 of the Wall Street Journal portion of the venerable Penn Treebank; additionally, the MaltParser English model is augmented with data from Question Bank.

In-depth analysis and discussion of the EPE shared task results is an ongoing (and daunting) task. It is important to take into consideration that the system was designed and tuned around the original set of sentences, tokens, lemmas, tags, and their conversion to 'basic' Stanford Dependencies from the PTB-style constituent trees in the original CD data. This means that features, label sets, and heuristics were tested (and discarded) empirically, considering the Stanford scheme for syntactic dependency trees. In the extreme, 'chasing' the best possible results in the EPE context would mean repeating a similar process of feature engineering for each submission. With that in mind, simply 'plugging in' a new set of pre-processing annotations nevertheless yields better ST and SM performance than the original system (as shown in

[3] https://github.com/ltgoslo/sherlock.

Team	Run	Development Set				Evaluation Set			
		SM	ST	ET	FN	SM	ST	ET	FN
ECNU	0	80.85	89.10	73.83	62.69	80.10	88.78	66.87	62.33
	1	79.57	87.98	76.63	63.78	80.10	89.14	66.25	62.33
	2	80.00	89.36	73.58	62.69	80.38	88.37	68.30	62.33
	3	79.14	88.69	72.90	61.60	80.10	89.11	68.75	62.69
	4	80.43	87.77	75.96	65.37	78.35	88.28	67.69	60.89
Paris–Stanford	0	76.92	86.82	70.94	61.04	79.72	87.89	65.39	59.78
	1	78.14	87.04	73.08	59.35	78.28	87.45	63.98	58.29
	2	80.43	87.88	69.86	61.04	78.35	87.18	62.00	55.98
	3	80.43	88.24	72.30	61.04	78.64	87.88	61.69	56.75
	4	80.43	88.94	70.47	60.48	78.64	86.53	61.33	55.59
	5	78.26	85.95	68.90	57.61	79.62	87.31	62.20	55.59
	6	77.82	86.90	70.48	58.78	78.93	88.37	59.67	56.75
	7	80.14	88.19	71.77	60.48	78.35	88.42	61.44	56.36
	8	79.14	88.74	69.90	58.20	78.93	87.47	63.40	57.14
	9	78.70	87.71	70.87	59.91	78.93	88.21	63.52	55.98
	10	78.26	88.50	67.96	58.78	77.45	87.00	59.33	53.18
	11	80.43	88.87	72.12	61.04	80.95	88.61	63.79	56.75
Peking	0	80.00	88.01	75.83	63.78	79.33	88.23	67.73	60.89
	1	78.26	87.10	71.22	59.35	78.84	88.80	67.50	61.26
	2	78.26	87.36	73.27	59.35				
	3	79.57	87.37	70.64	61.04				
	4	79.29	87.05	75.60	64.31	79.43	88.42	64.99	58.67
	5	77.38	86.67	70.36	59.35	79.14	88.53	65.84	59.41
Prague	0	76.47	86.85	72.12	58.78	79.13	88.41	63.95	58.83
	1	77.82	87.94	73.93	61.60	77.86	88.16	68.50	61.62
	2	74.62	86.26	73.93	58.78	80.29	89.43	63.75	59.95
	3	77.38	87.71	71.77	59.35	78.54	88.08	64.82	59.95
	4	71.75	85.73	71.22	54.00	69.61	86.74	60.97	50.85
Stanford–Paris	0	80.85	88.23	76.28	64.85	82.08	**89.65**	69.70	65.13
	1	80.85	88.83	75.83	64.31	80.10	88.53	68.69	63.05
	2	81.27	88.34	75.36	63.78	80.38	88.92	68.32	63.75
	3	79.57	88.18	74.88	62.69	81.52	89.56	67.69	64.10
	4	78.70	87.30	75.60	61.60	79.52	88.73	69.38	63.05
	5	80.43	88.95	75.93	63.78	80.67	88.70	**70.34**	64.80
	6	80.43	88.57	76.55	65.37	**82.63**	89.11	**70.34**	**66.16**
	7	80.43	89.93	76.19	62.69	81.23	88.92	68.52	63.75
	8	80.43	89.18	75.00	61.60	82.35	89.71	69.75	64.45
	9	80.00	88.72	74.64	62.15	79.52	89.11	67.29	61.62
	10	**82.10**	**89.99**	77.21	**65.89**	81.80	89.13	70.34	65.13
Szeged	0	78.42	86.64	75.47	62.15	80.00	89.17	67.90	61.98
	1	77.98	87.28	76.78	63.24	79.14	88.19	67.71	60.53
	2	77.98	87.38	72.90	59.91	81.14	89.27	65.20	61.26
	3	78.86	87.07	76.14	63.78	80.38	88.75	64.05	59.78
	4	77.98	85.97	74.26	62.15	79.72	88.91	63.52	59.05
UPF	0	77.38	86.59	73.36	62.69	79.14	88.68	66.66	59.78
	1	44.35	71.09	61.63	32.85	42.46	73.70	53.04	33.34
	2	39.07	67.65	58.33	26.13	38.75	71.16	52.81	30.67
UW	0	76.47	85.79	**77.67**	62.15	77.67	86.99	63.72	56.75

Table 3: Final F_1 scores for all Sherlock runs submitted by the eight participating teams.

Table 2, Stanford–Paris run #6 compared to UiO_2; recall that negated event resolution in the original system was aided by ad-hoc heuristics on the CD tags), which is an encouraging point of departure for further analysis and comparison of the wealth of pre-processing and parsing approaches provided by the EPE shared task.

6 Conclusion & Outlook

In this paper we presented Sherlock, an updated version of one of the top-performing systems in the 2012 *SEM shared task on Negation Resolution. The system was augmented to accept arbitrary tokenization and dependency graphs, and serves as one of three extrinsic evaluators in the EPE 2017 shared task. More in-depth discussion and analysis across different downstream applications is ongoing work; for future work we would like to conduct both quantitative and qualitative error analysis, grounded in a contrastive analysis of which negation instances are comparatively easy or difficult for a majority of systems. Furthermore, we plan to re-tune and calibrate the system around a subset of the EPE submissions, attempting to make the most of the individual strengths of the different segmentations and morpho-syntactic analysis approaches.

Acknowledgments

The development of Sherlock and adaptation for 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). We are grateful to our NLPL colleagues and to the Nordic tax payers.

References

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

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.

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.

Angelina Ivanova, Stephan Oepen, Rebecca Dridan, Dan Flickinger, and Lilja Øvrelid. 2013. On different approaches to syntactic analysis into bi-lexical dependencies. An empirical comparison of direct, PCFG-based, and HPSG-based parsers. In *Proceedings of the 13th International Conference on Parsing Technologies*. Nara, Japan, page 63 – 72.

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

Emanuele Lapponi. 2012. *Why Not! Sequence Labeling the Scope of Negation Using Dependency Features*. Master's thesis, University of Oslo, Oslo, Norway.

Emanuele Lapponi, Jonathon Read, and Lilja Øvrelid. 2012a. Representing and resolving negation for sentiment analysis. In *Proceedings of the 2012 ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction*. Brussels, Belgium.

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

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, page 504 – 513.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task. Resolving the scope and focus of negation. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*. Montréal, Canada, page 265 – 274.

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, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryığıt, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(2).

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

EPE 2017: The Trento–Gothenburg Opinion Extraction System

Richard Johansson
Department of Computer Science and Engineering
University of Gothenburg and Chalmers University of Technology
Gothenburg, Sweden
richard.johansson@gu.se

Abstract

We give an overview of one of the three downstream systems in the Extrinsic Parser Evaluation shared task of 2017: the Trento–Gothenburg system for opinion extraction. We describe the modifications required to make the system agnostic to its input dependency representation, and discuss how the input affects the various submodules of the system. The results of the EPE shared task are presented and discussed, and to get a more detailed understanding of the effects of the dependencies we run two of the submodules separately. The results suggest that the module where the effects are strongest is the opinion holder extraction module, which can be explained by the fact that this module uses several dependency-based features. For the other modules, the effects are hard to measure.

1 Introduction

Applications that use dependency representations of sentences are affected by a number of interacting factors that can be hard to tease apart (Miyao et al., 2008; Johansson and Nugues, 2008b; Elming et al., 2013). We expect the quality of the parser to have an impact: in general, a good parser should also lead to the downstream application being more successful. But this is not the end of the story, because some types of dependency representations may be more or less suitable for a given application, or may be harder or easier for automatic parsers to produce. In the Extrinsic Parser Evaluation (EPE) shared task of 2017 (Oepen et al., 2017),[1] we aim to investigate these questions more systematically by considering several parsers and representations, and measuring their effect on three different downstream applications: opinion extraction, biomedical event extraction, and negation resolution.

In this paper, we describe how the Trento–Gothenburg opinion extraction system (Johansson and Moschitti, 2013) was adapted to the EPE shared task. This system extracts opinion expressions according to the annotation model in the MPQA corpus (Wiebe et al., 2005). The system consists of a number of submodules operating as a pipeline, with a reranker that selects the final output, and some of these modules use features derived from a dependency-parsed representation of the input sentence. For this reason, this application is a useful testbed for measuring the effect of representational design choices and the efficacy of parsers.

We discuss how the opinion extraction system is affected by the dependencies produced by the parsers participating in the EPE shared task. In particular, we are interested in the following questions:

- Can variations in the output of the opinion extraction system be attributed to differences in the dependency inputs, or to other aspects of the input such as tokenization, lemmatization, and tagging?
- In case the dependency structures do have an effect, what parts of the analysis are affected the most?
- Does the type of representation matter, or is the choice of parser more important? For instance, are the semantically oriented dependency representations investigated in the SDP Shared Task (Oepen et al., 2015) useful, or are more traditional syntactic dependencies more suitable?

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

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 31–39,
Pisa, Italy, September 20, 2017.

We first give an introduction to the opinion extraction task as defined by Wiebe et al. (2005), after which we give an overview of the system by Johansson and Moschitti (2013) and how it was adapted to the EPE shared task. Next, we present the opinion extraction results in the shared task, and we carry out some additional experiments to try to understand to what extent and in what ways the quality of the system is affected by the parsers.

2 Task Description

The MPQA project (Wiebe et al., 2005) defined an annotation scheme and created a corpus of annotated expressions of opinions (or private states). The main building blocks in this scheme are three types of linguistic expressions:

- *direct-subjective expressions* (DSEs), which explitly mention emotions and opinions, such as *enjoy* or *disapproval*, or evaluative speech events, such as *criticize* or *label*;
- *expressive-subjective elements* (ESEs), which do not explicitly mention an emotion but in which the choice of words helps us understand an attitude – such as *great*, *heresy*, or *fifth column*;
- *objective statement expressions* (OSEs), which refer to speech events that do not express an opinion – such as *says* or *statement*.

Each instance of these types of expressions is connected to an *opinion holder* (or *source*, in the terminology of Wiebe et al., 2005). This is a lingustic expression that refers to the person expressing the opinion or experiencing the emotion. This person may not be explicitly mentioned in the text, for instance if this is the writer of the text. Furthermore, every DSE and ESE is associated with a *polarity*: positive, negative, or neutral.[2]

To exemplify, in the sentence

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

the expression *full of absurdities* is an ESE with a negative polarity, *said* a DSE, also with a negative polarity, and *Xirao-Nima* the opinion holder of the DSE as well as of the ESE.

[2]Following Choi and Cardie (2010) and Johansson and Moschitti (2013), we mapped the polarity value *both* to neutral, and e.g. *uncertain-positive* to positive, etc.

3 System Description

In this section, we give an overview of the opinion analysis system described by Johansson and Moschitti (2013). In particular, we focus on how the system was adapted for the EPE shared task, and how the parts of the pipeline are affected by the linguistic analysis provided by the participating parsing systems.

As a running example, Figure 1 shows how the sentence above could be analyzed by a hypothetical participant in the EPE shared task. In this case, the representation follows the CoNLL-2008 format (Surdeanu et al., 2008) and consists of a combination of syntactic edges, drawn above the sentence, and semantic edges, drawn below. Each word is tagged using a Penn Treebank-style (Marcus et al., 1993) part-of-speech tag.

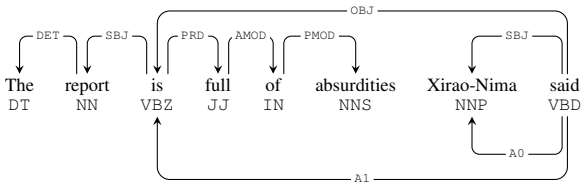


Figure 1: A hypothetical analysis of the example sentence.

3.1 Description of the Pipeline

The system by Johansson and Moschitti (2013) is implemented as a combination of three different submodules – opinion expression extraction, holder extraction, and polarity classification – followed by a reranker that picks the final output based on the results of the previous three steps. Figure 2 shows an overview.

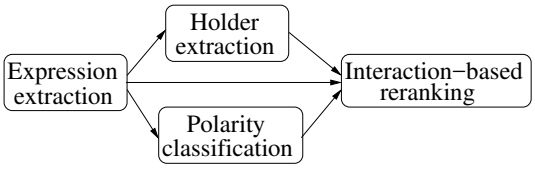


Figure 2: Parts of the opinion extraction system.

3.1.1 Extracting Opinion Expressions

The first step of the pipeline extracts opinion expressions (DSEs, ESEs, OSEs) using a standard sequence labeler (Collins, 2002) operating on the sequence of tokens. This sequence labeler extracts basic grammatical and lexical features (word, lemma, and PoS tag), as well as prior

polarity and intensity features derived from the lexicon created by Wilson et al. (2005). Features based on words (and bigrams), lemmas, and PoS tags, as well as polarity and intensity values, were extracted in a window of size 3 around the word in focus. Expression brackets are encoded using IOB2 tags. The tagging model is trained using the online Passive–Aggressive algorithm (Crammer et al., 2006).

3.1.2 Polarity Classification

Each subjective expression (ESE or DSE) is assigned a polarity value by a separate classifier (a linear SVM). It uses a feature representation consisting of bags of words, word bigrams, and PoS tags inside the expression and in a small window around it, as well as prior polarity and intensity values from the MPQA lexicon.

3.1.3 Opinion Holder Extraction

The module that extracts opinion holders is also implemented as a linear SVM classifier. Given an opinion expression, for instance the DSE *said* in the example, the model assigns a score to each non-punctuation token in the sentence that is not contained in any opinion expression. In addition, it considers two special cases: the *writer* entity, representing the author of the text, and the *implicit* entity, for cases where the holder is not explicitly mentioned. The dependency node (or one of the two special cases) that maximizes this score is selected as the opinion holder.

The feature set used by the classifier in the holder extraction module makes heavy use of the dependency graph. This is because the holder extraction task consists of determining a *relation* between parts of the sentence; it is fairly similar to event participant or semantic role filler extraction; this is particularly true of DSEs, such as *Xirao-Nima* being the holder and filling the speaker role of *said*. For this reason, the feature set used by this module is fairly similar to a typical set of features used in semantic role labeling, relying heavily on paths and other syntactic patterns. Systems for such tasks tend to be sensitive to variations in the input representation (Johansson and Nugues, 2008b; Miyao et al., 2008).

3.1.4 Interaction-based Reranking

Johansson and Moschitti (2013) found considerable improvements in all subtasks by designing *interaction-based* features that describe the relations between different opinion expressions. For instance, these features can describe that

- a DSE (*said*) is connected to an ESE (*full of absurdities*) via an `OBJ` edge;
- a DSE and an ESE have the same opinion holder (*Xirao-Nima*);
- a DSE is connected to an ESE via an `OBJ` edge, and both of them are negative.

Three different groups of interaction features were investigated, based respectively on expressions holders, and polarity. In this work, we used a the combination of all three groups. Several of these features make use of the dependency representation of the sentence.

Considering pairs of opinion expressions makes inference harder, so Johansson and Moschitti (2013) used a reranking approach: first generate the top k solutions from the expression tagger, polarity classifier, and holder classifier, and then rescore the k candidates using a linear scoring model that uses the interaction-based features.

3.2 Changes for the EPE Task

Out of the four modules described above, the holder extraction module and the reranker required significant modifications for the EPE shared task in order to make them agnostic to the structure of the input dependency representation. The opinion extraction tagger and polarity classifier were unchanged from the implementation described by Johansson and Moschitti (2013), but we still expect these modules to see some effect from the expressivity of the PoS tagset and the quality of the PoS and lemma predictions.

