EPE2017: Towards a Reusable Infrastructure for Automated Extrinsic Parser Evaluation

Anonymous ACL submission

Abstract

1 Introduction & Motivation

2 Methodological Challenges

3 Related Work

Even though the bulk of work on parser evaluation focuses on intrinsic evaluation, there have been a few previous studies devoted to extrinsic parser evaluation and more specifically on the comparison of different types of syntactic representations.

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-based parsers assign a CoNLL-style analysis (see below) and are compared to PTB-style constituent parsers and the HPSG-based ENJU parser and 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 LTH and CoNLL07 representations in the down-stream task of semantic role labeling. They find that the dependency-based systems performs slightly better in the subtask of argument classification and whereas the constituent-based parsers achieve slightly higher results in argument identification. They further find that the LTH dependency scheme performs better than the CoNLL07 scheme in the task of argument classification.

The previous work that is most similar to ours 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: negation 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-X representation (based on the LTH converter of Johansson and Nugues (2007) using the -conll07 flag), the conversion scheme used in the English Web Treebank (based on the Stanford basic scheme (de Marneffe et al., 2006)) and the LTH scheme (based on the LTH converter of Johansson and Nugues (2007) using the -oldLTH flag). 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 scheme performs best, whereas the Stanford and LTH schemes provide superior SRL performance.

Although the main focus of this task is on the comparison of different representations, there are of course several other important dimensions of variation that will affect the results. On such dimension is the choice of parser and parsing strategy, for example; parsing directly to a dependency representation; parsing to constituent trees and then converting this to dependencies; and possibly augmenting the initial dependency representation with additional information through post-processing. The choice of training data will also have an impact, in addition to the pre-processing (sentence segmentation, tokenization, etc.).

4 Dependency-Based Syntactico-Semantic Analysis

Figure 1 presents a range of different dependency analyses for the example sentence *A similar technique is almost impossible to apply to other crops.* In (a) we see the analysis employed in the CoNLL08 shared task (**?**), obtained by converting

