Evaluating the Impact of Alternative Dependency Graph Encodings on Solving Event Extraction Tasks

Ekaterina Buyko and **Udo Hahn**
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30, 07743 Jena, Germany
{ekaterina.buyko|udo.hahn}@uni-jena.de

Abstract

In state-of-the-art approaches to information extraction (IE), dependency graphs constitute the fundamental data structure for syntactic structuring and subsequent knowledge elicitation from natural language documents. The top-performing systems in the BioNLP 2009 Shared Task on Event Extraction all shared the idea to use dependency structures generated by a variety of parsers — either directly or in some converted manner — and optionally modified their output to fit the special needs of IE. As there are systematic differences between various dependency representations being used in this competition, we scrutinize on different encoding styles for dependency information and their possible impact on solving several IE tasks. After assessing more or less established dependency representations such as the *Stanford* and *CoNLL-X* dependencies, we will then focus on trimming operations that pave the way to more effective IE. Our evaluation study covers data from a number of constituency- and dependency-based parsers and provides experimental evidence which dependency representations are particularly beneficial for the event extraction task. Based on empirical findings from our study we were able to achieve the performance of 57.2% F-score on the development data set of the BioNLP Shared Task 2009.

1 Introduction

Relation and event extraction are among the most demanding semantics-oriented NLP challenge tasks (both in the newspaper domain such as for ACE[1], as well as in the biological domain such as for BioCreative[2] or the BioNLP Shared Task[3]), comparable in terms of analytical complexity with recent efforts directed at opinion mining (e.g., NTCIR-7[4] or TREC Blog tracks[5]) or the recognition of textual entailment.[6] The most recent *BioNLP 2009 Shared Task on Event Extraction* (Kim et al., 2009) required, for a sample of 260 MEDLINE abstracts, to determine all mentioned events — to be chosen from a given set of nine event types, including *"Localization"*, *"Binding"*, *"Gene Expression"*, *"Transcription"*, *"Protein Catabolism"*, *"Phosphorylation"*, *"Positive Regulation"*, *"Negative Regulation"*, and (unspecified) *"Regulation"* — and link them appropriately with *a priori* supplied protein annotations. The demands on text analytics to deal with the complexity of this Shared Task in terms of relation diversity and specificity are unmatched by former challenges.

For relation extraction in the biomedical domain (the focus of our work), a stunning convergence towards dependency-based syntactic representation structures is witnessed by the performance results of the top-performing systems in the *BioNLP'09*

[1] `http://papers.ldc.upenn.edu/LREC2004/ACE.pdf`
[2] `http://biocreative.sourceforge.net/`
[3] `www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/`
[4] `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings7/pdf/revise/01-NTCIR-OV-MOAT-SekiY-revised-20081216.pdf`
[5] `http://trec.nist.gov/data/blog08.html`
[6] `http://pascallin.ecs.soton.ac.uk/Challenges/RTE/`

Shared Task on Event Extraction.[7] Regarding the fact that dependency representations were always viewed as a vehicle to represent fundamental semantic relationships already at the syntactic level, this is not a great surprise. Yet, dependency grammar is not a monolithic, consensually shaped and well-defined linguistic theory. Accordingly, associated parsers tend to vary in terms of dependency pairing or structuring (which pairs of words join in a dependency relation?) and dependency typing (how are dependency relations for a particular pair labelled?).

Depending on the type of dependency theory or parser being used, various representations emerge (Miyao et al., 2007). In this paper, we explore these different representations of the dependency graphs and try, first, to pinpoint their effects on solving the overall event extraction task and, second, to further enhance the potential of JREX, a high-performance relation and event extractor developed at the JULIE Lab (Buyko et al., 2009).

2 Related Work

In the biomedical domain, the focus has largely been on binary relations, in particular protein-protein interactions (PPIs). Accordingly, the biomedical NLP community has developed various PPI-annotated corpora (e.g., LLL (Nédellec, 2005), AIMED (Bunescu et al., 2005), BIOINFER (Pyysalo et al., 2007)). PPI extraction does clearly not count as a solved problem, and a deeper look at its biological and representational intricacies is certainly worthwhile. The GENIA event corpus (Kim et al., 2008) and the BioNLP 2009 Shared Task data (Kim et al., 2009) contain such detailed annotations of PPIs (amongst others).

The BioNLP Shared Task was a first step towards the extraction of specific pathways with precise information about the molecular events involved. In that task, 42 teams participated and 24 of them submitted final results. The winner system, TURKU (Björne et al., 2009), achieved with 51.95% F-score the milestone result in that competition followed by the JULIELab system (Buyko et al., 2009) which peaked at 46.7% F-score. Only recently, an extension of the TURKU system, the TOKYO system, has been realized (Miwa et al., 2010). TOKYO system's event extraction capabilities are based on the TURKU system, yet TURKU's manually crafted rule system for post-processing and the combination of extracted trigger-argument relations is replaced by a machine learning approach in which rich features collected from classification steps for triggers and arguments are re-combined. TOKYO achieves an overall F-score of 53.29% on the test data, thus outperforming TURKU by 1.34 percentage points.