The holder extraction and reranking modules in the system by Johansson and Moschitti (2013) depend on the output of the parser by Johansson and Nugues (2008a), which consists of a syntactic dependency tree and a separate semantic dependency graph representing PropBank and NomBank relations (Surdeanu et al., 2008), more or less corresponding to Figure 1. Several aspects of the system, including the design of features and a number of heuristics, require that the input conforms to this format. In particular, the system relies on the fact that the syntactic side of the representation is tree-structured.

In the reengineered system for the EPE shared task, all assumptions about the structure of the input were removed. Most importantly, this applies to several features based on *paths* through

the dependency structure; as described in §3.1.3 and §3.1.4, such features are used by the holder extraction and reranking steps. The previous implementation assumed a tree-structured syntactic graph, which means that the path between two nodes in the graph is unique and easy to compute. For instance, when determining that *Xirao-Nima* is the opinion holder of the DSE *said*, we would extract one feature from the graph in Figure 1 that describes that *Xirao-Nima* is the syntactic subject (`SBJ`). The semantic dependency edge (`A0`, for the semantic role of speaker) would be represented as a separate feature. When relaxing the assumptions about the structural properties of the dependency graph, we instead use features based on *shortest paths*. In case there is no unique shortest path, i.e. if there is more than one with the minimal length, we create separate features for up to 8 paths with the minimal length. For example, the minimal path length between the DSE *said* and its opinion holder *Xirao-Nima* is 1, and there are two paths with this length: one via a syntactic edge (`SBJ`), and one via a semantic edge (`A0`).

Furthermore, in the reengineered system we removed the grammatical voice feature used by Johansson and Moschitti (2013) for the holder extraction module. The reason is that this feature was computed using hard-coded syntactic heuristics that are not applicable for general dependency representations. Hypothetically, we could imagine participating systems representing the voice of a verb as a morphological feature or via dependency edges (e.g. the `nsubj`/`nsubjpass` distinction in the Stanford representation).

Apart from these changes, we made no further adaptation of the system. In particular, we would like to point out that we did not have time to redesign and optimize features for each individual parser, which in principle could lead to a bias towards parsers or representations resembling those used by Johansson and Moschitti (2013). The only feature selection we did for individual parsers was that we investigated whether coarse-grained or fine-grained part-of-speech tags were more effective, and we found in all cases that the latter option was to be preferred.

4 Experimenal Setup

The systems participating in the EPE shared task were evaluated in three different subtasks, which correspond to the evaluations by Johansson and Moschitti (2013):

- marking up opinion expressions in the text, and determining their type (DSE, OSE, or ESE), for instance that the example contains an ESE (*full of absurdities*) and a DSE (*said*);
- determining the opinion holder for every extracted opinion expression, for instance that *Xirao-Nima* is the holder of the two expressions in the example;
- determining the polarity of each extracted subjective expression (that is, DSEs and ESEs), for instance that the two expressions in the examples are both negative.

4.1 Intersection-based Scoring

In all three evaluation scenarios mentioned above, the system marks up spans in the text (opinion expressions and holders), which are then compared to the gold-standard spans. However, the boundaries of opinion expressions in the annotation model of Wiebe et al. (2005) are not rigorously defined and the inter-annotator agreement at the boundary level tends to be low, particularly for ESEs. This makes it natural to apply evaluation metrics that allow for some leniency in evaluating the boundaries.

Johansson and Moschitti (2013) evaluated their system using *intersection-based precision and recall* metrics, which are based on the notion of *span coverage* to measure how well a span s covers another span s':

$$c(s, s') = \frac{|s \cap s'|}{|s'|}$$

The intersection $s \cap s'$ was defined as the set of shared tokens between the two spans, and the set cardinality $|\cdot|$ as the number of tokens. In label-aware evaluation scenarios, $c(s, s')$ was set to 0 if the labels of s and s' differ. The notion of span coverage was then used to define precision and recall measures. The precision measures how well the gold-standard spans cover the predicted spans, and vice versa for the recall:

$$P = \frac{\sum_{\substack{s_i \in S \\ g_j \in G}} c(g_j, s_i)}{|S|} \qquad R = \frac{\sum_{\substack{s_i \in S \\ g_j \in G}} c(s_i, g_j)}{|G|}$$

where S is the set of spans predicted by the system, and G the set of gold-standard spans.

In the EPE shared task, tokenization was provided by the different participating systems, which

required us to change the evaluation procedure to take differences in tokenization into account. To achieve this, we redefined the span coverage to count the number of shared *characters* instead of tokens.

4.2 Dataset Details

We trained and evaluated the system on version 2.0 of the MPQA corpus.[3] This release contains 692 documents, out of which we discarded four documents where the annotation was difficult to align with the text, or which consisted mostly of non-English text. We split the remaining 688 into a training set (450 documents), a development set (90 documents) and a test set (148 documents). The split corresponds to the setup described by Johansson and Moschitti (2013), except that three additional documents were removed. This is a multi-domain corpus but the split was done randomly, so we do not expect that there are significant domain differences between the training and test sets.

5 Results

We first evaluate the system as a whole as described in §4, and then consider individual modules in the pipeline to try to tease out what effects are in play.

5.1 Overall Results

Table 1 shows the results of all the 44 participating parsers evaluated for the three subtasks, as well as the macro-average of the three scores. We observe that the scores for the holder extraction task shows much more variation than for the other tasks. As discussed in §3.1.3, the holder extraction module uses several features derived from the dependencies in the input, so it is logical that this task is the one where we see the largest effects of representational design choices and quality of the parsers.

5.2 Opinion Expression Extraction

We carried out an evaluation of the sequence labeler that marks up and labels opinion expressions (§3.1.1) by running it in isolation, without the interaction-based reranker. Table 2 shows the results. This module uses token-level information such as word forms, lemmas, and PoS tags, but no dependencies. We can thus see this experiment as an extrinsic evaluation of the non-dependency part of the input.

As the results show, the variation among systems is fairly small: if we remove the two outliers (UPF runs 1 and 2), the F-measure standard deviation is just 0.54 and 0.33 in the development and test set, respectively. This is likely because most systems use similar tokenization procedures and Penn Treebank-style tags. The two outliers by the UPF team used an unconventional tokenization scheme that excludes many function words, and this caused difficulties for the opinion expression tagger.

Furthermore, we considered the differences between the expression extraction results in Tables 1 and 2: we would expect that since the interaction-based reranker (§3.1.4) uses several features based on the dependency representation, the parser should have some impact. However, we see no systematic effect: the reranker consistently gives a relative improvement of around 5–7%, which suggests that the choice of representation or parser does not have a strong impact here. This result seems surprising; it is imaginable that the dependency features that have an impact on the reranker are relatively easy for parsers to extract, but we would need to carry out more thorough investigations of features to answer this question in a reliable manner.

5.3 Holder Extraction

We ran the holder extraction module separately, giving the gold-standard opinion expressions as input to the system. We refer to this experiment as *in vitro holder extraction*. Table 3 shows the results of this evaluation. As already mentioned, there are clear differences in performance between the different systems: the F-measure standard deviation on the development and test set is 3.28 and 2.53, respectively.

In an evaluation of this kind, the variation in performance can be explained by several interacting factors, including the design of the sentence representation and the quality of the parser. To exemplify the effects of the choice of parser, we can consider the variation among parsers based on Universal Dependencies (McDonald et al., 2013), of which there are several: the F-measure in this group ranges from about 59 points up to 65 points.

[3] http://www.cs.pitt.edu/mpqa/databaserelease/

One clearly discernible effect of the representation is that the small group of parsers producing semantic dependencies (Paris-Stanford runs 0–1, Peking, UPF 1–2, UW) give considerably lower holder extraction scores than those based on more traditional syntactic dependencies (using Universal Dependencies, Stanford Dependencies, or CoNLL dependencies). The mean F-score on the test set for the group of semantic parsers is 58.76, while the score for the syntactic parsers is 62.94. While this suggests that the more shallow syntactic parsers give more reliable features for this task, it should be noted that these parsers are probably more similar to that originally used by Johansson and Moschitti (2013), and that no effort has been spent on feature design or tuning for the semantically oriented parsers.

Among the systems producing purely syntactic dependencies, it seems that the parser implementation has a stronger impact than the choice of representation: among this group, the best-performing and worst-performing are UD-based, and the few CoNLL-based parsers achieve moderate to high performance (Szeged, UPF run 0).

6 Conclusions

We presented the Trento–Gothenburg opinion extraction system and how it was adapted for the EPE shared task of 2017. The previous implementation by Johansson and Moschitti (2013) made a number of assumptions about the structure of the input – that it consisted of a CoNLL-style syntactic tree and a separate set of semantic dependencies – that had to be relaxed. The modified system does not require a tree-structured input or that semantic edges are stored separately from the syntactic edges. Furthermore, a few hand-crafted features (mainly the voice feature) assuming a CoNLL-style input have been removed.

The outputs of the 44 participating parsers were used to train the opinion extraction system, and we investigated the general performance as well as the performance of individual submodules. It turned out that holder extraction seems to be the part of the analysis that is most affected by the dependencies, and for this subtask we saw much variation among systems. For the other subtasks, the differences attributable to the choice of dependencies are negligible, and token-level linguistic information such as tagging and lemmatization seems to cause much of the variation.

Since the holder extraction task was the one most affected by the dependencies, we ran this module in isolation to highlight the differences. We could see an effect of the choice of representation type, since it seems that semantically oriented parsers, e.g. those coming from the SDP shared task (Oepen et al., 2015), give weaker results. However, we should be careful to draw conclusions from this result, since it could possibly be attributed to the semantic dependency parsers being more different from those used by Johansson and Moschitti (2013), and they may require additional feature engineering and optimization. Apart from this result, there seems to be more variation among parsers using the same representation (e.g. Universal Dependencies) than between different types of syntactic dependencies.

Acknowledgments

The author was supported by the Swedish Research Council under grant 2013–4944.

References

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 269–274.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. University of Pennsylvania, United States, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 2006(7):551–585.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez Alonso, and Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, United States, pages 617–626.

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

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural*

Language Learning. Manchester, United Kingdom, pages 183–187.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, United Kingdom, pages 393–400.

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

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria, pages 92–97.

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 ACL-08: HLT*. Columbus, United States, pages 46–54.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, United States, pages 915–926.

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

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 CoNLL 2008*. Manchester, United Kingdom, 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.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, Canada, pages 347–354.

A Evaluation Scores

		Development				Test			
System	Run	Expr	Holder	Polarity	Macro	Expr	Holder	Polarity	Macro
ECNU	0	58.99	46.36	51.18	52.18	59.10	44.18	49.74	51.01
	1	59.39	48.53	51.55	53.16	58.95	46.52	49.61	51.69
	2	59.58	49.18	**51.74**	53.50	58.85	46.04	49.47	51.45
	3	59.66	47.93	51.51	53.03	58.84	45.31	49.49	51.21
	4	**60.04**	**49.85**	51.47	**53.79**	**59.46**	**46.82**	**49.97**	**52.08**
Paris-Stanford	0	60.07	43.90	51.66	51.88	59.30	43.34	49.23	50.62
	1	60.28	45.60	**51.69**	52.52	59.00	44.04	49.31	50.78
	2	59.88	50.66	51.43	53.99	59.49	48.06	49.60	52.38
	3	59.87	50.40	51.31	53.86	58.91	**49.05**	49.52	**52.49**
	4	59.83	50.45	51.23	53.84	59.28	47.76	49.40	52.15
	5	60.01	**50.93**	51.41	54.12	59.31	48.82	49.69	52.61
	6	59.74	49.87	51.11	53.57	59.52	46.98	49.44	51.98
	7	60.12	50.12	51.23	53.82	**59.55**	47.66	**50.00**	52.40
	8	60.12	50.50	51.21	53.94	59.07	48.16	49.47	52.23
	9	**60.45**	50.41	51.51	**54.12**	59.27	48.54	49.41	52.41
	10	59.98	50.59	51.31	53.96	58.81	48.09	49.14	52.01
	11	60.19	49.92	51.46	53.86	59.16	47.82	49.20	52.06
Peking	0	**59.06**	**45.41**	50.18	**51.55**	58.14	43.33	48.67	50.05
	1	58.65	45.32	**50.52**	51.50	**58.15**	**43.78**	**48.90**	**50.28**
Prague	0	59.51	46.78	50.72	52.34	**59.33**	45.08	49.49	51.30
	1	**59.89**	**47.93**	**51.54**	**53.12**	59.06	**46.46**	49.50	**51.67**
	2	59.48	46.63	50.26	52.12	59.15	44.32	**49.60**	51.02
	3	59.75	45.88	51.23	52.29	58.89	44.46	49.11	50.82
	4	59.42	46.27	50.76	52.15	58.84	44.38	48.79	50.67
Stanford-Paris	0	60.27	51.28	51.40	54.32	59.61	49.52	49.75	52.96
	1	60.76	51.56	52.21	54.84	59.83	49.15	49.80	52.93
	2	60.76	52.06	51.69	54.84	59.89	**50.30**	**49.93**	**53.37**
	3	60.73	51.92	51.84	54.83	59.98	50.21	49.78	53.32
	4	**60.89**	52.84	52.11	55.28	**60.04**	49.91	49.79	53.25
	5	60.81	52.11	51.94	54.95	59.75	49.58	49.69	53.01
	6	60.67	52.93	52.26	55.29	59.73	49.98	49.62	53.11
	7	60.82	52.79	**52.49**	**55.37**	59.92	49.96	49.85	53.24
	8	60.60	**53.00**	51.85	55.15	59.53	49.91	49.76	53.07
	9	60.87	52.02	52.29	55.06	59.89	49.31	49.84	53.01
	10	60.63	52.57	52.04	55.08	59.53	49.50	49.64	52.89
Szeged	0	**59.82**	49.21	**52.47**	53.83	59.33	**50.61**	49.87	53.27
	1	59.76	**49.68**	52.43	**53.96**	59.32	50.52	50.02	**53.29**
	2	59.33	48.99	51.93	53.42	59.05	48.62	49.70	52.46
	3	59.29	48.86	52.12	53.42	**59.53**	48.49	**50.26**	52.76
	4	59.68	48.81	51.89	53.46	58.90	47.91	49.75	52.19
UPF	0	**59.60**	**50.01**	**51.19**	**53.60**	**59.26**	**48.81**	**49.37**	**52.48**
	1	56.04	45.68	47.57	49.76	56.02	45.51	46.81	49.45
	2	54.87	41.08	47.19	47.71	55.12	42.60	46.19	47.97
UW	0	**59.48**	**45.85**	**51.91**	**52.41**	**59.80**	**45.67**	**50.15**	**51.87**

Table 1: F-scores on the development and test sets. For each subtask, the best result for each team is in boldface and the best result overall is underlined.

System	Run	Development			Test		
		P	R	F	P	R	F
ECNU	0–3	65.62	49.27	56.28	66.69	48.84	56.38
	4	**66.06**	**49.68**	**56.71**	**66.83**	**49.38**	**56.80**
Paris-Stanford	0	**66.51**	50.43	57.37	**66.58**	49.21	56.59
	1	66.05	50.69	57.36	65.95	49.35	56.46
	2–11	66.26	**50.73**	**57.46**	66.27	**49.52**	**56.68**
Peking	0–1	**65.36**	**48.16**	**55.45**	**66.33**	**47.90**	**55.63**
Prague	0	65.26	49.61	56.37	66.20	**48.85**	56.22
	1	**65.71**	**50.00**	**56.79**	**66.48**	48.83	**56.31**
	2	65.06	49.30	56.10	66.47	48.70	56.21
	3	65.27	49.65	56.40	65.99	48.53	55.93
	4	65.22	49.35	56.19	65.99	48.67	56.02
Stanford-Paris	0–10	**65.87**	**50.46**	**57.15**	**66.45**	**49.76**	**56.90**
Szeged	0–4	**65.76**	**50.18**	**56.92**	**66.24**	**49.25**	**56.50**
UPF	0	**64.99**	**49.81**	**56.40**	**66.33**	**49.32**	**56.57**
	1	60.89	46.80	52.92	62.31	45.73	52.75
	2	62.04	45.12	52.24	62.09	43.81	51.37
UW	0	**65.76**	**50.18**	**56.92**	**66.24**	**49.25**	**56.50**

Table 2: Expression extraction scores without reranking.