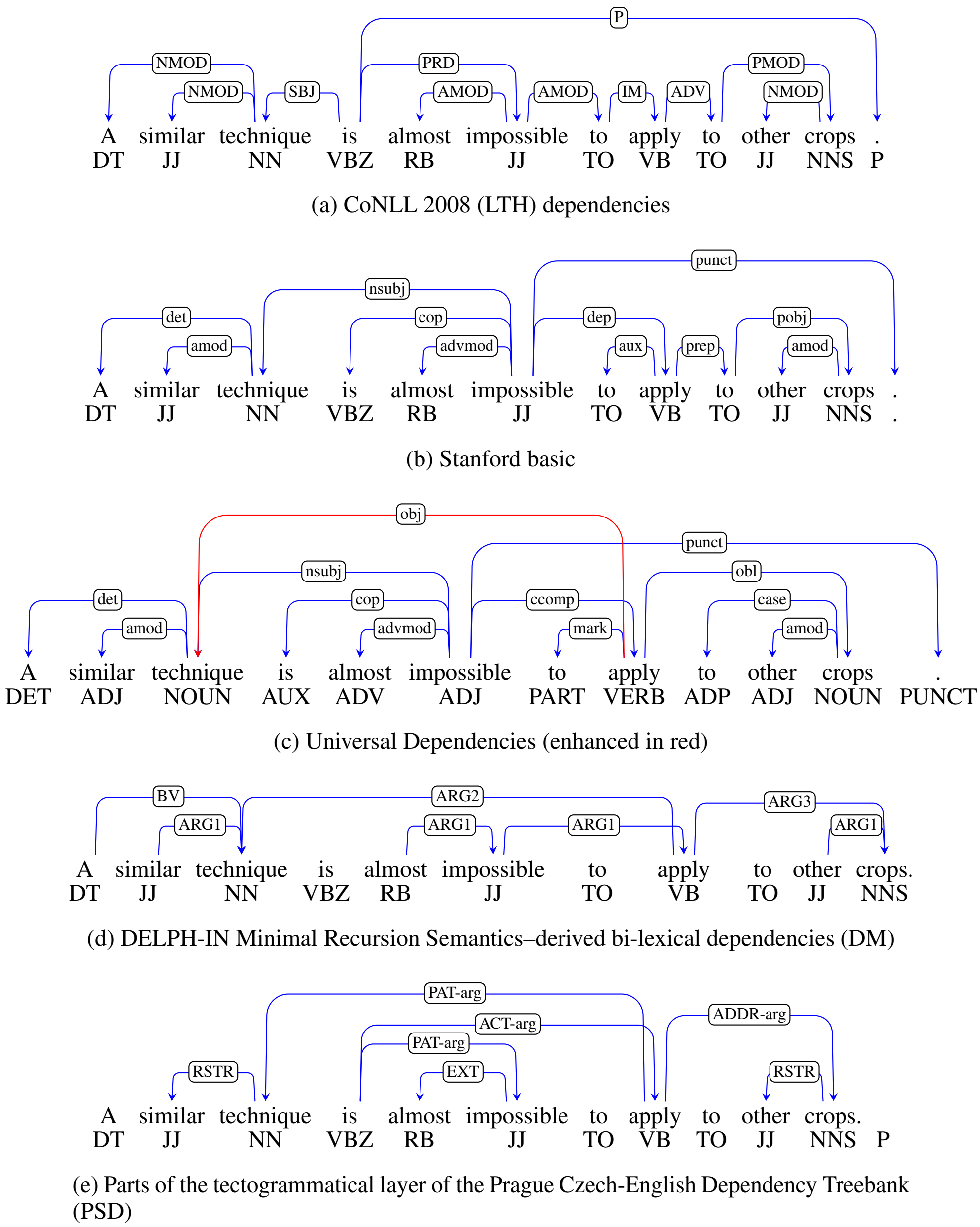

Figure 1: Dependency representations in (a) CoNLL08, (b) Stanford basic, (c) Universal Dependencies, (d) DM and (e) PSD.

PTB trees with the LTH pennconverter software (Johansson and Nugues, 2007), which relies on head finding rules (?) and the functional annotation already present in the PTB annotation. The Stanford basic representation in (b), is also a result of a conversion from PTB-style phrase structure trees—combining head finding rules with rules that target specific linguistic constructions, such as passives or attributive adjectives (de Marneffe et al., 2006). The Universal Dependencies (UD) representation in (c) (?) builds on several previous initiatives for universally common morphological (??) and syntactic dependency (??) annotation. This representation was employed in the recent CoNLL 2017 shared task (?), which was devoted to multilingual parsing from raw text for more than 40 different languages.

Whereas the three first representations are largely syntactic in nature, the following two provide examples of so-called semantic dependency representations: the DELPH-IN Minimal Recursion Semantics–derived dependencies (DM) in (d) and dependencies derived from the tectogrammatical layer of the Prague Czech-English Dependency Treebank (PSD) in (e).

The representations vary along several dimensions. First, we can distinguish between largely syntactic and semantic dependency representations. These vary both in terms of formal properties and the dependency relations employed. The syntactic representations, corresponding to the first three representations in Figure 1 largely assume that the dependency graphs are rooted trees, in the formal sense where every node can be reached via a single directed path from a distinguished root node[1]. The semantic dependency graphs, on the other hand, do not make this assumption and can be characterized formally as labeled directed graphs which allow for both node re-entrancies (such as that examplified by the token *technique* in (d)) and partial connectivity of the graph, i.e. leaving functional tokens, like infinitival markers and prepositions unanalyzed. The syntactic and semantic representations typically also differ in terms of dependency relation inventory. Whereas, the syntactic analyses are based on morphosyntactic categories and syntactic functions, the semantic relations encode deep arguments of predicates, and semantic roles.

Among the syntactic representations, we may distinguish between dependency representations that take a largely functional view of head status—e.g. functional elements like auxiliaries, subjunctions, and infinitival markers are heads—and more content-centered approaches where the lexical verbs or arguments of the copula are heads. In Figure 1 we observe that the CoNLL08 representation in (a) employs a functional head strategy (appointing the copula and infinitival marker as head), whereas the Stanford scheme largely chooses content words as heads. The UD scheme, which is clearly based on the Stanford scheme and has many similarities to it, takes this even further and additionally analyzes prepositional complements as heads (with prepositions as dependent case markers).

5 Definitions for the Task

Dependency Representation

Parsing System

Downstream Application

[1]Note however that this assumption does not hold for the UD representations in their v2.0 which also introduces so-called enhanced dependencies, which are not required to be trees.