The three now top-performing systems, TOKYO, TURKU and JULIELab, all rely on dependency graphs for solving the event extraction tasks. While the TURKU system exploits the Stanford dependencies from the McClosky-Charniak parser (Charniak and Johnson, 2005), and the JULIELab system uses the CoNLL-like dependencies from the GDep parser (Sagae and Tsujii, 2007),[8] the TOKYO system overlays the Shared Task data with two parsing representations, *viz.* Enju PAS structure (Miyao and Tsujii, 2002) and GDep parser dependencies. Obviously, one might raise the question as to what extent the performance of these systems depends on the choice of the parser and its output representations. Miyao et al. (2008) already assessed the impact of different parsers for the task of biomedical relation extraction (PPI). Here we perform a similar study for the task of event extraction and focus, in particular, on the impact of various dependency representations such as Stanford and CoNLL'X dependencies and additional trimming procedures.

For the experiments on which we report here, we performed experiments with the JULIELab system. Our main goal is to investigate into the crucial role of proper representation structures for dependency graphs so that the performance gap from Shared Task results between the best-performing TOKYO system and the JULIELab system be narrowed.

3 Event Extraction

3.1 Objective

Event extraction is a complex task that can be subdivided into a number of subtasks depending on

[7] `www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/results/results-master.html`

[8] The GDep parser has been trained on the GENIA Treebank pre-official version of the version 1.0 converted with the script available from `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`

whether the focus is on the event itself or on the arguments involved:

Event trigger identification deals with the large variety of alternative verbalizations of the same event type, e.g., whether the event is expressed in a verbal or in a nominalized form ("*A is expressed*" as well as "*the expression of A*" both refer to the same event type, *viz.* $Expression(A)$). Since the same trigger may stand for more than one event type, event trigger ambiguity has to be resolved as well.

Event trigger disambiguation selects the correct event name from the set of alternative event triggers.

Argument identification is concerned with finding all necessary participants in an event, i.e., the arguments of the relation.

Argument ordering assigns each identified participant its functional role within the event, mostly $Agent$ and $Patient$.

3.2 JULIELab System

The JULIELab solution can best be characterized as a single-step learning approach for event detection as the system does not separate the overall learning task into independent event trigger and event argument learning subtasks.[9] The JULIELab system incorporates manually curated dictionaries and machine learning (ML) methodologies to sort out associated event triggers and arguments on dependency graph structures. For argument extraction, the JULIELab system uses two ML-based approaches, a feature-based and a kernel-based one. Given that methodological framework, the JULIELab team scored on 2nd rank among 24 competing teams, with 45.8% precision, 47.5% recall and 46.7% F-score on all 3,182 events. After the competition, this system was updated and achieved 57.6% precision, 45.7% recall and 51.0% F-score (Buyko et al., 2010) using modified dependency representations from the MST parser (McDonald et al., 2005). In this study, we perform event extraction experiments with various dependency representations that allow us to measure their effects on the event extraction task and to increase the overall JULIELab system performance in terms of F-score.

[9]The JULIELab system considers all relevant lexical items as potential event triggers which might represent an event. Only those event triggers that can eventually be connected to arguments, finally, represent a true event.

4 Dependency Graph Representations

In this section, we focus on representation formats of dependency graphs. In Section 4.1, we introduce fundamental notions underlying dependency parsing and consider established representation formats for dependency structures as generated by various parsers. In Section 4.2, we account for selected trimming operations for dependency graphs to ease IE.

4.1 Dependency Structures: Representation Issues

Dependency parsing, in the past years, has increasingly been recognized as an alternative to long-prevailing constituency-based parsing approaches, particularly in semantically-oriented application scenarios such as information extraction. Yet even under purely methodologically premises, it has gained wide-spread attention as witnessed by recent activities performed as part of the "CoNLL Shared Tasks on Multilingual Dependency Parsing" (Buchholz and Marsi, 2006).

In a nutshell, in dependency graphs of sentences, nodes represent single words and edges account for head-modifier relations between single words. Despite this common understanding, concrete syntactic representations often differ markedly from one dependency theory/parser to the other. The differences fall into two main categories: dependency pairing or structuring (which pairs of words join in a dependency relation?) and dependency typing (how are dependency relations for a particular pair labelled?).

The CoNLL'X dependencies, for example, are defined by 54 relation types,[10] while the Stanford scheme (de Marneffe et al., 2006) incorporates 48 types (so called grammatical relations or Stanford dependencies). The Link Grammar Parser (Sleator and Temperley, 1991) employs a particularly fine-grained repertoire of dependency relations adding up to 106 types, whereas the well-known MINIPAR parser (Lin, 1998) relies on 59 types. Differences in dependency structure are at least as common as differences in dependency relation typing (see below).