System	Run	Development			Test		
		P	R	F	P	R	F
ECNU	0	62.28	60.46	61.36	60.27	57.42	58.81
	1	64.72	**63.71**	64.21	62.86	60.04	61.42
	2	64.94	63.31	64.12	62.15	59.75	60.92
	3	63.92	62.14	63.02	62.11	58.17	60.08
	4	**65.33**	63.41	**64.35**	**63.32**	**61.07**	**62.17**
Paris-Stanford	0	66.13	51.96	58.19	65.04	51.32	57.37
	1	66.80	51.48	58.15	65.80	52.73	58.55
	2	67.65	63.62	65.57	65.87	61.30	63.50
	3	67.61	62.91	65.18	66.22	**62.43**	**64.27**
	4	68.15	63.71	65.86	65.10	61.75	63.38
	5	67.83	**64.13**	**65.93**	66.62	62.03	64.24
	6	66.34	63.63	64.96	64.21	60.27	62.18
	7	67.57	62.35	64.85	65.78	60.96	63.28
	8	67.10	63.81	65.41	65.59	62.42	63.97
	9	68.19	61.71	64.79	**66.77**	61.04	63.78
	10	**68.24**	63.48	65.78	65.86	60.92	63.30
	11	66.33	62.77	64.50	64.90	60.56	62.66
Peking	0	66.02	**54.07**	59.45	65.63	53.64	59.04
	1	**66.21**	54.05	**59.52**	**66.57**	**54.55**	**59.96**
Prague	0	65.41	59.79	62.47	62.61	57.21	59.79
	1	64.21	**61.65**	**62.91**	62.31	**59.74**	**61.00**
	2	**65.59**	57.05	61.02	**63.45**	54.63	58.71
	3	63.50	58.89	61.11	61.26	56.72	58.90
	4	63.93	59.96	61.88	61.00	56.25	58.53
Stanford-Paris	0	69.48	64.01	66.63	67.26	60.54	63.72
	1	69.02	64.18	66.52	67.47	61.30	64.23
	2	70.59	64.94	67.65	67.69	61.02	64.18
	3	70.31	65.16	67.64	67.43	61.58	64.37
	4	70.56	66.42	68.43	66.68	61.95	64.23
	5	70.12	65.31	67.63	68.18	61.56	64.70
	6	**71.49**	65.80	68.53	**68.86**	61.81	65.14
	7	71.16	65.87	68.41	68.44	62.25	**65.20**
	8	71.05	**66.73**	**68.82**	67.64	**62.57**	65.01
	9	69.42	65.00	67.14	66.68	61.42	63.94
	10	70.21	65.85	67.96	67.30	62.01	64.55
Szeged	0	63.98	62.03	62.99	66.73	65.04	65.88
	1	64.53	**62.69**	**63.60**	**67.04**	**65.63**	**66.33**
	2	**66.44**	60.77	63.48	66.05	60.45	63.13
	3	64.93	61.31	63.06	65.35	61.28	63.25
	4	62.68	61.81	62.24	63.37	61.66	62.51
UPF	0	**65.70**	**60.41**	**62.94**	**66.25**	**61.19**	**63.62**
	1	63.87	56.29	59.84	64.65	56.71	60.42
	2	58.98	49.63	53.90	61.03	51.50	55.86
UW	0	**67.96**	**53.94**	**60.14**	**67.31**	**54.41**	**60.17**

Table 3: In vitro holder extraction scores.

ECNU at EPE 2017: Universal Dependencies Representations Parser

Tao Ji, Yuekun Yao, Qi Zheng, Yuanbin Wu, Man Lan
Department of Computer Science and Technology
East China Normal University
{taoji, ykyao, qizheng}.cs@gmail.com
{ybwu, mlan}@cs.ecnu.edu.cn

Abstract

We present the ECNU submission to the *Extrinsic Parser Evaluation (EPE 2017) Shared Task*. Following Kiperwasser and Goldberg (2016), our parser consists of a bidirectional-LSTM (BiLSTM) feature extractor and a multi-layer perceptron (MLP) classifier. We use universal dependencies representation and trained our transition-based projective parser on UD English 2.0 dataset.

In the *EPE 2017 Shared Task*, the official results show that the F1 score of our *ECNU* system is 56.24%.

1 Introduction

In recent years, dependency-based target representations have gained wide popularity among researchers due to their relatively convenient interface to grammatical structure. While there are abundant researches about syntactico-semantic dependency analysis have been developed during this time, dependency-based target representations today also suffer from the problem that different representaion schemes varies significantly. The First Shared Task (Oepen et al., 2017) on Extrinsic Parser Evaluation [1] (EPE2017) seeks to address this issue by estimating the utility of different dependency representations in different downstream applications and comparing the results, which mitigates the burden of various downstream applications that heavily depend on grammatical structure analysis.

EPE 2017 is limited to parsing English text and there are many kinds of representations for English. In the last century, Eva Hajicov proposed The Prague Dependency Treebank (1999), which is still popular nowadays. Then Hiroyasu Yamada and Yuji Matsumoto also proposed the Yamada-Matsumoto scheme (2003). In 2007, Johansson and Nugues defined the LTH format (2007). Later Johansson used Melcuk-style analysis of coordination to solve the CoNLL Shared Task 2008 (Johansson, 2008), and de Marneffe and Manning (2008) developed the Stanford Dependencies (SD). Recently, Oepen et al. (2016) proposed Semantic Dependency Parsing (SDP) and Nivre et al. (2016) also proposed Universal Dependencies (UD) representations. So we can see that even for a single language, great variation across their representations exists.

The downstream application consists of three parts. The first part is Biological Event Extraction (Björne et al., 2011; Björne et al., 2017), referring to the BioNLP 2009 Shared Task (Kim et al., 2010). Biological Event Extraction is the task of recognizing bio-molecular events mentioned in biomedical literature. The second part is Fine-Grained Opinion Analysis (Johansson and Moschitti, 2012; Johansson, 2017) against the MPQA Opinion Corpus (Wiebe et al., 2005). Fine-Grained Opinion Analysis concerns labeling types of expressions in a sentence, which was proposed in the MPQA project (Wiebe et al., 2005), and extracting their opinion holder or polarity. The last part is Negation Scope Resolution (Lapponi et al., 2012, 2017), referring to the 2012 *SEM Shared Task (2012). Negation Scope Resolution mainly concerns cue detection and scope resolution, while in this shared task, we only consider scope resolution because only the latter is sensitive to grammatical analysis.

In this paper, we present our UD representation parsing system for *EPE 2017 Shared Task*. The system contains a BiLSTM feature extractor for feature representation and an MLP classifier for the transition system. The inputs of our system are

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

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 40–46,
Pisa, Italy, September 20, 2017.

word forms and part of speech (POS) tags (coarse-grained and fine-grained) for each token. Based on this input, the system finds a governor for each token, and assigns a universal dependency relation label to each syntactic dependency.

Our official submission obtains 56.24% macro-averaged F1 score on three downstream applications. The highest F1 scores that we achieve on the three downstream systems are 45.46% (7th) for Biological Event Extraction, 62.69% (2nd) for Negation Scope Resolution, and 62.17% (5th) for Fine-Grained Opinion Analysis. The rest of this paper is organized as follows. Section 2 discusses the transition-based model (Kiperwasser and Goldberg, 2016) and our implementation. Section 3 describes our 5 submissions. Finally, we present experimental and official results in Section 4.

2 System Description

We implement a transition-based projective parser following Kiperwasser and Goldberg (2016). The configuration of our parser in this paper is similar as that in (Ji et al., 2017). We describe model and our implementation in the following sections in detail.

2.1 Arc-Hybrid System

In this work, we use the arc-hybrid transition system (Kuhlmann et al., 2011). We first define a stack α, a buffer β, and a set of dependency arcs A. And thus we can define the configuration in our system: $c = (\alpha, \beta, A)$. Given n words sentence $s = w_1, \cdots, w_n$, the configuration will be initialized as $c = (\emptyset, [1, 2, \cdots, n, root], \emptyset)$, in which $root$ is the special root index. The terminal configuration set contains configurations with an empty stack, an arc set and a buffer containing only $root$.

For each configuration $c = (\sigma|s_1|s_0, b_0|\beta, A)$, the arc-hybrid system has 3 kinds of transitions, $T = \{\text{SHIFT}, \text{LEFT}_l, \text{RIGHT}_l\}$ (in the SHIFT transition, the s_1 and s_0 are allowed to be none and in the LEFT transition, the s_1 is allowed to be none):

$$\begin{aligned}
&\text{SHIFT}(c) = (\sigma|s_1|s_0|b_0, \beta, A) \\
&\quad s.t.\ |\beta| > 0 \\
&\text{LEFT}_l(c) = \big(\sigma|s_1, b_0|\beta, A \cup \{(b_0, s_0, l)\}\big) \\
&\quad s.t.\ |\beta| > 0, |\sigma| > 0 \\
&\text{RIGHT}_l(c) = \big(\sigma|s_1, b_0|\beta, A \cup \{(s_1, s_0, l)\}\big) \\
&\quad s.t.\ |\sigma| > 0, s_0 \neq root
\end{aligned}$$

We apply a classifier to determine the best action for a configuration. Following Chen and Manning (2014), we use an MLP with one hidden layer. The score of the transition $t \in T$ is defined as:

$$\begin{aligned}
&MLP_\theta\big(\phi(c)\big) = W^{(2)} \cdot tanh\big(W^{(1)} \cdot \phi(c) + b^{(1)}\big) + b^{(2)} \\
&\text{SCORE}_\theta\big(\phi(c), t\big) = MLP_\theta\big(\phi(c)\big)[t]
\end{aligned}$$

where $\theta = \{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$ are the model parameters, and $\phi(c)$ is the feature representation of the configuration c. $MLP_\theta\big(\phi(c)\big)[t]$ denotes an indexing operation taking the output element which is the class of transition t.

2.2 The Feature Representation

For an input sequence $s = w_1, \cdots, w_n$, we associate each word w_i with a vector x_i:

$$\begin{aligned}
x_i &= e(w_i) \circ e(p_i) \\
x_i &= e(w_i) \circ pe(w_i) \circ e(p_i) \\
x_i &= e(w_i) \circ pe(w_i) \circ e(p_i) \circ e(q_i)
\end{aligned}$$

where $e(w_i)$ is the embedding vector of word w_i, $pe(w_i)$ is the pre-trained embedding vector of word w_i, $e(p_i)$ is the embedding vector of POS tag p_i, and $e(q_i)$ is the embedding vector of coarse-grained POS (CPOS) tag q_i. We use different token representations in different runs. The embeddings $e(w_i), e(p_i), e(q_i)$ are randomly initialized (without pre-training) and jointly trained with the parsing model. Then, in order to encode context features, we use a 2-layer sentence level BiLSTM on top of $x_{1:n}$:

$$\begin{aligned}
\vec{h}_t &= LSTM(\vec{h}_{t-1}, x_i, \vec{\theta}) \\
\overleftarrow{h}_t &= LSTM(\overleftarrow{h}_{t+1}, x_i, \overleftarrow{\theta}) \\
v_i &= \vec{h}_i \circ \overleftarrow{h}_i
\end{aligned}$$

$\vec{\theta}$ are the model parameters of the forward hidden sequence $\vec{h}$. $\overleftarrow{\theta}$ are the model parameters of the backward hidden sequence $\overleftarrow{h}$. The vector v_i is our final vector representation of ith token in s, which has took into account both the entire history $\vec{h}_i$ and the entire future $\overleftarrow{h}_i$ by concatenating the output of the matching Long Short-Term Memory Network (LSTM) cells.

For $\phi(c)$, our function returns concatenated vectors which consist of the top 3 items on the stack, the first item on the buffer, the right-most and left-most modifiers of s_0, s_1 and s_2 and the left-most

modifier of b_0. Thus we can get a total of 4 BiLSTM vectors and 7 embedding vectors as feature representation. A configuration c is represented by:

$$\begin{aligned}\phi(c) =& v_{s_2} \circ v_{s_1} \circ v_{s_0} \circ v_{b_0} \\ & \circ el\big(r(s_2)\big) \circ el\big(r(s_1)\big) \circ el\big(r(s_0)\big) \\ & \circ el\big(l(s_2)\big) \circ el\big(l(s_1)\big) \circ el\big(l(s_0)\big) \\ & \circ el\big(l(b_0)\big)\end{aligned}$$

where $el(\cdot)$ is a function to get the label embedding vector, $r(s_0)$ is a function to get the right-most modifier of s_0 and $l(s_0)$ is a function to get the left-most modifier of s_0.

2.3 Training Details

Our training aims to get as many correct transitions as possible, and thus we need to make sure that the score of correct transitions are always higher than the one of incorrect transitions. To implement this goal, we can maximize the margin between the transitions with highest correct score and those with highest incorrect score. Specifically, we assume T_{gold} is the set of gold transitions at the current configuration c. At each time stamp, the objective function tries to maximize the margin between T_{gold} and $T - T_{gold}$. The hinge loss of a configuration c is defined as:

$$\begin{aligned}Loss_\theta(c) = \big(1 - & \max_{t_o \in T_{gold}} \mathrm{SCORE}_\theta(\phi(c), t_o) \\ & + \max_{t_p \in (T - T_{gold})} \mathrm{SCORE}_\theta(\phi(c), t_p)\big)_+\end{aligned}$$

Our system use the backpropagation algorithm to calculate the gradients of the entire network (including the MLP and the BiLSTM).

Since our parser can only deal with projective dependency trees, we exclude all training examples with non-projective dependencies.

3 Submission Description

We have 5 submissions (ECNU{00, 01, 02, 03, 04}) for the official evaluation period of the task. All of the 5 submissions use the tokenized and sentence-split version of the raw data as provided in the .tt-files by the shared task organizers. Some of the 5 submissions use the UDPipe (Straka and Straková, 2017) system to do part of speech tagging (POS tagging). The UDPipe system is trained on the Universal Dependencies English 2.0 (Nivre et al., 2017) treebank. Our 300 dimensions pre-trained word vectors trained on Wikipedia use fastText [2] (Bojanowski et al., 2016).

- **ECNU00:** "00" uses the UDPipe system for UPOS tagging, XPOS tagging and parsing. It does not use pre-trained word vectors.
- **ECNU01:** "01" uses the UDPipe system for XPOS tagging. Our parser is used for parsing. It does not use pre-trained word vectors.
- **ECNU02:** "02" uses the UDPipe system for XPOS tagging. Our parser is used for parsing. It uses pre-trained word vectors.
- **ECNU03:** "03" uses the UDPipe system for UPOS and XPOS tagging. Our parser is used for parsing. It uses pre-trained word vectors.
- **ECNU04:** "04" uses the UDPipe system for UPOS tagging and the XPOS tags in the provided .tt-file for XPOS tagging. Our parser is used for parsing. It uses pre-trained word vectors.

Our visual examples can be seen in Figure 1. We use the 6th sentence in the training set of the task of Negation Scope Resolution and visualized its dependency tree in each submission. In this process, we use Dependency Viewer [3] as our visualization tool. We can see from the figure that

- only the parser in **ECNU03** uses UPOS tags. Although our submissions except **ECNU03** also contain the UPOS predicted by UDPipe, we do not use them in the process of parsing.
- the XPOS tag of the word "across" in UDPipe is predicted to be *NN* while the one in the .tt-file is *IN*.
- except the **ECNU02** and **ECNU03**, dependency tree in each submission suffers from difference.

4 Experiments

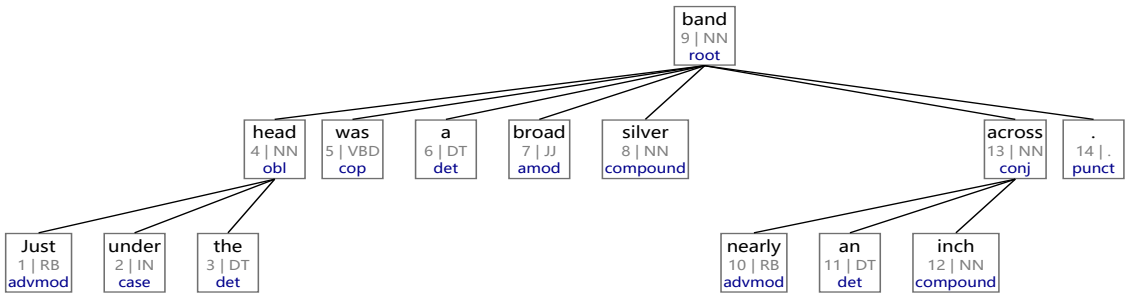
We used the Dynet neural network library to build our system (Neubig et al., 2017).