6 The EPE Dependency Interchange Format

7 Downstream Applications

7.1 Biological Event Extraction

Event extraction refers to the detection of complex semantic relations. It differs from pairwise relation extraction in that events 1) have a defined trigger word (usually a verb) 2) can have 1–n arguments and 3) events 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 Challenge the event dataset used for training and evaluation is the GENIA corpus from the BioNLP'09 Shared Task, the task for which TEES was originally built (Kim et al., 2009). This corpus defines nine types of biochemical events annotated for over 10k sentences. A typical GENIA annotation could be for example for the sentence "Protein A regulates the binding of proteins B and C" a nested two-event structure *REGULATION(A, BINDING(B, 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 word 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.

7.2 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*, *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 expression *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 agnostic to the structure of the input representation. The original implementation made strong assumptions that the input conforms to the linguistic model of the CoNLL-2008 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 new model represents a set of shortest paths instead of a single unique path.

7.2.1 Evaluation Metrics

In the EPE shared task, we evaluated the systems in three different subtasks, corresponding to the evaluations by 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;
- determining the polarity of each extracted subjective expression, for instance that the two expressions in the examples are both negative.

For each of these subtasks, 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 "soft" 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. **[NB: I'm not sure what you decided in the end.]** For the final ranking of systems, we used the macro-average of the F-scores in the three subtasks.

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 subtask where we see most of the variation in performance.

7.3 Negation Resolution

The Negation Resolution (NR) system (Sherlock; Lapponi et al., 2012) determines, for a given sentence, the scope of negation cues. The system is built on the annotations of the ConanDoyle-neg data set (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 the events in question did not take place. In the example

Since we have been so **un***fortunate* as to miss him [. . .]

the cue (in bold) affects the proposition *we have been so fortunate as to miss him* (its scope, underlined), and *fortunate* is its negated event.

Sherlock looks at NR as a classical sequence labeling problem. The main component in the Sherlock pipeline 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 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 model's feature set 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 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 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 tag-sets, this step was removed.

Evaluation measures for Sherlock runs in the EPE shared task include scope tokens (ST), event match (EM), scope match (SM), and full negation (FN) F_1 scores. ST and EM 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, including (FN) and excluding (SM) negated events.

8 Participating Teams

9 Experimental Results

10 High-Level Reflections

11 Conclusion & Outlook

Acknowledgments

Richard Johansson was supported by the Swedish Research Council under grant 2013–4944.