[10]Computed by using the conversion script on WSJ data (accessible via `http://nlp.cs.lth.se/pennconverter/`; see also Johansson and Nugues (2007) for additional information). From the GENIA corpus, using this script, we could only extract 29 CoNLL dependency relations.

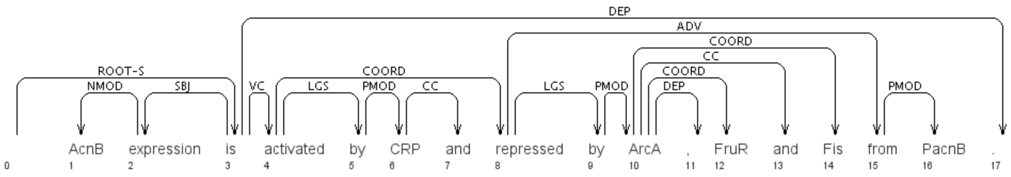

Figure 1: Example of CoNLL 2008 dependencies, as used in most of the native dependency parsers.

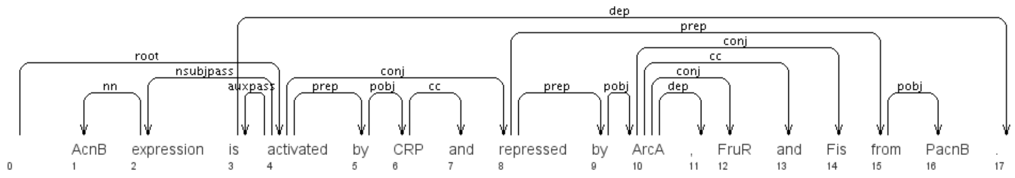

Figure 2: Stanford dependencies, *basic* conversion from Penn Treebank.

In general, dependency graphs can be generated by syntactic parsers in two ways. First, native dependency parsers output CoNLL'X or Stanford dependencies dependent on which representation format they have been trained on.[11] Second, in a derivative dependency mode, the output of constituency-based parsers, e.g., phrase structure representations, is subsequently converted either into CoNLL'X or Stanford dependencies using Treebank conversion scripts (see below). In the following, we provide a short description of these two established dependency graph representations:

- **CoNLL'X dependencies (CD).** This dependency tree format was used in the CoNLL'X Shared Tasks on multi-lingual dependency parsing (see Figure 1). It has been adopted by most native dependency parsers and was originally obtained from Penn Treebank (PTB) trees using constituent-to-dependency conversion (Johansson and Nugues, 2007). It differs slightly in the number and types of dependencies being used from various CoNLL rounds (e.g., CoNLL'08 provided a dependency type for representing appositions).[12]

- **Stanford dependencies (SD).** This format was proposed by de Marneffe et al. (2006) for semantics-sensitive applications using dependency representations, and can be obtained using the Stanford tools[13] from PTB trees. The Stanford format is widely used in the biomedical domain (e.g., by Miyao et al. (2008) or Clegg and Shepherd (2005)).

There are systematic differences between CoNLL'X and Stanford dependencies, e.g., as far as the representation of passive constructions, the position of auxiliary and modal verbs, or coordination representation is concerned. In particular, the representation of the *passive* construction and the role of the auxiliary verb therein may have considerable effects for semantics-sensitive tasks. While in SD the subject of the passive construction is represented by a special `nsubj` dependency label, in CD we find the same subject label as for active constructions `SUB(J)`. On CoNLL'08 data, the logical subject is marked by the `LGS` dependency edge that connects the passive-indicating preposition *"by"* with the logical subject of the sentence.

The representation of *active* constructions are similar in CD and SD though besides the role of auxiliary and modal verbs. In the Stanford dependency representation scheme, rather than taking auxiliaries to be the heads in passive or tense constructions, main verbs are assigned this grammatical function (see Figure 2). The CoNLL'X represen-

[11]We disregard in this study other dependency representations such as MINIPAR and LINK GRAMMAR representations.

[12]For the differences between CoNLL'07 and CoNLL'08 representations, cf. `http://nlp.cs.lth.se/software/treebank_converter/`

[13]Available from `nlp.stanford.edu/software/lex-parser.shtml`

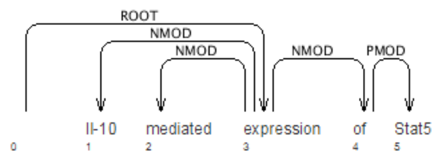

Figure 3: Noun phrase representation in CoNLL'X dependency trees.

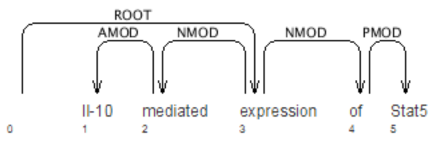

Figure 4: Trimming procedure *noun phrase* on CoNLL'X dependency trees.

tation scheme is completely different in that auxiliaries – much in common with standard dependency theory – are chosen to occupy the role of the governor (see Figure 1). From the perspective of relation extraction, however, the Stanford scheme is certainly closer to the desired predicate-argument structure representations than the CoNLL scheme.