The hyper-parameters of the final system used for all the reported experiments are:

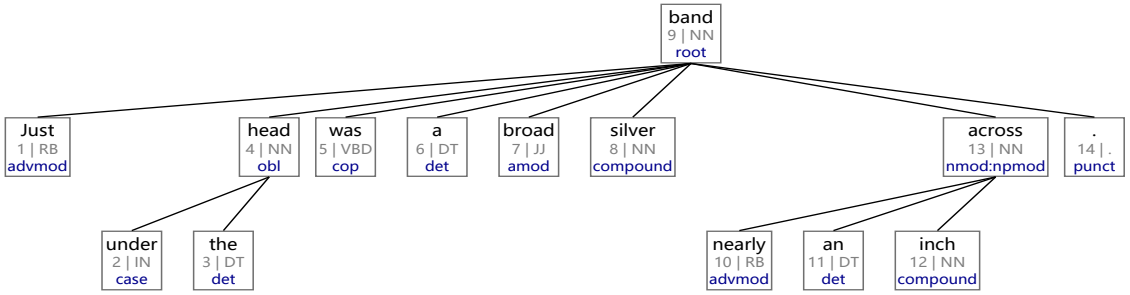
- dimensionality of the embeddings of each word, pos tag, cpos tag and label are 100, 25, 10, 25

[2] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

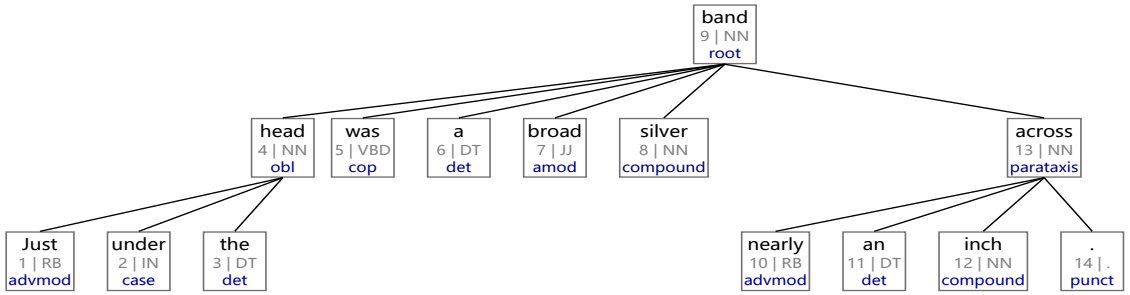
[3] http://nlp.nju.edu.cn/tanggc/tools/DependencyViewer.html



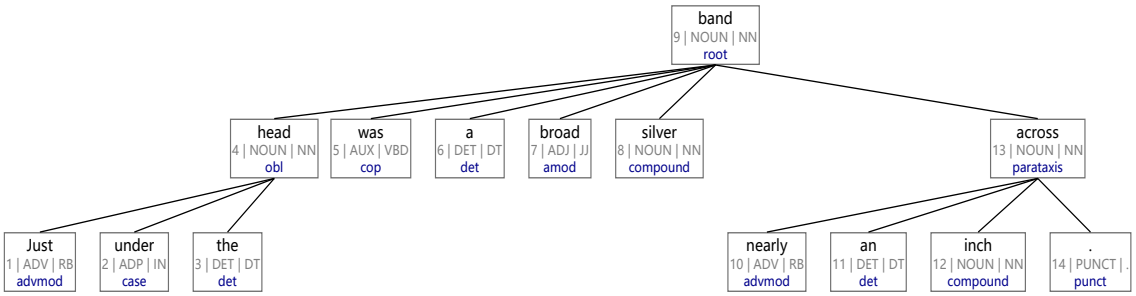
(a) ECNU00



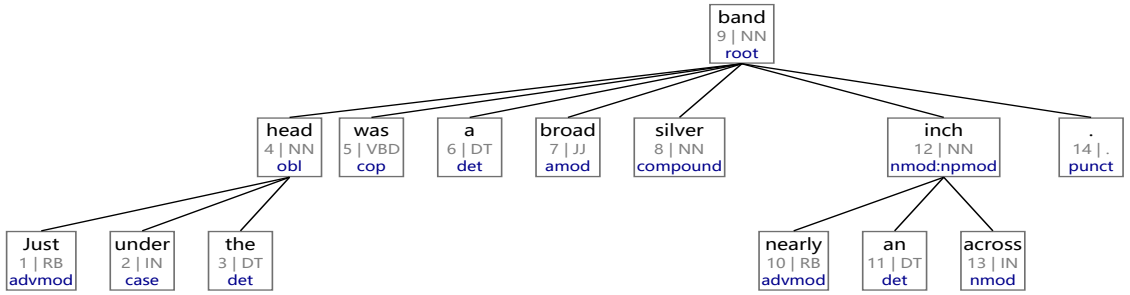
(b) ECNU01



(c) ECNU02



(d) ECNU03



(e) ECNU04

Figure 1: Visual examples of a sentence in the training set in each submission.

submission	dev LAS	EE	NR	OA	Avg
00 (UDPipe)	85.82	43.62	62.33	58.81	54.92
01 (Our)	85.82	44.08	62.33	61.42	55.94
02 (+exWordVec)	86.66	**45.46**	62.33	60.93	**56.24**
03 (+UPOS)	**87.53**	43.06	**62.69**	60.08	55.28
04 (+.tt XPOS)	**87.53**	45.00	60.89	**62.17**	56.02

Table 1: Results of our submissions, including the development set LAS score, the Event Extraction (**EE**) task score, the Negation Resolution (**NR**) task score, the Opinion Analysis (**OA**) task score and the average score of three tasks.

	Input	EE	NR	OA	Avg
ecnu	tt & UDPipe	45.46	**62.33**	60.93	**56.24**
prague	tt	**45.54**	61.62	61.00	56.05
ecnu-04	tt	45.00	60.89	**62.17**	56.02

Table 2: Inputs and Results of our 02 and 04 submissions and prague submissions.

- 100 hidden units for MLP
- 2 BiLSTM layers
- 125 BiLSTM hidden and output layer dimensions
- word dropout with rate of 0.25
- learning rate of 0.1
- Adam optimization algorithm

4.1 Training Data

sentences	tokens	relations
12,543	204,585	24
UPOSs	**XPOSs**	
18	31	

Table 3: Details of our training data. Including the number of sentences, the number of tokens, the number of relations, the number of different types of UPOS and XPOS.

We trained our parser and UDPipe (Straka and Straková, 2017) parser on the UD English 2.0 (Nivre et al., 2017) treebank. The size of our training data is detailed in Table 3.

4.2 ECNU Submissions

Our 5 submissions are detailed in Table 1. ECNU00 and ECNU01 are our baseline systems. Although they have the same score on the development set, the average score we earned on the downstream system was 1.02% higher than UDPipe. We gained 0.84% LAS score increase and 0.3% average downstream system score increase, after adding the pre-trained word vectors. Incorporating pre-trained word vectors can improve parsing and downstream performance. We gained 0.87% LAS score increase and 0.96% average downstream system score reduction, after adding the UPOS features. Incorporating UPOS features only improves parsing performance. We gained 0.74% average downstream system score increase, after replacing the XPOS feature predicted by UDPipe with the XPOS feature in the .tt file.

We use the same sentence segmentation, tokenization and XPOS tags in ECNU01, ECNU02 and ECNU03. We can see in the three submissions that although the parser works better, the performance of the downstream system is not improved. The above analysis allows us to doubt that the results of the parser are not consistent with the results of the downstream system.

4.3 UD v2.0 representation

Task	EE	NR	OA
STDEV	2.24	6.85	2.53

Table 4: The standard deviation of all the results from all teams on each downstream system.

Due to the reason that our respective parsers use different dependency representations and are trained on different training sets, it would be hard to make comparison with other teams. However, both our parser and team *Prague*'s (2017) highest scored parser use the UD v2.0 representation and are trained on English 2.0 training set. Judge

from table 2, we can see that our best results (run02) is 0.19% higher than team *Prague*'s best results. Since our run02 uses the UDPipe system for XPOS tagging and different POS tags would also affect the results, it is inappropriate to compare the performance of two teams' best parser directly.

Both our parser in run04 and team *Prague*'s highest scored parser use the .tt files provided by official as inputs, and our results is 0.03% lower than team *Prague*'s best results. According to extrinsic results, performance of run04 parser is slightly lower than Team prague's highest scored parser.

4.4 Three Downstream Systems

There are various downstream tasks that depend on dependency syntax informations. Degree of dependency may also varies across different downstream tasks. We would like to make some simple analysis on this.

We calculate the standard deviation of all the results from all teams on each downstream system. The results are shown in Table 4. The results show that the event extraction task and the opinion analysis task are less affected by the performance of the parser than the Negation Resolution task.

5 Conclusions

In this paper, we present our dependency parsing system for the *EPE 2017 Shared Task*, which consists of a BiLSTM feature extractor and an MLP classifier. Our system uses UD version English 2.0 dataset. The parser ranks high in the Negation Resolution task. We will continue to improve our system and add reinforcement learning techniques in our future work.

Acknowledgments

We would like to thank the *Extrinsic Parser Evaluation (EPE 2017) Shared Task* organizers (Stephan Oepen et al.).

This research is supported by NSFC(61402175).

References

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

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2011. Extracting contextualized complex biological events with rich graph-based feature sets. *Computational Intelligence* 27(4):541–557. https://doi.org/10.1111/j.1467-8640.2011.00399.x.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Eva Hajiov. 1999. The prague dependency treebank: Crossing the sentence boundary. In *International Workshop on Text, Speech and Dialogue*. pages 20–27.

Tao Ji, Yuanbin Wu, and Man Lan. 2017. A fast and lightweight system for multilingual dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 237–242. http://www.aclweb.org/anthology/K/K17/K17-3025.pdf.

Richard Johansson. 2008. Dependency syntax in the conll shared task 2008 .

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

Richard Johansson and Alessandro Moschitti. 2012. Relational features in fine-grained opinion analysis. *Computational Linguistics* 39(3):317–325.

Richard Johansson and Pierre Nugues. 2007. Lth: semantic structure extraction using nonprojective dependency trees. *Proc of Semeval* 13(4):227–230.

Jin Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'Ichi Tsujii. 2010. Overview of bionlp'09 shared task on event extraction. In *The Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. pages 1–9.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. pages 673–682. http://www.aclweb.org/anthology/P11-1068.

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

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 2012, June 7-8, 2012, Montréal, Canada.*. pages 319–327. http://aclweb.org/anthology/S/S12/S12-1042.pdf.

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.

Roser Morante and Eduardo 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 2012, June 7-8, 2012, Montréal, Canada.*. pages 265–274. http://aclweb.org/anthology/S/S12/S12-1035.pdf.

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 Hajic, Christopher D. Manning, Ryan T. 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, Portorož, Slovenia, May 23-28, 2016.*. http://www.lrec-conf.org/proceedings/lrec2016/summaries/348.html.

Joakim Nivre et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague, http://hdl.handle.net/11234/1-1983. http://hdl.handle.net/11234/1-1983.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*. http://www.lrec-conf.org/proceedings/lrec2016/summaries/887.html.

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

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 88–99. http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Milan Straka, Jana Strakov, 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 61 – 70.

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. https://doi.org/10.1007/s10579-005-7880-9.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. *Proceedings of Iwpt* pages 195–206.

Paris and Stanford at EPE 2017: Downstream Evaluation of Graph-based Dependency Representations

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

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

Departments 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 has also 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 build on the new Universal Dependencies (UD) representation (Nivre et al., 2016) with augmentations that 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 neutralizing several syntactic alternations and adding subjects of infinitival verbs controlled not only by verbs but also by nouns and adjectives.

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.

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 47–59,
Pisa, Italy, September 20, 2017.

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 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 for French that showed that parsing directly to graphs outperforms a two-stage approach of first parsing to dependency trees and then applying a rule-based conversion to obtain graphs.

Regardless of the parsing framework, an obvious method of comparing different models that vary in representation and parser architecture is to evaluate parsing systems in terms of the value they add to downstream tasks. Performing such cross-framework extrinsic evaluation is a challenging enterprise as it requires the development of non-trivial feature extractors that can cope with annotation schemes idiosyncrasies (Crouch et al., 2002; Sagae et al., 2008), as well as the selection of a set of tasks and metrics that can factor out structural divergences of the parses (Miyao et al., 2009; Miwa et al., 2010; Elming et al., 2013).

We therefore took the opportunity of the *Extrinsic Parser Evaluation (EPE) Shared Task* (Oepen et al., 2017) 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 are 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 (Björne et al., 2017) 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 uses 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 (Lapponi and Oepen, 2017) 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:

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

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 basic 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 (Johansson, 2017) 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 an 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 classifier 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 (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 enabling 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 of 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 is treebank-based and the other is grammar-based, and 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 de-

pendency 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. We evaluate the representation as output by the converter in CoreNLP version 3.8.0.

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. For example, the relation label of a nominal modifier (`nmod`) that is introduced with the preposition *by* is `nmod:by`.

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 and extends it in two ways. First, it contains more argumental relations, for example, relations between infinitival verbs controlled by nouns or adjectives and their controllers, and relations between participles and their subjects and objects. Second, this representation neutralizes some syntactic alternations. For example, demoted agents in the form of *by*-phrases are turned into subjects and passive subjects into direct 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.

Some of the differences and commonalities of the various representations are visualized based on an example sentence in Appendix A.

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

	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*.

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 constructs 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 *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 (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

[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.

[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.

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 basic* and *UD diathesis*-- ones) contain a large set of labels. This does not seem to cause much of an issue 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,[4] 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 representation, 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 de-

[4]The official table of results which includes detailed results for each downstream task can be found at http://epe.nlpl.eu.

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.

pendency tree and then augment the output using 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 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. This is particularly suprising as it contradicts the results by Elming et al. (2013). 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.

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

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 training 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 a broader range of domains 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.

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.	Ave.
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, there is no evidence that representations that primarily focus on encoding the argument structures of predicates such as the *DM* and *PAS* representations have benefits over representations that are derived from surface-syntax representations. At the same time, our results indicate 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.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Further, we would like to thank the entire EPE Shared Task organizing committee, and in particular, Stephan Oepen, for organizing this shared task and setting up the downstream task infrastructure. This work was supported in part by gifts from Google, Inc. and IPSoft, Inc. The first author is also supported by a Goodan Family Graduate Fellowship. The Paris authors were partly funded by the French ANR projects ParSiTi (ANR-16-CE33-0021 and SoSweet (ANR-15-CE38-0011-01), as well as by the Program "Investissements d'avenir" ANR-10-LABX-0083 (Labex EFL).

References

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

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 *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*. pages 320–327.

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)*.

Richard Crouch, Ronald M Kaplan, Tracy H King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad-coverage stochastic parser. In *Proceedings of the LREC Beyond PARSEVAL Workshop*. pages 67–74.

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*. pages 243–252.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proceedings of the Coling 2008 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.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez Alonso, and Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2013)*. pages 617–626.

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. 2017. EPE 2017: The Trento–Gothenburg opinion 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 –.

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 and Stephan Oepen. 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 –.

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.

Olivier Michalon, Corentin Ribeyre, Marie Candito, and Alexis Nasr. 2016. Deeper syntax for better semantic parsing. In *Proceedings the 26th International Conference on Computational Linguistics (COLING 2016)*.

Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010. Evaluating dependency representation for event extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. pages 779–787.

Yusuke Miyao, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics* 25(3):394–400.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. 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 (LREC 2012)*. 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, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter, Erik Velldal, and Lilja Øvrelid. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017). 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 –.

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.

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

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the 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 *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 11)*.

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 (NAACL 2015)*.

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.

Kenji Sagae, Yusuke Miyao, Takuya Matsuzaki, and Junichi Tsujii. 2008. Challenges in mapping of syntactic representations for framework-independent parser evaluation. In *Proceedings of the Workshop on Automated Syntactic Annotations for Interoperable Language Resources*.

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)*.

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.

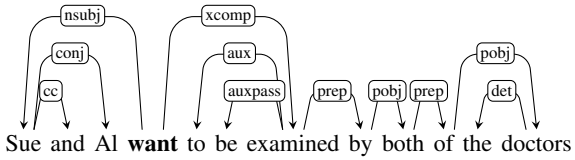
Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast,

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.

Appendix A: Example Parses

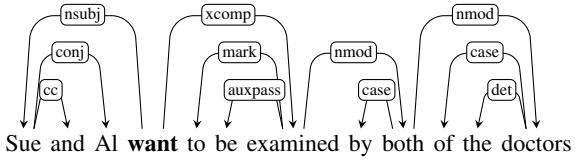
Stanford Dependencies

The following dependency tree shows the sentence *Sue and Al want to be examined by both of the doctors* in the Stanford Dependencies representation. The root of the tree, *want*, is highlighted in bold.



