Representing Layered and Structured Data in the CoNLL-ST Format

Pavel Straňák and Jan Štěpánek

Institute of Formal and Applied Linguistics, Charles University in Prague
Malostranské náměstí 25, 118 00 Praha, Czech Republic
E-mail: `{stranak,stepanek}@ufal.mff.cuni.cz`

Abstract

In this paper, we investigate the CoNLL Shared Task format, its properties and possibility of its use for complex annotations. We argue that, perhaps despite the original intent, it is one of the most important current formats for syntactically annotated data. We show the limits of the CoNLL-ST data format in its current form and propose several simple enhancements that push those limits further and make the format more robust and future proof. We analyse several different linguistic annotations as examples of varying complexity and show how they can be efficiently stored in the CoNLL-ST format.

1 Introduction

Despite the efforts to establish a standard for representing linguistically annotated data (cf. Linguistic Annotation Framework, (Ide and Romary, 2004)), none has been adopted by the community yet. Nevertheless, there are formats that are used more often than others and serve more than one purpose. One of them is so called CoNLL Shared Task format ("CoNLL-ST format" for short) designed to encode dependency trees (Hajič et al., 2009). The idea behind it is very simple (which is probably why the format spread): every sentence corresponds to a table whose rows correspond to words while the columns bear additional information about the words. The simplicity of the format is its main drawback at the same time: it limits, or almost disables, the extensibility of the format. Some ways to extend the format are investigated in this article.

We do not claim that the CoNLL-ST format is the universal and generic format to use for all the NLP data. It is rather a good one-purpose exchange format, but it has been used quite a bit even as a substitute of the original data. In fact it has become a sort of interchange standard for treebank data. There are several reasons for this:

- The main users of the raw treebank data are NLP researchers who want to process the data. Typically with tools that accept tables on input and produce tables on output. This is not going to change in the near future, and CoNLL-ST format is precisely such a table that can be easily processed.

- The major treebanks in use today use either their own XML formats, or even other in-house solutions. Such formats are not always easy to validate and parse and tools to do this are not always readily available.

- LAF or any other annotation standard cannot supplant the CoNLL-ST format as the de facto standard for the researchers in parsing and other areas, who need to work with many different treebanks, because it would still need to be processed into a CoNLL-ST -like table in the end and would only bring added complexity.

- Existing treebanks sometimes have additional annotations that are hard to put together in the original formats. Let's take Penn Treebank (Marcus et al., 1993), PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) as examples. In the CoNLL-ST format there is the PennTreebank with NomBank tokenisation and PropBank frames. It has been widely distributed, so it is a de facto standard for those who want to combine these pieces of information.

Moreover, here are several examples of the usage of CoNLL-ST format outside CoNLL-ST environment:

- The data for the ICON 2009 contest in parsing of Indian languages have been released in the original, idiosyncratic format, as well as the CoNLL-ST format.
- A well known researcher asked for the PDT data to test his parser. The answer pointing to the original data was not met with any enthusiasm. He opted for the data from CoNLL-ST 2009 instead.
- CoNLL-ST format is used in the project described in (Dickinson and Ragheb, 2009). The encoding used for lists is surprisingly similar to that proposed in our article.

For these reasons we do not believe CoNLL-ST format can be replaced with any of the more complex standards, at least in the near future. Seeing that it gets even more use we decided to examine its limits and propose a simple extension to make it more robust.

2 Lack of Meta-Information

In fact, there is nothing like a single CoNLL-ST format, there are rather several similar formats, each for one year of the shared task. However, it is not easy (if even possible) to detect the version of a data file: the files have no headers nor meta-information. The number of columns differs from version to version (moreover, it is variable in both years 2008 and 2009) and their meaning differs too.

The easiest way to rectify this is to introduce a *comment character*, for example `#`. The comment character would be allowed only at the beginning of a line, thus making the comment lines easily recognisable from data lines that start with a number, and empty lines that separate sentences. It also makes the conversion to the "comment-less" form simple—just by removing all the comment lines. The first line of a data file might then look like this:

```
# CoNLL-ST-2006
```

which is just a shorthand for this:

```
# ID FORM LEMMA CPOSTAG POSTAG FEATS
HEAD DEPREL PHEAD PDEPREL
```

To define a set of columns (as ARG in 2008 and APRED in 2009 Shared Task), some meta-character must be used, since the number of columns is variable. For example, "+" could be used:

```
# ID FORM LEMMA PLEMMA POS PPOS FEAT
PFEAT HEAD PHEAD DEPREL PDEPREL
FILLPRED PRED APRED+
```

Note that "+" can be used just once on a line: if there were two sets of columns of variable length, there would be no way how to tell which column belongs to a set and which does not when parsing the data. The set, on the other hand, does not have to be the last in the list, but if it is not, the columns following it must be parsed right-to-left. See Section 3 on how to represent several lists in the CoNLL-ST format.