4.2 Dependency Graph Modifications in Detail

Linguistic intuition suggests that the closer a dependency representation is to the format of the targeted semantic representation, the more likely will it support the semantic application. This idea is directly reflected in the Stanford dependencies which narrow the distance between nodes in the dependency graph by collapsing procedures (the so-called *collapsed* mode of phrase structure conversion). An example of collapsing is the conversion of "*expression* $\xrightarrow{\text{nmod}}$ *in* $\xrightarrow{\text{pmod}}$ *cells*" to "*expression* $\xrightarrow{\text{prep\_in}}$ *cells*". An extension of collapsing is the re-structuring of coordinations with sharing the dependency relations of conjuncts (the so-called *ccprocessed* mode of phrase structure conversion).

According to the Stanford scheme, Buyko et al. (2009) proposed collapsing scenarios on CoNLL'X dependency graphs. Their so-called *trimming* operations treat three syntactic phenomena, *viz.* coordinations (*coords*), auxiliaries/modals (*auxiliaries*), and prepositions (*preps*). For coordinations, they propagate the dependency relation of the first conjunct to all the other conjuncts within the coordination. For auxiliaries/modals, they prune the auxiliaries/modals as governors from the dependency graph and propagate the dependency relations of these nodes to the main verbs. Finally, for prepositions, they collapse a pair of typed dependencies into a single typed dependency (as illustrated above).

For the following experiments, we extended the trimming procedures and propose the re-structuring of noun phrases with action adjectives to make the dependency representation even more compact for semantic interpretation. The original dependency representation of the noun phrase selects the rightmost noun as the head of the NP and thus all remaining elements are its dependents (see Figure 3). For the noun phrases containing action adjectives (mostly verb derivations) this representation does not reflect the true semantic relations between the elements. For example, in *"IL-10 mediated expression"* it is *"IL-10"* that mediates the expression. Therefore, we re-structure the dependency graph by changing the head of *"IL-10"* from *"expression"* to *"mediated"*. Our re-coding heuristics selects, first, all the noun phrases containing action adjectives ending with *"-ed"*, *"-ing"*, *"-ible"* suffixes and with words such as *"dependent"*, *"specific"*, *"like"*. In the second step, we re-structure the noun phrase by encoding the adjective as the head of all the nouns preceding this adjective in the noun phrase under scrutiny (see Figure 4).

5 Experiments and Results

In this section, we describe the experiments and results related to event extraction tasks based on alternative dependency graph representations. For our experiments, we selected the following top-performing parsers — the first three phrase structure based and thus the origin of derivative dependency structures, the last three fully dependency based for making native dependency structures available:

- **C+J**, Charniak and Johnson's reranking parser (Charniak and Johnson, 2005), with the WSJ-trained parsing model.
- **M+C**, Charniak and Johnson's reranking parser (Charniak and Johnson, 2005), with the self-trained biomedical parsing model from McClosky (2010).

- **Bikel**, Bikel's parser (Bikel, 2004) with the WSJ-trained parsing model.
- **GDep** (Sagae and Tsujii, 2007), a native dependency parser.
- **MST** (McDonald et al., 2005), another native dependency parser.
- **MALT** (Nivre et al., 2007), yet another native dependency parser.

The native dependency parsers were re-trained on the GENIA Treebank (Tateisi et al., 2005) conversions.[14] These conversions,[15] i.e., Stanford *basic*, CoNLL'07 and CoNLL'08 were produced with the currently available conversion scripts. For the Stanford dependency conversion, we used the Stanford parser tool,[16] for CoNLL'07 and CoNLL'08 we used the treebank-to-CoNLL conversion scripts[17] available from the CoNLL'X Shared Task organizers.

The phrase structure based parsers were applied with already available models, i.e., the Bikel and C+J parsers as trained on the WSJ corpus, and M+C as trained on the GENIA Treebank corpus. For our experiments, we converted the prediction results of the phrase structure based parsers into five dependency graph representations, *viz.* Stanford *basic*, Stanford *collapsed*, Stanford *ccprocessed*, CoNLL'07 and CoNLL'08, using the same scripts as for the conversion of the GENIA Treebank.

The JULIELab event extraction system was re-trained on the Shared Task data enriched with different outputs of syntactic parsers as described above. The results for the event extraction task are represented in Table 1. Due to the space limitation of this paper we provide the summarized results of important event extraction sub-tasks only, i.e., results for basic events (*Gene Expression*, *Transcription*, *Localization*, *Protein Catabolism*) are summarized under SVT-TOTAL; regulatory events are summarized under REG-TOTAL; the overall extraction results are listed in ALL-TOTAL (see Table 1).

Obviously, the event extraction system trained on various dependency representations indeed produces truly different results. The differences in terms of F-score come up to 2.4 percentage points for the SVT-TOTAL events (cf. the MALT parser, difference between SD *basic* (75.6% F-score) and CoNLL'07 (78.0% F-score)), up to 3.6 points for REG-TOTAL (cf. the M+C parser, difference between SD *ccprocessed* (40.9% F-score) and CoNLL'07 (44.5% F-score)) and up to 2.5 points for ALL-TOTAL (cf. the M+C parser, difference between SD *ccprocessed* (52.8% F-score) and CoNLL'07 (55.3% F-score)).