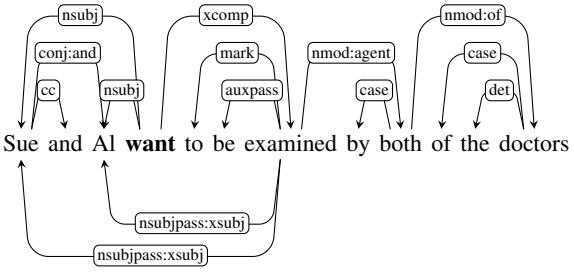
UD v1 basic

The parse of this sentence in *basic UD* is very similar to the one in *SD*. The main difference is the treatment of prepositional phrases: In *UD*, the head of the PP is the prepositional complement whereas in *SD*, it is the preposition.



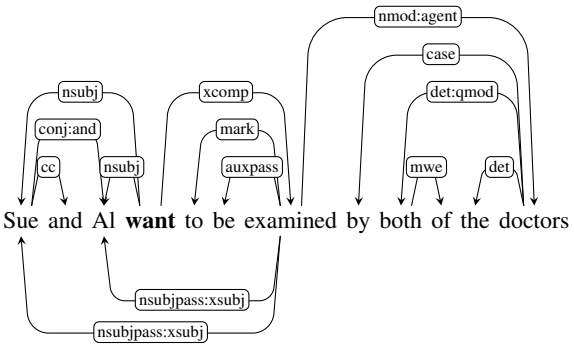
UD v1 enhanced

The parse of this sentence in *enhanced UD* contains two additional edges between the controlled verb *want* and the two controllers (*Sue* and *Al*). Further, it contains an additional edge to the second subject of *want* (*Al*). Lastly, the relation labels of the nominal modifiers (`nmod`) and conjuncts (`conj`) contain the respective function word.



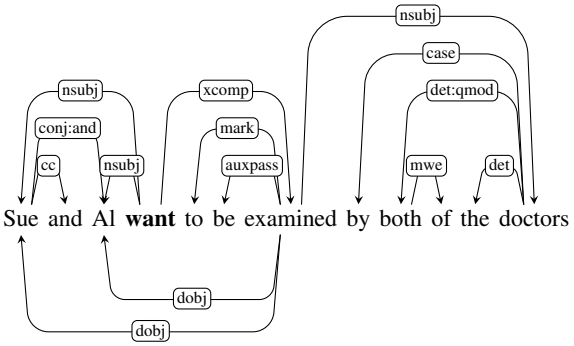
UD v1 enhanced++

This parse is identical to the parse in *enhanced UD* with the exception that it treats the phrase *both of* differently. As *both of* semantically acts as a quantificational determiner, it attaches to *doctors* and *doctors* directly attaches to the verb *examined*.



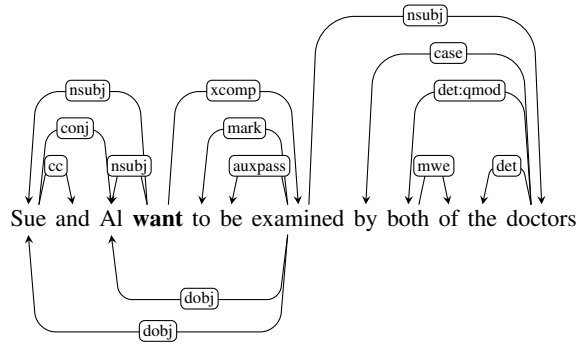
UD v1 diathesis

This parse differs from the one in *enhanced++ UD* in the treatment of the passive constructions. The passive-specific labels are replaced by `nsubj` and `dobj`.



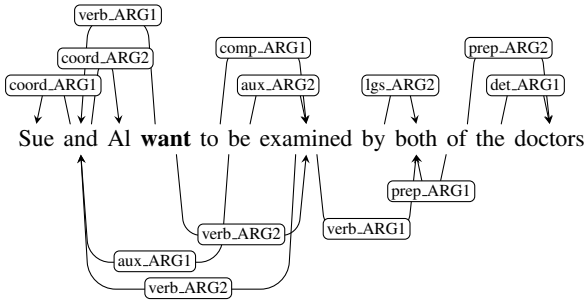
UD v1 diathesis--

This parse is identical to the previous one except for the simplified `conj` label, which no longer contains the function word.



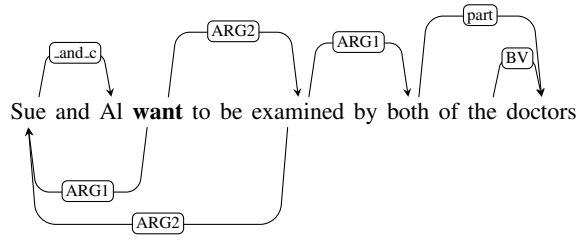
PAS

The parse of this sentence in *PAS* differs in several regards from the *UD* and *SD* parses. First, coordination is analyzed differently with the conjunction being the head of the coordinated phrase. Second, the auxiliary takes both the subject as well as the main verb as an argument. Third, the determiner *the* takes the noun *doctors* as an argument. However, *doctors* is still the argument of the preposition *of*. Lastly, the preposition *of* takes two arguments, the verb that it is modifying as well as the prepositional complement. Note that compared to the enhanced variants of *UD*, this parse contains more relations between verbs and arguments but it does not contain a direct relation between the verb and the two conjuncts *Sue* and *Al*.



DM

The parse in *DM* contains considerably fewer dependencies as it primarily encodes the relations between content words. It also neutralizes the passive alternation in a similar manner as the *PAS* and the *UD diathesis* representations.



Peking at EPE 2017: A Comparison of Tree Approximation, Transition-based and Maximum Subgraph Models for Semantic Dependency Analysis

Yufei Chen, Junjie Cao, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{yufei.chen,junjie.cao,ws,wanxiaojun}@pku.edu.cn

Abstract

This paper is a description of our system for EPE 2017: Extrinsic Parser Evaluation. We are concerned with three different models, namely tree approximation, transition-based and Maximum Subgraph models for semantic dependency parsing. We present a preliminary comparison of these models by showing some intrinsic and extrinsic evaluation results.

1 Introduction

Semantic dependency parsing (SDP) has been addressed in the literature recently (Oepen et al., 2014, 2015; Du et al., 2015a; Zhang et al., 2016; Cao et al., 2017a). SDP provides a lightweight and effective way to encode rich semantic information of natural language sentences. Existing approaches to data-driven parsing for semantic dependency structures can be categorized into four classes: tree approximation (Du et al., 2014, 2015b; Željko Agić et al., 2015), transition-based (Zhang et al., 2016), factorization-based (Martins and Almeida, 2014; Du et al., 2015a) and Maximum Subgraph (Kuhlmann and Jonsson, 2015; Cao et al., 2017a) approaches.

Different from traditional intrinsic evaluation method, the First Shared Task on Extrinsic Parser Evaluation (EPE2017; Oepen et al., 2017) introduces downstream systems which depend on grammatical analysis to evaluate parser performance.

For this task, we present a preliminary comparison of tree approximation, transition-based and Maximum Subgraph models by showing some intrinsic and extrinsic evaluation results.

2 Three Models

2.1 Tree Approximation

Previous work shows that a traditional dependency tree parser can be combined with graph transformation techniques to produce more general graph structures (Schluter and Van Genabith, 2009; Cetinoglu et al., 2010; Schluter et al., 2014; Du et al., 2014, 2015b; Željko Agić et al., 2015). This approach is referred to as Tree Approximation by some authors. During the SemEval SDP 2014, 2015 shared tasks (Oepen et al., 2014, 2015), we showed that Tree Approximation models can produce reasonably good semantic analysis (Du et al., 2014, 2015b). In addition, Tree Approximation is very useful to enhance other types of parsing techniques via feature engineering (Zhang et al., 2016) or model integration (Du et al., 2015a).

In this paper, we evaluate a tree approximation model based on the weighted graph-to-tree conversion introduced in our previous work (Du et al., 2014; Zhang et al., 2016).

2.2 Transition-based Parsing

Transition-based approach is a simple yet effective approach for syntactic parsing (Yamada and Matsumoto, 2003; Nivre et al., 2004; Chen and Manning, 2014). It has been extended to build general directed graphs (Sagae and Tsujii, 2008; Henderson et al., 2013; Sun et al., 2014; Zhang et al., 2016). Especially, different transition systems have been developed for generating more flexible graphs, e.g. DAG or even arbitrary graphs. For SDP, our previous work showed that the transition-based approach can produce semantic graphs with comparable accuracy to other more complicated approaches, e.g. grammar-based or factorization-based approahces (Zhang et al., 2016).

In this paper, we evaluate a two-stack based

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 60–64,
Pisa, Italy, September 20, 2017.

model introduced in our previous work (Zhang et al., 2016).

2.3 Maximum Subgraph Parsing

Parsing for deep dependency representations can be viewed as the search for Maximum Subgraphs for a certain graph class $\mathcal{G}$ (Kuhlmann and Jonsson, 2015). This is a natural extension of the Maximum Spanning Tree (MST) perspective (McDonald et al., 2005) for dependency tree parsing. Previous work showed that this perspective is not only elegant in theory but also effective in practice (Kuhlmann and Jonsson, 2015; Cao et al., 2017a). Our previous work showed that 1-endpoint-crossing, pagenumber-2 graphs are an appropriate graph class for modeling semantic dependency structures. On the one hand, it is highly expressive to cover a majority of semantic analysis. On the other hand, the corresponding Maximum Subgraph problem with an arc-factored disambiguation model can be solved in low-degree polynomial time (Cao et al., 2017a,b).

In this paper, we evaluate the 1-endpoint-crossing, C-free parsing model introduced in our previous work (Cao et al., 2017a).

3 Downstream Systems

The EPE2017 Shared Task uses three applications which depend strongly on syntactic or semantic analysis of the sentence. a) The biological event extraction system (Johansson, 2017) originated from the BioNLP 2009 Shared Task (Kim et al., 2009) focus on the extraction of bio-events, including the event type and theme, particularly on proteins or genes. b) The negation scope resolution system (Lapponi and Oepen, 2017) originated from the 2012 *SEM Shared Task (Morante and Blanco, 2012) is aimed at detecting the scope and focus from negation phenomenon in a sentence. c) The fine-grained opinion analysis system (Johansson, 2017) based on the MPQA Opinion Corpus (Wiebe et al., 2005) focus on marking up opinion expressions, finding opinion holders, and determining the polarities of opinion expressions.

4 Experiments

4.1 Training Data

We use Minimal Recursion Semantics (Copestake et al., 2005) derived Semantic Dependencies (DM; Ivanova et al., 2012), whose annotations are based on the parsing results given a large-scale linguistically-precise grammar and manually disambiguated, and Combinatory Categorial Grammar (Steedman, 1996, 2000) derived Deep Dependencies (CCD; Clark et al., 2002), which represents Functor-Argument Structures that are based on type-logical semantics, for training and intrinsic evaluation. Figure 1 demonstrates the two representations. The data is provided by Linguistic Data Consortium[1] (Oepen et al., 2016).

4.2 Tree Parsing Models

A tree approximation model need to be coupled with a tree parser. For EPE 2017, we use the first-order graph-based projective parser introduced in (Kiperwasser and Goldberg, 2016). The data is transformed into projective approximation trees, and then fed into a neural tree parser. The outputs of this tree parser can be directly converted to the corresponding graphs, and can also assist the transition-based and Maximum Subgraph parsers as features. Different from tree parsing, the transition-based and Maximum Subgraph parsers apply linear models for disambiguation.

4.3 Intrinsic Evaluation

Table 4.3 presents the intrinsic evaluation using the standard test data. We can see that when equipped with state-of-the-art neural parsing techniques, Tree Approximation is simple yet effective. The Maximum Subgraph model, which is comparable to graph-based tree parsing models, is slightly better than the transition-based model for both DM and CCD representations. We think the transition-based and Maximum Subgraph models can benefit from neural disambiguation models.

4.4 Extrinsic Evaluation

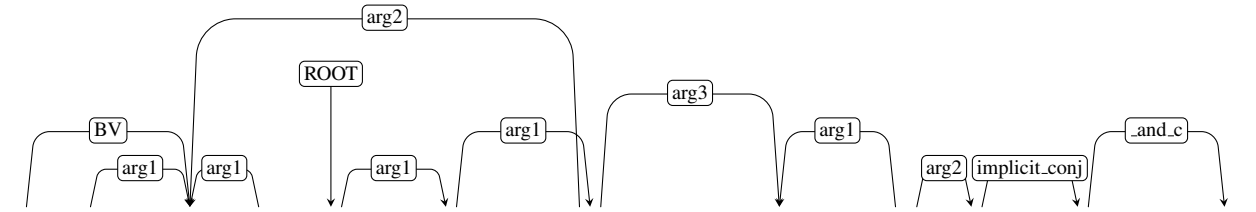
EPE 2017 includes three successive tasks, namely Event Extraction, Negation Resolution and Opinion Analysis. The extrinsic evaluation results on the development set and the evaluation set are summarized in Table 4.4 and 4.4[2].

5 Conclusion

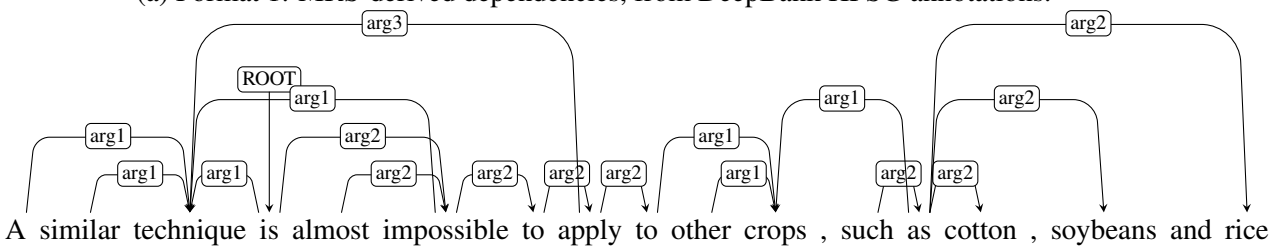
We present a preliminary evaluation of three approaches, namely tree approximation, transition-based and Maximum Subgraph approaches, to

[1] https://catalog.ldc.upenn.edu/LDC2016T10

[2] Due to our mistakes, the evaluation result of transition-based parser is missing.



(a) Format 1: MRS-derived dependencies, from DeepBank HPSG annotations.



(b) Format 2: Functor-argument structures, from CCGBank.

Figure 1: Dependency representations in (a) DeepBank (DM) and (b) CCGBank (CCD) formats.

	DM					
	UP	UR	UF	LP	LR	LF
Tree Approximation	91.21	88.88	90.03	89.74	87.45	88.58
Transition-based	90.10	89.90	90.00	87.92	87.73	87.82
Maximum Subgraph	92.77	88.17	90.41	91.45	86.91	89.12
	CCD					
	UP	UR	UF	LP	LR	LF
Tree Approximation	94.41	91.30	92.83	91.22	88.20	89.68
Transition-based	93.31	93.58	93.44	89.83	90.10	89.96
Maximum Subgraph	95.17	92.54	93.84	91.96	89.42	90.67

Table 1: Accuracy of different graph parsing models on the test data sets.

SDP. We show both intrinsic and extrinsic evaluation results. We hope to extend the empirical study in the future.

References

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017a. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 2110–2120. http://aclweb.org/anthology/P17-1193.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017b. Quasi-second-order parsing for 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of EMNLP 2017*. Association for Computational Linguistics.

Ozlem Cetinoglu, Jennifer Foster, Joakim Nivre, Deirdre Hogan, Aoife Cahill, and Josef van Genabith. 2010. Lfg without c-structures.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*. pages 327–334. http://www.aclweb.org/anthology/P02-1042.pdf.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* pages 281–332.

Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015a.

	Event Extraction			Negation Resolution			Opinion Analysis		
	P	R	F	P	R	F	P	R	F
	Tree Approximation								
DM	52.81	47.79	50.17	100	47.4	64.31	66.44	54.43	59.84
CCD	55.39	48.91	51.95	100	42.2	59.35	66.86	54.87	60.27
	Transition Based								
DM	55.06	48.69	51.68	100	42.2	59.35	66.03	54.23	59.55
CCD	55.78	47.79	51.48	100	43.93	61.04	66.9	53.85	59.67
	Maximum Subgraph								
DM	54.46	41.31	46.98	100	46.82	63.78	66.02	54.07	59.45
CCD	56.15	45.72	50.40	100	42.2	59.35	66.21	54.05	59.52

Table 2: Partial evaluation result on the development set of EPE2017.