The comment character can be used for a wider range of purposes, though. Let us imagine we need to refer to surrounding sentences to annotate some context information (e.g. coreference). We can either use integers (zero for the current sentence, negative numbers for preceding sentences, and positive numbers for the following ones) or sentence identifiers. The latter approach is more robust—if the sentences are to be shuffled or some of them removed, the identifier reference would still point to the same target (or none if it was removed) while in case of integers the references would get muddled. Each sentence can be assigned an identifier just by prepending this line to its representation:

```
# ID S134
```

If the comment character is followed by unrecognised text, it is considered a proper comment.

3 Lists

There already are two ways how to represent a list in the CoNLL-ST format. The first one was introduced in CoNLL-ST 2006 for the FEATS column (morphological features), it uses vertical bar "|" to separate members of the list. The format of a member was not specified, but we can find two possibilities in the data: either the members are just values (“sg”), or pairs of a name and value (“num=sg”).[1]

The second way of representing a list was introduced in CoNLL-ST 2008: each member corresponds to a column (the “+” notation in Section 2). Note that such a list of columns can be used just once on a line (cf. ibid.) But the impossibility to have more than one list is not the only disadvantage of this approach.

In CoNLL-ST 2008, the first column in the list corresponds to the first predicate, the second column to the second predicate, and so on. The number of columns of a sentence thus corresponds to the number of predicates in the sentence. Thanks to this semantics of the columns, the number of columns in the data is the same for all the lines of a sentence. Different sentences, though, can have a different number of columns. As a consequence, the columns are rather “sparse”, but their number is high. In CoNLL-ST 2009 data, the “widest” sentence in the English data has 36 columns (25 of them making the list) and the widest Czech sentence has 138 columns (127 in the list). However, 99.3% of the Czech predicates and 93.7% of the English predicates have none or one argument. To make the data denser and to lower the number of columns, a different approach has to be chosen: for every predicate, its arguments would form a list on the argument’s line. The maximal number of arguments is 10 in the English data and just 6 in the Czech data. Moreover, the list of arguments can be represented in one column with vertical bar as a separator, making the table even thinner. See the difference between Tables 1 and 2.

When using columns to represent list members, one can even arrive to a situation where the number of columns will be different for each line. This is not possible with lists separated by vertical bar, which in our opinion makes them the option to choose.

Lists of n-tuples can be represented by n columns, the first column containing the (“|”-separated) list of first members of the tuples, the second column containing the list of sencond members of the tuples and so on. The n has to be fixed and known beforehand. For $n = 2$, just one column can be used if the FEATS syntax is adopted (see also Quotes in Section 4.1.2).

In order to represent nested lists (i.e. lists of lists), we can add grouping characters. For example, this is a list of three lists, of which the first has one member, the second one is empty and the third one has two members:

```
<sem1=ACT>|<>|<sem1=PAT|sem2=EFF>
```

Any other pair of characters can be picked up to group lists, there should also be a “quoting” character to allow including of the grouping character into lists in their literal meaning. Quoting character would be also useful for including the underscore (“_”) as a literal, i.e. not denoting the empty string. All the special characters can be also (re-)defined in the header of the file on the second comment line… Let us stop here. This way we are going to change CoNLL-ST format into something like reinvented SGML. It would be so complex, unreadable, and would need so much documentation, that it is arguably better to use the original format that was designed to represent this information in the first place. Such format is usually well documented, can be validated, has a handful of existing tools, etc.

The CoNLL-ST format is popular mainly for its simplicity. If we do not keep that we get just another arbitrary format with no benefit over the others.

4 A Problem of Units

Most treebanks have one “basic” layer that defines the basic units, their attributes and relations between them. Other annotations are then “additional”, meaning they are defined in terms of these “basic” units and usually do not cover the whole sentence. There may however be treebanks with several distinct layers of annotation that each cover the whole sentence but their units differ. In the most interesting (i.e. worst) case the mapping of units of different layers is M to N.

CoNLL-ST format has only one “layer”, which requires us to choose one layer of annotation as “basic”, setting the units of CoNLL-ST representation. Then the problem of mapping the units, attributes and relations of other layers to these CoNLL-ST units needs to be solved.

[1]The pairs can still be treated as single values if there is no need to parse them, or even the whole list can be treated as a single value.