The top three event extraction results on the development data based on different syntactic parsers results are achieved with M+C parser – CoNLL'07 representation (55.3% F-score), MST parser – CoNLL'08 representation (54.6% F-score) and MALT parser – CoNLL'08 representation (53.8% F-score) (see Table 1, ALL-TOTAL). Surprisingly, both the CoNLL'08 and CoNLL'07 formats clearly outperform Stanford representations on all event extraction tasks. Stanford dependencies seem to be useful here only in the *basic* mode. The *collapsed* and *ccprocessed* modes produce even worse results for the event extraction tasks.

Our second experiment focused on trimming operations on CoNLL'X dependency graphs. Here we performed event extraction after the trimming of the dependency trees as described in Section 4.2 in different modes: *coords* – re-structuring coordinations; *preps* – collapsing of prepositions; *auxiliaries* – propagating dependency relations of auxiliars and modals to main verbs; *noun phrase* – re-structuring noun phrases containing action adjectives. Our second experiment showed that the extraction of selected events can profit in particular from the trimming procedures *coords* and *auxiliaries*, but there is no evidence for a general trimming configuration for the overall event extraction task.

In Table 2 we summarize the best configurations we found for the events in focus. It is quite evident that the CoNLL'08 and CoNLL'07 dependencies modified for auxiliaries and coordinations are the best configurations for four events (out of nine). For three events no modifications are necessary and

[14] For the training of dependency parsers, we used from the available Stanford conversion variants only Stanford *basic*. The *collapsed* and *ccprocessed* variants do not provide dependency trees and are not recommended for training native dependency parsers.

[15] We used the GENIA Treebank version 1.0, available from `www-tsujii.is.s.u-tokyo.ac.jp`

[16] `http://nlp.stanford.edu/software/lex-parser.shtml`

[17] `http://nlp.cs.lth.se/software/treebank_converter/`

Parser	SD basic			SD collapsed			SD ccprocessed			CoNLL'07			CoNLL'08		
SVT-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	70.5	75.5	**72.9**	70.7	74.5	72.5	71.6	73.5	72.5	69.4	75.9	72.5	69.7	75.7	72.6
C+J	73.0	77.4	75.1	73.2	77.3	75.2	72.8	77.2	75.0	73.5	78.3	**75.8**	73.0	77.9	75.4
M+C	76.4	78.0	77.2	76.4	77.6	77.0	76.4	77.2	76.8	76.4	79.0	77.7	76.6	79.3	**77.9**
GDEP	77.1	77.5	**77.3**	N/A	N/A	N/A	N/A	N/A	N/A	72.5	80.2	76.1	72.6	77.2	74.8
MALT	73.1	78.2	75.6	N/A	N/A	N/A	N/A	N/A	N/A	75.9	80.3	**78.0**	73.7	78.2	75.9
MST	76.4	78.5	77.4	N/A	N/A	N/A	N/A	N/A	N/A	74.8	78.4	76.6	76.7	80.8	**78.7**
REG-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	35.3	40.6	**37.8**	33.8	40.3	36.8	34.3	39.6	36.8	33.9	39.2	36.3	34.0	41.0	37.2
C+J	36.2	41.8	38.8	37.3	41.8	39.4	36.5	41.9	39.0	38.1	43.9	**40.8**	37.4	44.0	40.4
M+C	39.4	45.5	42.3	38.8	45.3	41.8	38.5	43.7	40.9	41.9	47.4	**44.5**	40.1	47.9	43.7
GDEP	39.6	42.8	41.6	N/A	N/A	N/A	N/A	N/A	N/A	38.4	43.7	40.9	39.8	44.4	**42.0**
MALT	38.8	44.3	41.4	N/A	N/A	N/A	N/A	N/A	N/A	39.0	44.3	41.5	39.2	46.4	**42.5**
MST	39.5	43.6	41.4	N/A	N/A	N/A	N/A	N/A	N/A	39.6	45.6	42.4	40.6	45.8	**43.0**
ALL-TOTAL															
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Bikel	47.4	51.5	**49.4**	46.3	50.8	48.5	46.9	50.2	48.5	44.8	50.7	47.6	44.7	51.8	48.0
C+J	49.3	53.8	51.5	49.6	52.8	51.2	49.0	53.0	50.9	50.3	54.4	**52.3**	49.5	54.3	51.8
M+C	52.3	56.4	54.3	51.8	55.7	53.7	51.3	54.3	52.8	53.2	57.5	**55.3**	52.2	58.2	55.0
GDEP	52.7	54.5	**53.6**	N/A	N/A	N/A	N/A	N/A	N/A	50.6	55.2	52.8	51.3	55.0	53.1
MALT	50.4	54.7	52.4	N/A	N/A	N/A	N/A	N/A	N/A	51.5	56.0	53.7	51.2	56.8	**53.8**
MST	52.3	54.8	53.5	N/A	N/A	N/A	N/A	N/A	N/A	51.7	56.4	53.9	52.4	56.9	**54.6**

Table 1: Results on the Shared Task development data for Event Extraction Task. Approximate Span Matching/Approximate Recursive Matching.