	Event Extraction			Negation Resolution			Opinion Analysis			Average
	P	R	F	P	R	F	P	R	F	
	Tree Approximation									
DM	55.42	40.95	47.10	99.10	41.67	58.67	65.74	53.66	59.09	54.95
CCD	54.73	42.17	47.64	99.12	42.42	59.41	66.97	54.84	60.30	55.78
	Maximum Subgraph									
DM	59.28	34.22	43.39	99.15	43.94	60.89	65.63	53.64	59.03	54.44
CCD	58.26	40.07	47.48	99.15	44.32	61.26	66.57	54.55	59.96	56.23

Table 3: Partial evaluation result of EPE2017.

A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1545–1555. http://www.aclweb.org/anthology/P15-1149.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 459–464. http://www.aclweb.org/anthology/S14-2080.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015b. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 927–931. http://www.aclweb.org/anthology/S15-2154.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics* 39(4):949–998.

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, pages 2–11.

Richard Johansson. 2017. EPE 2017: The Trento–Gothenburg opinion extraction system.

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*. Association for Computational Linguistics, pages 1–9.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570.

Emanuele Lapponi and Stephan Oepen. 2017. EPE 2017: The Sherlock negation resolution downstream application.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second or-

der features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 471–476. http://www.aclweb.org/anthology/S14-2082.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 523–530.

Roser Morante and Eduardo 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-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 265–274.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In Hwee Tou Ng and Ellen Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 49–56.

Stephan Oepen, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter, Erik Velldal, and Lilja Øvrelid. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017).

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Semantic Dependency Parsing (SDP) graph banks release 1.0 LDC2016T10. Web Download.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*. Coling 2008 Organizing Committee, Manchester, UK, pages 753–760. http://www.aclweb.org/anthology/C08-1095.

Natalie Schluter, Anders Søgaard, Jakob Elming, Dirk Hovy, Barbara Plank, Hector Martinez Alonso, Anders Johannsen, and Sigrid Klerke. 2014. Copenhagen-malmö: Tree approximations of semantic parsing problems. In *SemEval@ COLING*. pages 213–217.

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.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monographs. Mit Press. http://books.google.ca/books?id=Mh1vQgAACAAJ.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding, and Xiaojun Wan. 2014. Grammatical relations in Chinese: GB-ground extraction and data-driven parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 446–456. http://www.aclweb.org/anthology/P14-1042.

Željko Agić, Alexander Koller, and Stephan Oepen. 2015. Semantic dependency graph parsing using tree approximations. In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*. London.

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

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*. pages 195–206.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.

Prague at EPE 2017: The UDPipe System

Milan Straka and **Jana Straková** and **Jan Hajič**
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
{straka,strakova,hajic}@ufal.mff.cuni.cz

Abstract

We present our contribution to The First Shared Task on Extrinsic Parser Evaluation (EPE 2017). Our participant system, the UDPipe, is an open-source pipeline performing tokenization, morphological analysis, part-of-speech tagging, lemmatization and dependency parsing. It is trained in a language agnostic manner for 50 languages of the UD version 2. With a relatively limited amount of training data (200k tokens of English UD) and without any English specific tuning, the system achieves overall score 56.05, placing as the 7^{th} participant system.

1 Introduction

Language syntax has been a topic of interest and research for hundreds of years. Syntactical analysis, most commonly in form of constituency or dependency trees, has therefore been one of the long-standing goals of computational linguistics.

Syntactic analysis was considered crucial to understand the semantics of a message. Lately, statistical and especially neutral network models have achieved superb results in natural language processing without explicit syntax recognition, by considering sentences to be merely a sequence of words. However, quite recently, syntactic trees have been shown to improve performance compared to the sequential models, especially in tasks requiring deeper understanding of text, like text summarization (Kong et al., 2017) or textual entailment (Hashimoto et al., 2016).

Consequently, syntactic parsing has its merit both as a standalone application and as a preprocessing step for further language processing, resulting in two kinds of evaluation methods – either intrinsic or extrinsic. While the intrinsic evaluation is straightforward and commonly used, extrinsic evaluation is much more complex, and to our best knowledge, there had been no standardized set of tasks serving as extrinsic evaluation.

Recently, performance of raw text parsing has been evaluated in the *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies* (Zeman et al., 2017),[1] providing a rich intrinsic evaluation of 33 systems across 81 treebanks in 49 languages of the latest version of UD, the Universal Dependencies project (Nivre et al., 2016), which seeks to develop cross-linguistically consistent treebank annotation of morphology and syntax.

The First Shared Task on Extrinsic Parser Evaluation (EPE 2017) (Oepen et al., 2017)[2] proposes an extrinsic parser evaluation metric, by means of three downstream applications that are known to depend heavily on syntactic analysis. All tasks, the *biological event extraction* (Björne et al., 2017), *negation scope resolution* (Lapponi et al., 2017) and *fine-grained opinion analysis* (Johansson, 2017), require pre-processing of raw English texts into the EPE interchange format, which is a general format encoding arbitrary dependency graphs. Each such graph represents a sentence and consists of several nodes, which correspond to substrings of the original document and include POS tags, lemmas and arbitrary morphological features. The aforementioned tasks then process these graphs and compute individual evaluation metrics, whose unweighted combination is the final EPE score.

This paper describes performance of the UDPipe system in the EPE 2017 shared task. UDPipe (Straka and Straková, 2017)[3] is an open-source

[1] http://ufal.mff.cuni.cz/conll-2017-shared-task
[2] http://epe.nlpl.eu
[3] http://ufal.mff.cuni.cz/udpipe

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 65–74,
Pisa, Italy, September 20, 2017.

tool which automatically performs sentence segmentation, tokenization, POS tagging, lemmatization and dependency trees, using UD version 2 treebanks as training data. This system has been used both as a baseline system and also a participant system in CoNLL 2017 shared task, ranking 8^{th} in the official (intrinsic) evaluation (Straka and Straková, 2017).

Even if the EPE 2017 shared task is only in English, the submitted UDPipe system is trained in a strictly language-agnostic manner without any specific handling of English, using UD 2.0 training data only. It is therefore interesting to compare it to other English-tailored participating systems.

In Section 2, we briefly discuss related work. The UDPipe system including chosen hyperparameters for English is described in Section 3. The extrinsic evaluation results of UDPipe are presented in Section 4, together with intrinsic metrics and discussion of observed performance. Finally, we conclude in Section 5.

2 Related Work

Deep neural networks have achieved remarkable results in many areas of machine learning. In NLP, end-to-end approaches were initially explored by Collobert et al. (2011). With a practical method for precomputing word embeddings (Mikolov et al., 2013) and utilization of recurrent neural networks (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) and sequence-to-sequence architecture (Sutskever et al., 2014; Cho et al., 2014), deep neural networks achieved state-of-the-art results in many NLP areas like POS tagging (Ling et al., 2015), named entity recognition (Yang et al., 2016) or machine translation (Vaswani et al., 2017).

The wave of neural network parsers was started recently by Chen and Manning (2014), who presented a fast and accurate transition-based parser. The proposed neural network architecture is simple, consisting of only an input layer, one hidden layer and an output softmax layer, without any recurrent connections. The UDPipe parser (Straka and Straková, 2017) is based on this architecture, and adds partially non-projective and fully non-projective transition systems, as well as a search-based oracle (Straka et al., 2015).

Many other parser models followed, employing various techniques like stack LSTM (Dyer et al., 2015), global normalization (Andor et al., 2016), contextualization of embeddings using bidirectional LSTM (Kiperwasser and Goldberg, 2016), biaffine attention (Dozat and Manning, 2017) or recurrent neural network grammars (Kuncoro et al., 2016), improving LAS score in English and Chinese dependency parsing by more than 2 points in 2016.

2.1 Motivation For Structured Sentence Processing

Although the sequence-to-sequence architecture provides overwhelming performance compared to traditional methods, there have been several attempts to enhance it utilizing syntactic information. Many such designs employ dependency trees not merely as features, but either to encode an input sentence according to syntactic tree (Tai et al., 2015; Li et al., 2017; Chen et al., 2017) or generate an output sentence as a dependency tree (Wu et al., 2017; Rabinovich et al., 2017).

A comprehensive comparison of processing input sentence either as a sequence or as a tree was performed by Yogatama et al. (2016). Additionally, the authors also considered the eventuality of utilizing task-specific syntax trees, both in semi-supervised manner (i.e., bootstrapping the task-specific syntax with manually annotated trees and allowing their change later) and in unsupervised manner. While the supervised syntax did not demonstrate much improvement, both semi-supervised and unsupervised approach (i.e., learning task-specific syntax) yielded substantial gains in all four examined tasks.

3 UDPipe

UDPipe[4] is an open-source pipeline which performs tokenization, morphological analysis, part-of-speech tagging, lemmatization and dependency parsing. It is a simple-to-use tool consisting of one binary and one model (per language) and can be easily trained using solely data in CoNLL-U format, without additional linguistic knowledge on the users' part. Precompiled binaries for Windows, Linux and OS X are available, as are bindings for Python,[5] Perl,[6] Java and C#. Source code is available on GitHub[7] under MPL license.

The initial UDPipe 1.0 release (Straka et al., 2016) processed CoNLL-U v1 files and was dis-

[4] `http://ufal.mff.cuni.cz/udpipe`
[5] PyPI package `ufal.udpipe`
[6] CPAN package `UFAL::UDPipe`
[7] `http://github.com/ufal/udpipe`

tributed with 36 pretrained models[8] on UD 1.2 data. Updated UDPipe 1.1 and UDPipe 1.2 versions (Straka and Straková, 2017) process CoNLL-U v2 files and were employed in CoNLL 2017 shared task, with UDPipe 1.1 serving as a baseline system and UDPipe 1.2 attending as a participant system. Recently, pretrained models for 50 languages were released based on UD 2.0 data.[9]

All UDPipe models, especially the ones participating in the CoNLL 2017 shared task, have been evaluated using several intrinsic metrics (Zeman et al., 2017). Therefore, we employed these exact models to participate in EPE 2017, in order to facilitate comparison between the extrinsic and intrinsic measurements.

We now briefly describe the most recent version, UDPipe 1.2, together with the chosen hyperparameters for the English model (according to performance on the development set). More detailed language-independent description of the system and its gradual updates are available in Straka and Straková (2017) and in Straka et al. (2016).

3.1 Tokenizer

The tokenizer performing sentence segmentation and tokenization is trained purely with the UD training data. The CoNLL-U format allows for the reconstruction of the original pre-tokenized text using the `SpaceAfter=No` feature, which indicates that a given token was not followed by a space separator in the original text. This facilitates training a model which predicts the probability of a token break after every character in a given plain text.

Sentence breaks can be trained analogously. However, the CoNLL-U v1 format does not provide markup for paragraph and document boundaries. These are often indicated by visual layout and/or spacing, but if not annotated in the data, a sentence segmenter has to predict the sentence boundary at the end of a paragraph only from the raw text. Considering the examples from the English UD data presented in Figure 1, such sentence breaks confuse the segmenter and prompt it to split sentences more often than necessary.

The CoNLL-U v2 format has been updated to include markup for paragraph and document boundaries. Unfortunately, only document boundaries are marked in English UD 2.0 data, resulting in the segmenter still being trained on sentence boundaries marked in Figure 1.

Keep in touch, / Mike / Michael J. McDermott

i have two options / using the metro or the air france bus / can anybody tell me if the metro runs directly ...

Figure 1: Examples of sentence breaks (denoted with slash) in English UD data which are hard to predict without inter-sentence spacing and layout.

Technically, the UDPipe tokenizer predicts for each character whether it is followed by a token break, sentence break or none of above. Each character is represented using randomly initialized embedding of dimension d and a bidirectional GRU (Cho et al., 2014) network is employed during the prediction. The details of the architecture, training and inference are explained in Straka et al. (2016).

For English, embedding and GRU dimension are set to 64. The network is trained using dropout rate of 10% before and after the GRU cells for 100 epochs, each consisting of 200 batches containing 50 segments of 50 characters. The network weights are updated using Adam (Kingma and Ba, 2014) with initial learning rate of 0.002. Additionally, space is assumed to always separate tokens (due to no training token containing a space) and the network is therefore trained to only predict token breaks which do not precede a space character.

3.2 Tagger

Part-of-speech tagging is performed in two steps: firstly, a set of candidate *(UPOS, XPOS, FEATS)* triples are generated for each token using its suffix of length at most 4, and secondly, these candidates are disambiguated on a sentence-level using averaged perceptron (Collins, 2002) with Viterbi decoding of order 3.

To facilitate the candidate generation, a guesser dictionary with a predefined number of most common candidate triples for every possible suffix of length 4 is constructed according to the training data. When searching the guesser dictionary, entires with longer suffix match are always preferred. Additionally, for every token in the training data, all its appearing *(UPOS, XPOS, FEATS)* analyses are kept.

In order to generate candidates for a given token, two cases are considered. If the token was

[8] `http://hdl.handle.net/11234/1-1659`

[9] `http://hdl.handle.net/11234/1-2364`

present in the training data, all its analyses appearing in the training data are returned, together with 6 another (differing) most common candidates from the guesser dictionary. If the tokes was not present in the training data, 10 most common candidates from the guesser dictionary are generated.

The candidates are disambiguated using averaged perceptron utilizing a predefined rich set of feature templates based on classification features developed by Spoustová et al. (2009) for Czech.

A lemmatizer is nearly identical to the above described part-of-speech tagger. For every token, the candidates are *(UPOS, lemma rule)* pairs, where the *lemma rule* is the shortest formula for generating a lemma from a given token, using any combination of "remove a specific prefix", "remove a specific suffix", "append a prefix" and "append a suffix" operations. For the English lemmatizer, at most 4 candidates are generated for every token, because of the smaller number of lemmas (compared to XPOS and morphological features).

Theoretically, both the part-of-speech tagging and lemmatizing could be performed jointly using candidate quadruples, but such approach results in lower performance (we hypothesise that the required number of candidate quadruples is too high for the disambiguation step to be performed effectively).

3.3 Dependency Parser

UDPipe utilizes fast transition-based neural dependency parser inspired by Chen and Manning (2014). The parser is based on a simple neural network with just one hidden layer and without any recurrent connections, using locally-normalized scores.

The parser offers several transition systems, from projective and partially non-projective to fully non-projective. For English, a projective arc-standard system (Nivre, 2008) with both a dynamic oracle (Goldberg et al., 2014) and a search-based oracle (Straka et al., 2015) yield the best performance. It is possible to combine both oracles, because the search-based oracle employs an arbitrary transition-based parser – even the one utilizing a dynamic oracle.

Even if a projective transition system is used, non-projective trees are used during training, with the parser trying to predict a (projective) subset of dependency edges.

The parser employs FORM, UPOS, FEATS and DEPREL embeddings. The form embeddings of dimension 64 are precomputed with `word2vec` on the training data only, with the following options:

```
word2vec -cbow 0 -size 64 -window 10 -negative 5
  -hs 0 -sample 1e-1 -iter 15 -min-count 2
```

The precomputed embeddings are used only for forms occurring at least twice in the training data; forms appearing only once are considered unknown forms and used to train the (initially random) embedding of unknown words. The UPOS, FEATS and DEPREL embeddings have dimension 20 and are initialized randomly. All embeddings are updated during training.

The size of the hidden layer is 200. The network is trained using SGD with minibatches of size 10, starting with learning rate 0.01 and gradually decaying it to the final 0.001. L2 regularization with weight 1.5e-6 is applied to reduce overfitting.

3.4 Training UDPipe

UDPipe is trained without any language specific knowledge. Even if we have so far described specific hyperparameter values used by the English models, the hyperparameters for each treebank are extensively tuned on the development set.

The UD 2.0 data contain three English treebanks. Consequently, in addition to training treebank-specific models, we also experiment with training a model using a union of all these tree treebanks. Even if the treebanks use different XPOS tags and there are annotation inconsistencies among the treebanks (which are observable using the intrinsic evaluation of the merged model on the individual treebanks' test sets), we hypothesise that the larger training data should benefit real-word applications.

3.5 Example Parse Tree

To illustrate the UD-style trees with universal dependency relations and universal POS tags, we provide an example of a (correctly parsed) tree in Figure 2.