Team	Run	Representation	Event Extraction			Negation Resolution			Opinion Analysis			Avg	Rank
			P	R	F	P	R	F	P	R	F		
ECNU	0	UD v2.0	49.48	39.00	43.62	99.17	45.45	62.33	60.27	57.42	58.81	54.92	
	1	UD v2.0	50.72	38.97	44.08	99.17	45.45	62.33	62.86	60.04	61.42	55.94	
	2	UD v2.0	52.24	40.23	**45.46**	99.17	45.45	62.33	62.15	59.75	60.93	**56.24**	5
	3	UD v2.0	54.53	35.58	43.06	99.18	45.83	**62.69**	62.11	58.17	60.08	55.28	
	4	UD v2.0	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	59.11	37.71	46.04	99.12	42.80	**59.78**	65.04	51.32	57.37	54.40	
	1	PAS	52.39	40.98	45.99	99.09	41.29	58.29	65.80	52.73	58.54	54.27	
	2	UD v1 basic	55.79	44.56	**49.55**	99.04	39.02	55.98	65.87	61.30	63.50	56.34	
	3	UD v1 enh	57.48	41.64	48.29	99.06	39.77	56.75	66.22	62.43	**64.27**	56.44	
	4	UD v1 enh++	58.55	39.50	47.17	99.03	38.64	55.59	65.10	61.75	63.38	55.38	
	5	UD v1 enh++ dia	55.58	43.37	48.72	99.03	38.64	55.59	66.62	62.03	64.24	56.18	
	6	UD v1 enh++ dia–	58.11	39.19	46.81	99.06	39.77	56.75	64.21	60.27	62.18	55.25	
	7	UD v1 basic	57.69	42.80	49.14	99.05	39.39	56.36	65.78	60.96	63.28	56.26	
	8	UD v1 enh	54.90	44.75	49.31	99.07	40.15	57.14	65.59	62.42	63.97	**56.81**	3
	9	UD v1 enh++	58.03	43.02	49.41	99.04	39.02	55.98	66.77	61.04	63.78	56.39	
	10	UD v1 enh++ dia	59.88	40.19	48.10	98.97	36.36	53.18	65.86	60.92	63.29	54.86	
	11	UD v1 enh++ dia–	58.92	40.07	47.70	99.06	39.77	56.75	64.90	60.56	62.65	55.70	
Peking	0	DM	59.28	34.22	43.39	99.15	43.94	60.89	65.63	53.64	59.03	54.44	
	1	CCD	58.26	40.07	**47.48**	99.15	44.32	**61.26**	66.57	54.55	59.96	**56.23**	6
	2	DM											
	3	CCD											
	4	DM				99.10	41.67	58.67	65.74	53.66	59.09		
	5	CCD				99.12	42.42	59.41	66.97	54.84	**60.30**		
Prague	0	UD v2.0	53.84	36.61	43.58	99.10	41.83	58.83	62.61	57.21	59.79	54.07	
	1	UD v2.0	56.35	38.21	**45.54**	99.16	44.70	**61.62**	62.31	59.74	**61.00**	**56.05**	7
	2	UD v2.0	53.22	37.87	44.25	99.12	42.97	59.95	63.45	54.63	58.71	54.30	
	3	UD v2.0	51.91	36.27	42.70	99.12	42.97	59.95	61.26	56.72	58.90	53.85	
	4	UD v1.2	51.71	37.12	43.22	98.90	34.22	50.85	61.00	56.25	58.53	50.86	
Stanford and Paris	0	Stanford Basic	56.93	45.03	***50.29***	99.22	48.48	65.13	67.26	60.54	63.72	59.71	
	1	UD v1 basic	57.59	40.76	47.73	99.19	46.21	63.05	67.47	61.30	64.24	58.34	
	2	UD v1 enh	57.24	40.98	47.76	99.20	46.97	63.75	67.69	61.02	64.18	58.57	
	3	UD v1 enh++	56.76	42.74	48.76	99.21	47.35	64.10	67.43	61.58	64.37	59.08	
	4	UD v1 enh++ dia	58.86	40.51	47.99	99.19	46.21	63.05	66.68	61.95	64.23	58.42	
	5	UD v1 basic	58.75	42.21	49.13	99.22	48.11	64.80	68.18	61.56	64.70	59.54	
	6	UD v1 enh	58.36	44.09	50.23	99.24	49.62	***66.16***	68.86	61.81	65.14	***60.51***	***1***
	7	UD v1 enh++	62.30	41.55	49.85	99.20	46.97	63.75	68.44	62.25	**65.20**	59.60	
	8	UD v1 enh++ dia	57.47	44.47	50.14	99.21	47.73	64.45	67.64	62.57	65.01	59.87	
	9	UD v1 enh++ dia–	55.29	43.21	48.51	99.16	44.70	61.62	66.68	61.42	63.94	58.02	
	10	UD v1 enh++ dia–	57.22	42.83	48.99	99.22	48.48	65.13	67.30	62.01	64.55	59.56	
Szeged	0		60.20	39.69	**47.84**	99.17	45.08	**61.98**	66.73	65.04	65.87	**58.57**	2
	1		59.09	39.53	47.37	99.14	43.56	60.53	67.04	65.63	***66.33***	58.07	
	2		57.93	39.13	46.71	99.15	44.32	61.26	66.05	60.45	63.13	57.03	
	3		55.14	40.48	46.69	99.12	42.80	59.78	65.35	61.28	63.25	56.57	
	4		55.12	39.41	45.96	99.11	42.05	59.05	63.37	61.66	62.50	55.84	
UPF	0	SSyntS	53.21	41.36	**46.54**	99.12	42.80	**59.78**	66.25	61.19	**63.62**	**56.65**	4
	1	DSyntS	54.06	39.94	45.94	98.15	20.08	33.34	64.65	56.71	60.42	46.57	
	2	PredArg	56.37	39.63	**46.54**	97.96	18.18	30.67	61.03	51.50	55.86	44.36	
UW	0	DM	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. For the Paris/Stanford runs, 'enh' is short for 'enhanced' and 'dia' for 'diathesis'. The best F-scores for each team for each task are indicated with bold face, while the globally best scores are indicated with bold and italics. 'Avg' shows the average F1 across tasks.