Event Class	Best Parser	Best Configuration	R	P	F
Gene Expression	MST	CoNLL'08, *auxiliaries, coords*	79.5	81.8	80.6
Transcription	MALT	CoNLL'07, *auxiliaries, coords*	67.1	75.3	71.0
Protein Catabolism	MST	CoNLL'08, *preps*	85.7	100	92.3
Phosphorylation	MALT	CoNLL'08	80.9	88.4	84.4
Localization	MST	CoNLL'08, *auxiliaries*	81.1	87.8	84.3
Binding	MST	CoNLL'07, *auxiliaries, coords, noun phrase*	51.2	51.0	51.1
Regulation	MALT	CoNLL'07, *auxiliaries, coords*	30.8	49.5	38.0
Positive Regulation	M+C	CoNLL'07	43.0	49.9	46.1
Negative Regulation	M+C	CoNLL'07	49.5	45.3	47.3

Table 2: Best Configurations for Dependency Representations for Event Extraction Task on the development data.

Binding	R	P	F
CoNLL'07	47.3	46.8	47.0
CoNLL'07 *auxiliaries, coords*	46.8	48.1	47.4
CoNLL'07 *auxiliaries, coords, noun phrase*	51.2	51.0	**51.1**

Table 3: Effects of trimming of *CoNLL* dependencies on the Shared Task development data for *Binding* events. Approximate Span Matching/Approximate Recursive Matching. The data was processed by the MST parser.

		JULIELab (M+C, CoNLL'08)			JULIELab Final Configuration			TOKYO System		
Event Class	gold	R	P	F	R	P	F	R	P	F
Gene Expression	356	79.2	80.3	79.8	79.5	81.8	80.6	78.7	79.5	79.1
Transcription	82	59.8	72.0	65.3	67.1	75.3	71.0	65.9	71.1	68.4
Protein Catabolism	21	76.2	88.9	82.0	85.7	100	92.3	95.2	90.9	93.0
Phosphorylation	47	83.0	81.2	82.1	80.9	88.4	84.4	85.1	69.0	76.2
Localization	53	77.4	74.6	75.9	81.1	87.8	84.3	71.7	82.6	76.8
SVT-TOTAL	559	76.4	79.0	**77.7**	78.2	82.6	**80.3**	77.3	77.9	**77.6**
Binding	248	45.6	45.9	45.8	51.2	51.0	51.1	50.8	47.6	49.1
EVT-TOTAL	807	66.9	68.7	**67.8**	69.9	72.5	**71.2**	69.1	68.1	**68.6**
Regulation	169	32.5	46.2	38.2	30.8	49.5	38.0	36.7	46.6	41.1
Positive_regulation	617	42.3	49.0	45.4	43.0	49.9	46.1	43.9	51.9	47.6
Negative_regulation	196	48.5	44.0	46.1	49.5	45.3	47.3	38.8	43.9	41.2
REG-TOTAL	982	41.9	47.4	**44.5**	42.2	48.7	**45.2**	41.7	49.4	**45.2**
ALL-TOTAL	1789	53.2	57.5	**55.3**	54.7	60.0	**57.2**	54.1	58.7	**56.3**

Table 4: Results on the Shared Task development data. Approximate Span Matching/Approximate Recursive Matching.

		JULIELab (Buyko et al., 2010)			JULIELab Final Configuration			TOKYO system		
Event Class	gold	R	P	F	R	P	F	R	P	F
Gene Expression	722	66.3	79.6	72.4	67.0	77.2	71.8	68.7	79.9	73.9
Transcription	137	33.6	61.3	43.4	35.0	60.8	44.4	54.0	60.7	57.1
Protein Catabolism	14	71.4	90.9	80.0	71.4	90.9	80.0	42.9	75.0	54.6
Phosphorylation	135	80.0	85.0	82.4	80.7	84.5	82.6	84.4	69.5	76.3
Localization	174	47.7	93.3	63.1	45.4	90.8	60.5	47.1	86.3	61.0
SVT-TOTAL	1182	61.4	80.3	**69.6**	61.8	78.2	**69.0**	65.3	76.4	**70.4**
Binding	347	47.3	52.4	49.7	47.3	52.2	49.6	52.2	53.1	52.6
EVT-TOTAL	1529	58.2	73.1	**64.8**	58.5	71.7	**64.4**	62.3	70.5	**66.2**
Regulation	291	24.7	40.5	30.7	26.8	38.2	31.5	28.9	39.8	33.5
Positive Regulation	983	35.8	45.4	40.0	34.8	45.8	39.5	38.0	48.3	42.6
Negative Regulation	379	37.2	39.7	38.4	37.5	40.9	39.1	35.9	47.2	40.8
REG-TOTAL	1653	34.2	43.2	**38.2**	34.0	43.3	**38.0**	35.9	46.7	**40.6**
ALL-TOTAL	3182	45.7	57.6	**51.0**	45.8	57.2	**50.9**	48.6	59.0	**53.3**