4 Experiments and Results

We submitted five different UDPipe configurations to the EPE 2017 shared tasks. These runs are described in Table 1. The run 0 is the English treebank model of UDPipe 1.2 from the CoNLL 2017 shared task (Zeman et al., 2017). The exactly same model is used as a run 1, but using the tok-

Run name	Run	Description	Tokens
UD2.0 En/UDPipe/20	0	UDPipe 1.2, UD 2.0 English data, UDPipe tokenizer, beam size 20	204.5k
UD2.0 En/EPE/20	1	UDPipe 1.2, UD 2.0 English data, EPE provided tokenizer, beam size 20	204.5k
UD2.0 EnMerged/UDPipe/20	2	UDPipe 1.2, UD 2.0 English + English LinES + English ParTUT data, UDPipe tokenizer, beam size 20	292.2k
UD2.0 EnMinus/UDPipe/5	3	UDPipe 1.1, first 95% of UD 2.0 English data, UDPipe tokenizer, beam size 5	192.5k
UD1.2 En/UDPipe/5	4	UDPipe 1.0, UD 1.2 English data, UDPipe tokenizer, beam size 5	204.5k
Stanford-Paris	*6*	*UD v1 enhanced dependencies, WSJ+Brown+GENIA data*	*1692.0k*

Table 1: Description of employed systems

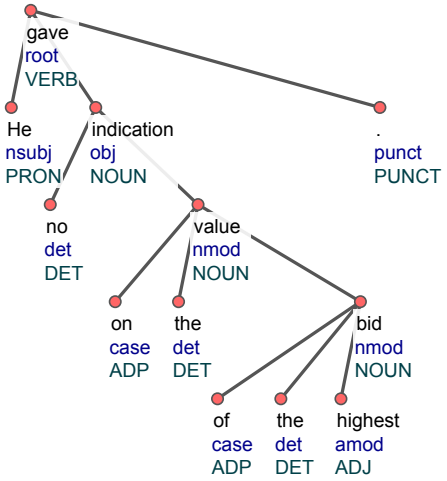


Figure 2: An example parse tree from UD 2.0 English data. For every word, its form, dependency relation and universal POS tag is displayed.

enization and segmentation of the data provided by the EPE 2017 organizers. The consequent run is the CoNLL 2017 shared task model trained on all three English treebanks. The last two runs are the English treebank models of UDPipe 1.1 and UDPipe 1.0. The first four runs are based on UD 2.0, and only the last run utilizes UD 1.2 data. Note that the run 3 uses only a subset of the training data, because the models were released for the CoNLL 2017 shared task before the release of the test data, using official development data for evaluation. For comparison, we additionally include the overall best participant system of the shared task.

The overall results of UDPipe in the EPE 2017 shared task are presented in Table 2. According to the overall score, UDPipe placed 7$^{\text{th}}$ out of 8 participants of the shared task, by a large margin compared to the best participating system. The performance on the three individual tasks are detailed in Tables 3, 4 and 5.

To enable interpretation of the results, we also provide the intrinsic evaluation of the employed models on the UD test sets in Table 6.

Overall Results

With the overall score of 56.05, UDPipe lacks behind nearly all other participant systems. The overall scores of the systems ranking immediately above UDPipe are 56.23, 56.24, 56.65, 56.81 and 58.57, with the best system achieving a respectable score of 60.51. The best overall UDPipe score is achieved by the English-only CoNLL 2017 UDPipe 1.2 model with EPE-provided tokenization.

One of the probable cause of our lower performance is the size of the training data – while the UD 2.0 data offer training data of 200k tokens (290k if all three English treebanks are merged), most other participants use Wall Street Journal corpus (Marcus et al., 1993) with 800k tokens, sometimes also together with Brown corpus (Francis and Kucera, 1979) an GENIA corpus (Ohta et al., 2002), resulting in circa 1700k tokens.

Furthermore, we emphasize that even though the EPE 2017 shared task focused on English language only, UDPipe is trained in a language agnostic manner for 50 languages without any adaptation for English other than setting up the hyperparameters of the artificial neural networks.

Tokenization Issues

The overall results in Table 2 indicate that the UDPipe tokenization is of lower quality – using the EPE-provided tokenizer improves the overall score by 2 points. By contrast, the evaluation on the UD 2.0 data (the Words and Sentences columns of Table 6) show opposite results, with the EPE-provided tokenizer substantially degrading performance on UD 2.0 test sets.

We therefore hypothesise that the lower extrinsic performance of UDPipe tokenization is a consequence of the tokenization and sentence segmentation annotated in the UD data. We argue that to improve the annotation, one possible course of action is to indicate paragraph boundaries in English UD 2.0 data, which might improve the performance of trained sentence segmenter.

UDPipe run	Event extraction	Negation resolution	Opinion analysis	Overall score
0-UD2.0 En/UDPipe/20	43.58	58.83	59.79	54.07
1-UD2.0 En/EPE/20	45.54	61.62	61.00	56.05
2-UD2.0 EnMerged/UDPipe/20	44.25	59.95	58.71	54.30
3-UD2.0 EnMinus/UDPipe/5	42.70	59.95	58.90	53.85
4-UD1.2 En/UDPipe/5	43.22	50.85	58.53	50.86
Stanford-Paris, run 6	*50.23*	*66.16*	*65.14*	*60.51*

Table 2: Overall EPE evaluation results

UDPipe run	Approximate span & recursive mode		
	Precision	Recall	F1-score
0-UD2.0 En/UDPipe/20	53.84	36.61	43.58
1-UD2.0 En/EPE/20	56.35	38.21	45.54
2-UD2.0 EnMerged/UDPipe/20	53.22	37.87	44.25
3-UD2.0 EnMinus/UDPipe/5	51.91	36.27	42.70
4-UD1.2 En/UDPipe/5	51.71	37.12	43.22
Stanford-Paris, run 6	*58.36*	*44.09*	*50.23*

Table 3: Event extraction evaluation results

UDPipe run	Scope match			Scope tokens			Event match			Full negation		
	P	R	F	P	R	F	P	R	F	P	R	F
0-UD2.0 En/UDPipe/20	99.39	65.73	79.13	90.12	86.76	88.41	66.23	61.82	63.95	99.10	41.83	58.83
1-UD2.0 En/EPE/20	98.77	64.26	77.86	88.58	87.75	88.16	70.44	66.67	68.50	99.16	44.70	61.62
2-UD2.0 EnMerged/UDPipe/20	99.40	67.34	80.29	91.45	87.49	89.43	65.81	61.82	63.75	99.12	42.97	59.95
3-UD2.0 EnMinus/UDPipe/5	99.38	64.92	78.54	90.84	85.48	88.08	66.46	63.25	64.82	99.12	42.97	59.95
4-UD1.2 En/UDPipe/5	97.81	54.03	69.61	90.40	83.36	86.74	62.11	59.88	60.97	98.90	34.22	50.85
Stanford-Paris, run 6	99.44	*70.68*	*82.63*	*93.06*	*85.48*	*89.11*	*72.33*	*68.45*	*70.34*	*99.24*	*49.62*	*66.16*

Table 4: Negation resolution evaluation results

UDPipe run	Expressions			Holders			Polarity			Holders (in vitro)		
	P	R	F	P	R	F	P	R	F	P	R	F
0-UD2.0 En/UDPipe/20	64.32	55.07	59.33	49.03	41.71	45.08	54.44	45.36	49.49	62.61	57.21	59.79
1-UD2.0 En/EPE/20	63.57	55.15	59.06	48.81	44.31	46.46	53.68	45.93	49.50	62.31	59.74	61.00
2-UD2.0 EnMerged/UDPipe/20	64.57	54.58	59.15	50.01	39.79	44.32	54.93	45.22	49.60	63.45	54.63	58.71
3-UD2.0 EnMinus/UDPipe/5	64.15	54.43	58.89	48.20	41.25	44.46	54.30	44.82	49.11	61.26	56.72	58.90
4-UD1.2 En/UDPipe/5	63.98	54.46	58.84	48.38	40.99	44.38	53.99	44.50	48.79	61.00	56.25	58.53
Stanford-Paris, run 6	*63.90*	*56.07*	*59.73*	*54.14*	*46.41*	*49.98*	*54.04*	*45.87*	*49.62*	*68.86*	*61.81*	*65.14*

Table 5: Opinion analysis evaluation results

Row	Data	Plain text processing						Using gold tokenization			
		Words	Sents	UPOS	XPOS	UAS	LAS	UPOS	XPOS	UAS	LAS
0-UD2.0 En/UDPipe/20	UD 2.0 En	**99.0**	**75.3**	**93.5**	**92.9**	**80.3**	**77.2**	94.4	93.8	**84.6**	**81.3**
	UD 2.0 EnMerged	**98.9**	**79.5**	91.8	—-	78.4	73.9	92.7	—-	81.4	76.6
	UD 1.2 En	**99.0**	**75.3**	87.9	**92.9**	75.7	63.7	88.8	93.8	79.1	66.8
1-UD2.0 En/EPE/20	UD 2.0 En	96.2	59.9	90.7	90.0	74.6	71.8	94.4	93.8	**84.6**	**81.3**
	UD 2.0 EnMerged	97.8	71.0	90.6	—-	75.8	71.4	92.7	—-	81.4	76.6
	UD 1.2 En	96.2	59.9	85.1	90.0	70.3	58.7	88.8	93.8	79.1	66.8
2-UD2.0 EnMerged/UDPipe/20	UD 2.0 En	**99.0**	**75.3**	93.4	92.6	79.8	76.7	94.4	93.6	84.0	80.6
	UD 2.0 EnMerged	**98.9**	**79.5**	**92.0**	—-	**79.1**	**74.9**	**92.9**	—-	**82.2**	**77.7**
	UD 1.2 En	**99.0**	**75.3**	87.8	92.6	75.6	63.4	88.7	93.6	78.9	66.3
3-UD2.0 EnMinus/UDPipe/5	UD 2.0 En	98.7	73.2	93.1	92.4	78.9	75.8	**94.5**	**93.9**	83.8	80.7
	UD 2.0 EnMerged	98.8	78.6	91.6	—-	77.7	73.1	92.8	—-	81.1	76.3
	UD 1.2 En	98.7	73.2	87.5	92.4	74.6	62.6	88.9	**93.9**	78.6	66.3
4-UD1.2 En/UDPipe/5	UD 2.0 En	98.4	72.3	87.3	92.2	73.9	62.0	88.8	93.8	78.8	66.3
	UD 2.0 EnMerged	98.7	77.8	86.5	—-	73.9	60.1	87.6	—-	77.2	63.0
	UD 1.2 En	98.4	72.3	**92.9**	92.2	**78.3**	**75.1**	**94.5**	93.8	**84.2**	**80.7**

Table 6: Intrinsic evaluation of UDPipe runs on the Universal Dependencies test data (Nivre et al., 2017).

Merged English Treebanks

Although the model trained on the three merged UD 2.0 English treebanks provide inconsistent XPOS tags and shows slight performance drop on the main English UD 2.0 treebank (cf. Table 6), the extrinsic evaluation of this model shows noticeable improvement. The improvement may be attributed both to the increased size and diversity of the training data, but also to the different annotation, which might serve as a regularization.

According to the extrinsic results, the merged model used together with the EPE-provided tokenizer should surpass the overall score of the best submitted UDPipe run.

Negation Resolution Results Drop of Run 4

The Table 2 indicates a surprising drop of performance of the run 4 (UDPipe 1.0 English model trained using UD 1.2 data) on the Negation resolution task, without a corresponding change on the two other tasks.

Note that the three EPE tasks are able to use only one kind of POS tags, i.e., either UPOS or XPOS in case of UDPipe. The decision on the type of POS tags used is performed by the EPE organizers according to the performance on the development set. For the UDPipe systems, XPOS tags are utilized overwhelmingly, with the UPOS tags being used only once – by the run 4 on the Negation resolution task. Therefore, we initially hypothesised that the drop is caused by the fact that the UPOS tags are much more coarse than the XPOS tags.

However, after evaluating the results on both XPOS and UPOS tags, we found out that XPOS tags are in fact used, and the information about used UPOS tags in the official results is incorrect.

After further investigation, we found out that the Negation resolution task texts are the only one containing non-ASCII characters (i.e., Unicode quotation marks, apostrophes and hyphens) and that the UDPipe 1.0 tokenizer does not correctly process them, resulting in poor tokenization and segmentation.

5 Conclusions and Future Work

We described the UDPipe systems used in the EPE 2017 shared task, presented the extrinsic and intrinsic evaluation of the submitted models, discussed the results and offered several hypotheses to interpret the data. For the immediate future work, we will carry out several experiments to support the hypotheses:

- When the paragraph boundaries are annotated in the UD data, does the trained sentence segmenter achieve better performance?
- Can a rule-based English tokenizer also improve the results?
- What effect would larger training data (like WSJ) have?
- What performance would a state-of-the-art dependency parser attain using the UD 2.0 data only?

Acknowledgments

This work has been partially supported and has been using language resources and tools developed, stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071). This research was also partially supported by OP VVV projects CZ.02.1.01/0.0/0.0/16_013/0001781 and CZ.02.2.69/0.0/0.0/16_018/0002373, by project No. DG16P02R019 supported by the Ministry of Culture of the Czech Republic and by SVV project number 260 453.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globaly normalized transition-based neural networks. In *Association for Computational Linguistic*. http://arxiv.org/abs/1603.06042.

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

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers). Association for Computational Linguistics, Vancouver, Canada, pages 1936–1945. http://aclweb.org/anthology/P17-1177.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR* abs/1409.1259. http://arxiv.org/abs/1409.1259.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

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

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

W. N. Francis and H. Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US. http://icame.uib.no/brown/bcm.html.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *TACL* 2:119–130. http://www.aclweb.org/anthology/Q14-1010.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* pages 5–6.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR* abs/1611.01587. http://arxiv.org/abs/1611.01587.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Lingpeng Kong, Chris Alberti, Daniel Andor, Ivan Bogatyy, and David Weiss. 2017. DRAGNN: A transition-based framework for dynamically connected neural networks. *CoRR* abs/1703.04474. http://arxiv.org/abs/1703.04474.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2016. What do recurrent neural network grammars learn about syntax? *CoRR* abs/1611.05774. http://arxiv.org/abs/1611.05774.

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

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 688–697. http://aclweb.org/anthology/P17-1064.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR* abs/1508.02096. http://arxiv.org/abs/1508.02096.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *COMPUTATIONAL LINGUISTICS* 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*

Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.. pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.* 34(4):513–553. https://doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague. http://hdl.handle.net/11234/1-1983.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher 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 (LREC 2016)*. European Language Resources Association, Portorož, Slovenia, pages 1659–1666.

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

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the Second International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, HLT '02, pages 82–86. http://dl.acm.org/citation.cfm?id=1289189.1289260.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1139–1149. http://aclweb.org/anthology/P17-1105.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-Supervised Training for the Averaged Perceptron POS Tagger. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, Athens, Greece, pages 763–771. http://www.aclweb.org/anthology/E09-1087.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Portorož, Slovenia.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Vancouver, Canada, pages 88–99. http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *CoRR* abs/1503.00075. http://arxiv.org/abs/1503.00075.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 698–707. http://aclweb.org/anthology/P17-1065.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR* abs/1603.06270. http://arxiv.org/abs/1603.06270.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *CoRR* abs/1611.09100. http://arxiv.org/abs/1611.09100.

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 Gökırmak,

Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher 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*. Association for Computational Linguistics.

Szeged at EPE 2017: First experiments in a generalized syntactic parsing framework

Zsolt Szantó, Richárd Farkas
University of Szeged, Institute of Informatics
{szantozs, rfarkas}@inf.u-szeged.hu

Abstract

In this article, we describe the submissions of Team Szeged for the EPE 2017 shared task. We introduce three approaches and first experiments to exploit the opportunities of the general dependency graph representation of the shared task.

1 Introduction

The goal of the First Shared Task on Extrinsic Parser Evaluation[1] (EPE 2017) (Oepen et al., 2017) was to estimate "the relative utility of different types of dependency representations for a variety of downstream applications that depend heavily on the analysis of grammatical structure."

To enable different types of dependency representations, the organizers of the shared task introduced a very general graph-based representation of 'relational' structure reflecting syntactic-semantic analysis. The nodes of this graph correspond to lexical units, and its edges represent labeled directed relations between two nodes. Nodes can be defined in terms of (in principle arbitrary) sub-strings of the surface form of the input sentence. This representation allows overlapping and empty (i.e. zero-span) node sub-strings as well. Moreover, nodes and edges are labeled by attribute–value maps without any restriction on the attribute set.

This very general graph-based representation opens brand new ways for expressing syntactic or semantic information besides the standard dependency tree formalism. We understood the call of the shared task in a generalized way and came up with ideas which aim to leverage the opportunities of the general representation beyond dependency parse trees. We experimented with a couple of such ideas (instead of trying to achieve high scores in the shared task) and we shall introduce them in this paper.

The contribution of this work consists of three independent sets of experiments in the EPE 2017 setting. We emphasize that these are only first experimental results and there are plenty of ways for further analyzing these approaches and also to come up with other ideas leveraging the broadly understood task specification.

In the first set of experiments (§2), we start from the classic dependency parsing approach but instead of a single dependency parse we express the distribution of possible dependency parses given a sentence in the graph-based general representation. In (§3), we introduce a possible solution for enriching the dependency parse by constituent information given by a standard phrase-structure parser. In this way, various syntactic representations can be represented in the graph and information is not lost because the downstream application can only accept a single dependency parse tree. Furthermore, in the EPE 2017 setting we can send a blended relational structure to the downstream task, like a parse distribution and blended version of different syntactic approaches, and the downstream application is able to machine learn which type of syntactic structure or phenomenon or even which combination of syntactic information is useful for itself.

Our last batch of experiments (§4) is a consequence of this objective, i.e. the relational representation has to be useful for the downstream application. Here, we tried to automatically recognize which dependency parse labels are useful to a downstream task and collapsed the useless ones.

[1] all submissions, results, and the complete evaluation infrastructure are available in http://epe.nlpl.eu

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 75–79,
Pisa, Italy, September 20, 2017.

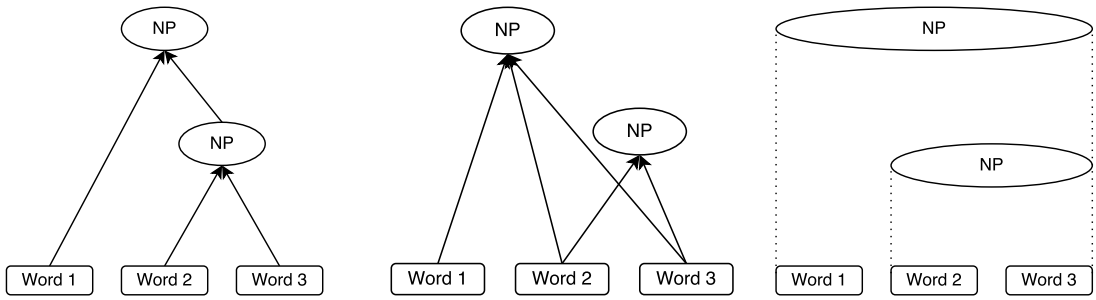


Figure 1: Alternatives for representing constituent trees in the general dependency graph format.

2 Parse Distribution as Input for Downstream Applications

Standard dependency parsers output a single dependency parse tree. Our hypothesis was that a downstream application could profit from having access to the distribution of possible parses and not just to the most likely parse tree. The distribution over possible parses estimated by the parsing model might be useful for a downstream application because it might reveal that edges or their labels are less confident or also point out relatively highly probable dependencies which are not part of the best single parse tree. The general graph-based representation of EPE 2017 enables one to express the distribution over possible edges and labels, i.e. possible parses.

Getting out the density estimation from a particular parser is usually complicated because of both theoretical and practical (software implementation) issues. Hence we decided to use an approximation of edge and label likelihoods based on top-k parses for our first experiments. Our assumption here is that the top k outputs of a parser model contains most of the useful bi-lexical dependencies and the frequency of a particular dependency counted among the k parses is a good enough approximation for its likelihood (this idea is similar to constituent-level strategy of the Berkeley product parser (Petrov, 2010)).

We added all edges from the k-best trees of a parser to the general dependency graph. We also added a new label to all edges whose value is the frequency of the same edge label pairs among the k parses. For these experiments we used the MST-Parser (McDonald et al., 2005) that we trained on Universal Dependencies v2 (Nivre et al., 2015) and we asked for the 10-best trees with default parameters.

3 Constituents in the (Bi-Lexical) Relational Representation

Constituency parsers focus on the phrases/constituents and phrase structure of the sentence, i.e. follow a non bi-lexical syntactic representation. Several applications might prefer bi-lexical representations (like the ones based on predicate-argument structures) while others might prefer constituency (like scope detection). Fortunately, the general graph representation of EPE 2017 enables us to put both the dependency and constituency parse output into a blended syntactic graph. Hence we do not have to choose between the two approaches but the downstream application can machine learn which syntactic phenomena is useful for itself or even can learn patterns in the graph consisting of information from both constituency and dependency. Several previous work has shown that the two syntactic representation and their parsers can work together efficiently cf. Farkas and Bohnet (2012). We believe that is especially true for using them jointly in downstream applications.

There are many possible ways to represent a constituent tree in the general dependency graph format. Although these representations contain the same information because of the feature extractors of the downstream applications they can have different effect in practice.

An interesting opportunity of the EPE 2017 general graph representaton is that it enables the creation of virtual nodes (which are not directly associated with a word). This feature gives the posibility to create a new node for each non-terminal in a constituent tree. Our three proposals differs in the way these virtual nodes are linked to the overt nodes in the graph.

1. In the first setup, shown by the Figure 1a, we

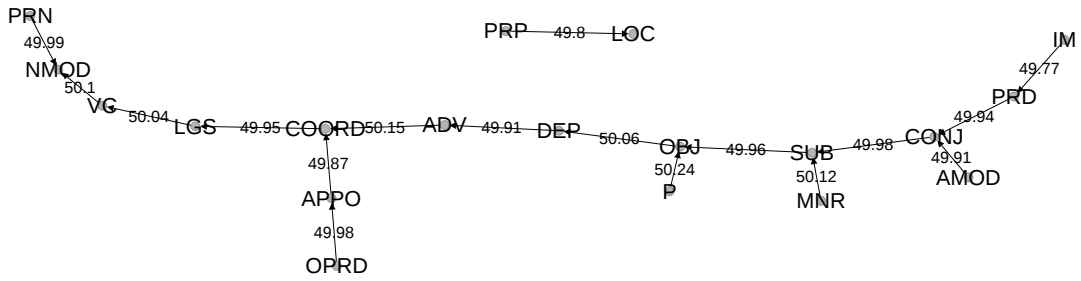


Figure 2: Label adjustment graph.

connect each of the children to their direct parents. In this way our graphs will be very similar to a constituent tree. In this example Word 2 and Word 3 are connected to an NP, that NP and Word 1 is connected to another NP.

2. Another posibility is when each of the nodes are connected to all ancestor non-terminals (Figure 1b). In that case there is a direct relation between a constituent and their descendants. In the current example the higher level NP directly contains the children (Word 2, Word 3) of its child. We hope this representation that the feature extractor of downstream applications can directly generate features about the ancestors without recursive rules

3. A different approach is where we give the covering area for each new non-terminal (Figure 1c). In these cases like in the previous we have not got direct information about the connection between the nonterminals. On the other hand, it can help for an application which uses the position of a node.

For constituent parsing we used the Berkeley Parser (Petrov et al., 2006) with default parameters and pretrained model (eng-sm6). In our submission we used the second and third methods in the dependency graph format.

4 Label Set Adjustment Driven by Downstream Applications

Different downstream applications might utilize different types of grammatical patterns. The simplest case is that a downstream application might extract important features from particular edge labels while features over other edge labels are negligible in its machine learnt model. Moreover, different applications might utilize different types of dependencies, see for example event recognition versus negation scope detection.

We propose a simple procedure to recognize edge labels which can be collapsed into other edge labels because their discrimination does not give any added value to the downstream application in question. We start from the full set of edge labels and systematically check what is the effect of collapsing two particular labels evaluated through the downstream application.

We calculated for all label pairs what is going to happen if we replace one dependency label with another. For our experiments we used the Turku Event Extraction System (TEES) (Björne et al., 2017), but because we did not have enough time to retrain the TEES system for all combinations, we trained it once with the full label set and we did the prediction part separately to each dependency label pairs. In this prediction part we replaced each of the labels with each of the another labels on the full development set. We got a complete directed graph where the nodes are the labels and edges contains the scores from the TEES system with the merged labels. For each node we kept the outcoming edge with maximum weight i.e. when the replacement was the most efficient. When there were two edges between two nodes we removed the smaller.

Figure 2 contains the graph we got. (When we ran the TEES system with default parameters we got 49.76 with original labels). By using this graph we started replacing the nodes from the highest edge weight to the lowest. We evaluated the new label set in every step and we found the best result after three steps, 50.36, which is slightly better than the best merged pair. After that we did the three replace steps in the full dataset. Because of lack of time, we could not make the replacement in the full dataset and retrain TEES before the submission deadline. We sent the trees with collapsed label set. We found lower scores

	Event Extraction	Negation Resolution	Opinion Analysis
mate - baseline	47.84	61.98	65.87
mate + label adjustment	47.37	60.53	66.33
mate + constituent	46.71	61.26	63.13
mate + mst - baseline	46.69	59.78	63.25
mate + mst - k-best	45.96	59.05	62.5

Table 1: Final result in evaluation set.

than the baseline on the TEES test set. We also could not use this method for the other downstream applications.

5 Results

Table 1 shows our official results achieved on the shared task. The *baseline - mate* is one of our baselines where we just run the mate parser (Bohnet, 2010) with a pretrained model. The second and third rows contain the result of *label adjustment* and *constituent parsing* experiments. The fourth row contains another baseline when we applied the MSTParser and the last row shows the scores of our *k-best experiment* (we used MSTParser here).

Unfortunately, the deeper analysis of results is left for future work. For example, the reason why the *combination of k-best parses* gets lower result in every task than the *baseline-MST* is maybe that feature extractors were prepared for dependency representations and not for distribution graphs.

5.1 Event Extraction

In the event extraction task we can not beat our baselines, all of our modifications – including the label adjustment which is optimized for this task – get a negative effect. The dependency label merging mechanism that we directly developed on this task also failed.

5.2 Negation Resolution

One of the main motivations of the constituent based approach was the negation resolution task (Lapponi and Oepen, 2017). The scope of the negations are usually a grammatical phrase that can we identify with constituent parsers. This *constituency-based system* got better results in three out of four scope-focused evaluation metrics than our baseline. Table 2 shows the detailed comparison of the baseline and the constituent system.

The following example shows how can the constituent parse helps:

	baseline		mate + const	
	dev	test	dev	test
Scope Match	**78.42**	80.00	77.98	**81.14**
Scope Tokens	86.64	89.17	**87.38**	**89.27**
Event Match	**75.47**	**67.90**	72.90	65.20
Full Negation	**62.15**	**61.98**	59.91	61.26

Table 2: Detailed results of the *baseline - mate* and the *mate + constituent* systems in negation resolution task.

*"I join in it because there is **no** other way in the world by which justice can be gained."*

The scope of the *no* negation clue started from *there* and ends with the *gained*. Our baseline system marked the negation from *there* to *justice*, but the constituent based method found the correct scope. If we look at the constituent tree we see that the full scope is covered by a constituent with *S* label. Instead of scope detection the constituent based information can't help in the event detection subtask.

5.3 Opinion Analysis

In the opinion analysis task (Johansson, 2017) the *label adjustment* method imporved 0.5 percetage points against the *mate-baseline* and got the best results in the shared task in Holders (In Vitro) metric. It seems the label combinations that our method found in the event extraction task is more general than we expected. On the other hand, it is still an open question why this label collapsing did not work at the event extraction task's evaluation set.

6 Conclusions

We introduced the contribution of team Szeged to the EPE 2017 Shared Task. We proposed three approaches for relational representation of syntax beyond the canonical dependency parse tree approach. Although these experiments are only the very first tries for such representations we hope

that this might give ideas about the important topic of syntactic and semantic parser solutions which aim to be (automatically) fine-tuned for a particular downstream applications demands.

Acknowledgments

We thank to the organizers of the EPE shared task for their hard work on extending the feature extractor of downstream applications according to our weird ideas.

The research work of Richárd Farkas has been supported by the János Bolyai scholarship of the Hungarian Academy of Sciences.

References

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

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 89–97.

Richárd Farkas and Bernd Bohnet. 2012. Stacking of dependency and phrase structure parsers. In *Proceedings of International Conference on Computational Linguistics 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 849–866.

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

Emanuele Lapponi and Stephan Oepen. 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 –.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 523–530.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, et al. 2015. Universal dependencies 1.2 .

Stephan Oepen, Jari Björne, Richard Johansson, Emanuele Lapponi, Filip Ginter, Erik Velldal, and Lilja Øvrelid. 2017. The 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017). 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 –.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '10, pages 19–27.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 433–440.

UPF at EPE 2017: Transduction-based Deep Analysis

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

Abstract

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

1 Credits

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

2 Introduction

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

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

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

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

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

3 Run 1: Surface-syntactic trees

3.1 Targeted dependency representation

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

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

Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017), pages 80–88,
Pisa, Italy, September 20, 2017.

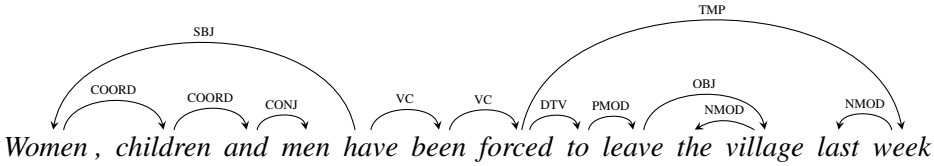


Figure 1: SSyntS for *Women, children and men have been forced to leave the village last week.*

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

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

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

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

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

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

3.2 Implementation

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

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

UAS	LAS	PoS
93.67	92.68	97.42

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

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

Table 2: Steps for surface-syntactic parsing

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

[2] https://nlp.stanford.edu/software/

[3] https://code.google.com/archive/p/mate-tools/downloads

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

4 Run 2: Deep-syntactic trees

4.1 Targeted dependency representation

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

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

The degree of "semanticity" of DSynt structures can be directly compared to Prague's tectogrammatical structures (PDT-tecto (Hajič et al., 2006), from which the *PSD* runs by some EPE participants stem from), which contain *autosemantic* words only. Thanks to the distinction between argumental and non-argumental edges, tectogrammatical structures are also trees, thus they maintain the syntactic structure of the sentence. The main differences between the two representations are: (i) in tectogrammatical structures, no distinction is made between governed and non-governed prepositions and conjunctions, and (ii) in tectogrammatical structures, the vocabulary used for edge labels emphasizes "semantic" content over predicate-argument information.

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

4.2 Implementation

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

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

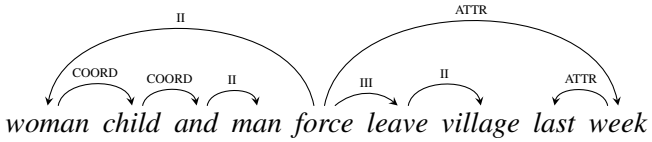


Figure 2: DSyntS for *Women, children and men have been forced to leave the village last week.*

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

Table 3: Rules for SSynt-DSynt mapping

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

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

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

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

```
rc:?Yr {
  rc:<=> ?Yl
  <=> ?Xl
  original_deprel = ?dep
}
```

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

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

For indicative purposes, a former evaluation

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

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

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

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

Hypernode Identification

Precision	Recall
97.00	99.96

Attachment and labeling

UAP	UAR	LAP	LAR
93.88	96.74	88.54	91.24

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

5 Run 3: Predicate-argument graphs

5.1 Targeted dependency representation

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

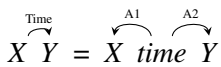


Figure 4: Correspondence between a non-core relation and a binary predicate

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

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

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

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

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

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

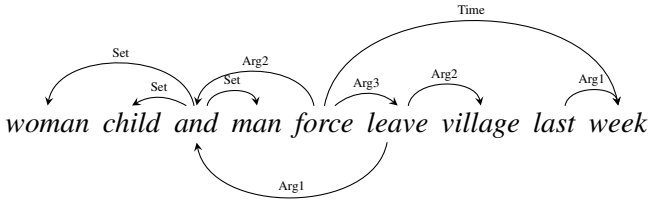


Figure 5: PredArgS for *Women, children and men have been forced to leave the village last week.*

(or valency) structure and ignores some functional elements (such as the *be* copula and governed prepositions) in the graph. DELPH-IN corresponds to the DM run of some EPE participants; see, e.g. (Chen et al., 2017; Schuster et al., 2017). The main differences are in the direction of some edges, the labels used, and the fact that all the words of the original sentence are in the representation, although not always connected with a dependency (in the same fashion as PropBank). Another example is the Enju annotation (Miyao, 2006), which is a pure predicate-argument graph over all words of a sentence. However, it distinguishes arguments of functional elements (auxiliaries, infinitive and dative TO, THAT, WHETHER, FOR complementizers, passive BY) in that they are attached to the semantic heads of these elements (rather than to the elements themselves). This facilitates the disregard of functional elements—as in DSyntSs.[10]

5.2 Implementation

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

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

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

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

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

Table 5: Rules for DSynt-PredArg mapping

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

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

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

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

6 Results and discussion

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

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

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

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

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

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

7 Future work

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

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

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

Acknowledgments

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Author Index