Table 5: Results on the Shared Task test data. Approximate Span Matching/Approximate Recursive Matching.

only one event profits from trimming of prepositions (*Protein Catabolism*). Only the *Binding* event profits significantly from noun phrase modifications (see Table 3). The increase in F-score for trimming procedures is 4.1 percentage points for *Binding* events.

In our final experiment we connected the best configurations for each of the BioNLP'09 events as presented in Table 2. The overall event extraction results of this final configuration are presented in Tables 4 and 5. We achieved an increase of 1.9 percentage points F-score in the overall event extraction compared to the best-performing single parser configuration (M+C, CoNLL'07) (see Table 4, ALL-TOTAL). The reported results on the development data outperform the results of the TOKYO system by 2.6 percentage points F-score for all basic events including *Binding* events (see Table 4, EVT-TOTAL) and by 0.9 percentage points in the overall event extraction task (see Table 4, ALL-TOTAL).

On the test data we achieved an F-score similar to the current JULIELab system trained on modified CoNLL'07 dependencies from the MST parser (see Table 5, ALL-TOTAL).[18] The results on the official test data reveal that the performance differences between various parsers may play a much smaller role than the proper choice of dependency representations.[19] Our empirical findings that the best performance results could only be achieved by event-specific dependency graph configurations reveal that the syntactic representations of different semantic events vary considerably at the level of dependency graph complexity and that the automatic prediction of such syntactic structures can vary from one dependency parser to the other.

6 Discussion

The evaluation results from Table 1 show that an increased F-score is basically due to a better performance in terms of precision. For example, the M+C evaluation results in the Stanford *basic* mode provide an increased precision by 2 percentage points compared to the Stanford *ccprocessed* mode. Therefore, we focus here on the analysis of false positives that the JULIELab system extracts in various modes.

For the first analysis we took the outputs of the systems based on the M+C parsing results. We scrutinized on the Stanford *basic* and *ccprocessed* false positives (fps) and we compared the occurrences of dependency labels in two data sets, namely the intersection of false positives from both system modes (set *A*) and the false positives produced only by the system with a worse performance (set *B*, *ccprocessed* mode). About 70% of all fps are contained in set *A*. Our analysis revealed that some dependency labels have a higher occurrence in set *B*, e.g., `nsubjpass, prep_on, prep_with, prep_in, prep_for, prep_as`. Some dependency labels occur only in set *B* such as `agent`, `prep_unlike, prep_upon`. It seems that collapsing some prepositions, such as *"with"*, *"in"*, *"for"*, *"as"*, *"on"*, *"unlike"*, *"upon"*, does not have a positive effect on the extraction of argument structures. In a second step, we compared the Stanford *basic* and CoNLL'07 false positive sets. The fps of both systems have an intersection of about 70%. We also compared the intersection of fps between two outputs (set *A*) and the set of additional fps of the system with worse results (Stanford *basic* mode, set *B*). The dependency labels such as `abbrev, dep, nsubj, nsubjpass` have a higher occurrence in set *B* than in set *A*. This analysis renders evidence that the distinction of `nsubj` and `nsubjpass` does not seem to have been properly learned for event extraction.

For the second analysis round we took the outputs of the MST parsing results. As in the previous experiments, we compared false positives from two mode outputs, here the CoNLL'07 mode and the CoNLL'07 modified for *auxiliaries* and *coordinations* mode. The fps have an intersection of 75%. The dependency labels such as `VC, SUBJ, COORD`, and `IOBJ` occur more frequently in the additional false positives from the CoNLL'07 mode than in the intersection of false positives from both system outputs. Obviously, the trimming of auxiliary and coordination structures has a direct positive effect on the argument extraction reducing false positive numbers especially with corresponding dependency labels in shortest dependency paths.

Our analysis of false positives shows that the distinction between active and passive subject labels,

[18]The current JULIELab system uses event-specific trimming procedures on CoNLL'07 dependencies determined on the development data set (see Buyko et al. (2010)).

[19]Trimmed CoNLL dependencies are used in both system configurations.

abbreviation labels, as well as collapsing prepositions in the Stanford dependencies, could not have been properly learned, which consequently leads to an increased rate of false positives. The trimming of auxiliary structures and the subsequent coordination collapsing on CoNLL'07 dependencies has indeed event-specific positive effects on the event extraction.

The main focus of this work has been on the evaluation of effects of different dependency graph representations on the IE task achievement (here the task of event extraction). But we also targeted the task-oriented evaluation of top-performing syntactic parsers. The results of this work indicate that the GENIA-trained parsers, i.e., M+C parser, the MST, MALT and GDep, are a reasonable basis for achieving state-of-the art performance in biomedical event extraction.

But the choice of the most suitable parser should also take into account its performance in terms of parsing time. Cer et al. (2010) and Miyao et al. (2008) showed in their experiments that native dependency parsers are faster than constituency-based parsers. When it comes to scaling event extraction to huge biomedical document collections, such as MEDLINE, the selection of a parser is mainly influenced by its run-time performance. MST, MALT and GDep parsers or the M+C parser with reduced reranking (Cer et al., 2010) would thus be an appropriate choice for large-scale event extraction under these constraints.[20]

7 Conclusion

In this paper, we investigated the role different dependency representations may have on the accomplishment of the event extraction task as exemplified by biological events. Different representation formats (mainly, Stanford *vs.* CoNLL) were then experimentally compared employing different parsers (Bikel, Charniak+Johnson, GDep, MST, MALT), both constituency based (for the derivative dependency mode) as well as dependency based (for the native dependency mode), considering different training scenarios (newspaper *vs.* biology domain).

From our experiments we draw the conclusion that the dependency graph representation has a crucial impact on the level of achievement of IE task requirements. Surprisingly, the CoNLL'X dependencies outperform the Stanford dependencies for four from six parsers. With additionally trimmed CoNLL'X dependencies we could achieve an F-score of 50.9% on the official test data and an F-score of 57.2% on the official development data of the BioNLP Shared Task on Event Extraction (see Table 5, ALL-TOTAL).

[20]For large-scale experiments an evaluation of the M+C with reduced reranking should be provided.

Acknowledgements

This research was partially funded by the BOOTSTREP project under grant FP6-028099 within the 6th Framework Programme (EC), by the CALBC project under grant FP7-231727 within the 7th Framework Programme (EC), and by the JENAGE project under grant 0315581D from the German Ministry of Education and Research (BMBF) as part of the GERONTOSYS funding initiative. We also want to thank Kenji Sagae (Institute for Creative Technologies, University of Southern California) for kindly providing the models of the GDEP parser.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings BioNLP 2009. Companion Volume: Shared Task on Event Extraction*, pages 10–18. Boulder, Colorado, USA, June 4-5, 2009.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on multilingual dependency parsing. In *CoNLL-X – Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164, New York City, N.Y., June 8-9, 2006.

Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2009. Event extraction from trimmed dependency graphs. In *Proceedings BioNLP 2009. Companion Volume: Shared Task on Event Extrac-*

tion, pages 19–27. Boulder, Colorado, USA, June 4-5, 2009.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2010. Syntactic simplification and semantic enrichment - Trimming dependency graphs for event extraction. *Computational Intelligence*, 26(4).

Daniel Cer, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *LREC'2010 – Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 1628–1632. Valletta, Malta, May 19-21, 2010.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *ACL'05 – Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180. Ann Arbor, MI, USA, 25-30, June, 2005.

Andrew B. Clegg and Adrian J. Shepherd. 2005. Evaluating and integrating Treebank parsers on a biomedical corpus. In *Proceedings of the ACL 2005 Workshop on Software*, pages 14–33. Ann Arbor, MI, USA, June 30, 2005.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC'2006 – Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–454. Genoa, Italy, 24-26 May 2006.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA 2007 – Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 105–112. Tartu, Estonia, May 24-25, 2007.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(10).

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 BioNLP 2009. Companion Volume: Shared Task on Event Extraction*, pages 1–9. Boulder, Colorado, USA, June 4-5, 2009.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the LREC'98 Workshop on the Evaluation of Parsing Systems*, pages 48–56. Granada, Spain, 28-30 May 1998.

David McClosky. 2010. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP 2005 – Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 523–530. Vancouver, B.C., Canada, October 6-8, 2005.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology*, 8:131–146.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *HLT 2002 – Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 292–297. San Diego, CA, USA, March 24-27, 2002.

Yusuke Miyao, Kenji Sagae, and Jun'ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAF 2007 Workshop*, CSLI Studies in Computational Linguistics Online, page 21 pages.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *ACL 2008 – Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 46–54. Columbus, Ohio, USA, June 15-20, 2008.

Claire Nédellec. 2005. Learning Language in Logic: Genic interaction extraction challenge. In *Proceedings LLL-2005 – 4th Learning Language in Logic Workshop*, pages 31–37. Bonn, Germany, August 7, 2005.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2007. MALTPARSER: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Jarvinen, and Tapio Salakoski. 2007. BIOINFER: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50).

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007 – Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*, pages 1044–1050, Prague, Czech Republic, June 28-30, 2007.

Daniel Sleator and Davy Temperley. 1991. Parsing English with a link grammar. Technical report, Department of Computer Science, CMU.

Yuka Tateisi, Akane Yakushiji, and Jun'ichi Tsujii. 2005. Syntax annotation for the GENIA corpus. In *IJCNLP 2005 – Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 222–227. Jeju Island, Korea, October 11-13, 2005.