ID	FORM	PRED	APRED1	APRED2	APRED3	APRED4	APRED5	APRED6
1	A	_	_	_	_	_	_	_
2	widening	widening.01	_	_	A0	_	_	_
3	of	_	A1	_	_	_	_	_
4	the	_	_	_	_	_	_	_
5	deficit	_	_	_	_	_	_	_
6	,	_	_	_	_	_	_	_
7	if	_	_	_	AM-ADV	_	_	_
8	it	_	_	A1	_	_	_	_
9	were	_	_	_	_	_	_	_
10	combined	combine.01	_	_	_	_	_	_
11	with	_	_	A2	_	_	_	_
12	a	_	_	_	_	_	_	_
13	stubbornly	_	_	_	_	_	_	_
14	strong	_	_	_	_	_	_	_
15	dollar	_	_	_	_	_	_	_
16	,	_	_	_	_	_	_	_
17	would	_	_	_	AM-MOD	_	_	_
18	exacerbate	exacerbate.01	_	_	_	_	_	_
19	trade	_	_	_	_	A2	_	_
20	problems	problem.01	_	_	A1	_	_	_
21	–	_	_	_	_	_	_	_
22	but	_	_	_	_	_	_	_
23	the	_	_	_	_	_	_	_
24	dollar	_	_	_	_	_	A1	_
25	weakened	weaken.01	_	_	_	_	_	_
26	Friday	_	_	_	_	_	AM-TMP	_
27	as	_	_	_	_	_	AM-ADV	_
28	stocks	_	_	_	_	_	_	A1
29	plummeted	plummet.01	_	_	_	_	_	_
30	.	_	_	_	_	_	_	_

Table 1: Predicates and arguments in CoNLL-ST 2008 format.

4.1 Representing PDT 2.0 in CoNLL-ST Format

We decided to illustrate the problem of basic units on an example of Prague Dependency Treebank 2.0 (PDT, (Hajič et al., 2006)), because it contains two layers that each contains a different tree for each sentence and the units on each of these layers differ. **A-layer** (analytical, "shallow syntax") trees are relatively simple: each node in a tree corresponds to one token in the (tokenised) sentence. **T-layer** (tectogrammatical, "deep syntax", (Mikulová et al., 2006)) contains trees in which nodes might represent several tokens of a sentence (and a-layer, e.g. compound verbs, preposition + nount etc.) as well as nodes that do not have any corresponding token in the sentence (and a-layer, e.g. dropped subjects, elided words). Thus the mapping of a-nodes to t-nodes is M to N.

4.1.1 A-layer as the Starting Point

The choice of a-layer as the starting point gives a familiar CoNLL-ST table in which we can read the sentence in the second column (see Table 3). It is a good solution in many aspects: For one it is easily readable, because the lines correspond to tokens and they keep the surface word order, so the sentence can be easily read. This representation is a matter of course for the treebanks that only contain one syntactic layer. For PDT, though, it is not so simple.

Representing the information from t-layer however poses a serious problem: where should be put the information related to t-nodes that do not have any equivalent on the surface of a sentence, i.e. they have no lines in the CoNLL-ST table. Common examples of this type of nodes are dropped subjects of many Chinese, Czech, or Japanese sentences, and other zero anaphoras.[2]

We can solve this problem by resolving the coreference and using the "original" node. That poses problems of its own though: there is often a coreference chain, so we must decide whether to use the closed antecedent, or the first. Then we must refer to it (e.g. as a subject of a predicate, as mentioned above). How to refer to a node that is not in the same sentence is discussed in Section 2. In the current CoNLL-ST format it would mean that sentences can not be shuffled or simply cut from files without losing information.

Also this solution would result in overloading a "resolved" node with information: it could be a member of quite a few valency frames in several roles.

We could talk about other deficiencies of this solution, for instance loosing a distinction be-

[2]An English example of zero anaphora: "There are two ways: the short [] and the long []."

ID	FORM	PRED	ARGS
1	A	_	_
2	widening	widening.01	A1=3
3	of	_	_
4	the	_	_
5	deficit	_	_
6	,	_	_
7	if	_	_
8	it	_	_
9	were	_	_
10	combined	combine.01	A1=8 | A2=11
11	with	_	_
12	a	_	_
13	stubbornly	_	_
14	strong	_	_
15	dollar	_	_
16	,	_	_
17	would	_	_
18	exacerbate	exacerbate.01	A0=2|AM-ADV=7|AM-MOD=17|A1=20
19	trade	_	_
20	problems	problem.01	A2=19
21	–	_	_
22	but	_	_
23	the	_	_
24	dollar	_	_
25	weakened	weaken.01	A1=24|AM-TMP=26|AM-ADV=27
26	Friday	_	_
27	as	_	_
28	stocks	_	_
29	plummeted	plummet.01	A1=28
30	.	_	_

Table 2: Predicates and arguments in the modified denser CoNLL-ST format.

ID	FORM	LEMMA	POS	HEAD	DEPREL	PRED	APRED1	APRED2	APRED3
1	Neither	neither	DT	3	AuxY	_	_	_	_
2	they	they	PRP	3	Sb.member	_	_	PAT	_
3	nor	nor	CC	6	Coord	_	_	_	_
4	Mr.	Mr.	NNP	5	Atr	_	RSTR	_	_
5	McAlpine	McAlpine	NNP	3	Sb.member	McAlpine	_	PAT	_
6	could	can	MD	0	Pred	_	_	_	_
7	be	be	VB	8	AuxV	_	_	_	_
8	reached	reach	VBN	6	Obj	reach	_	_	_
9	for	for	IN	8	AuxP	_	_	_	_
10	comment	comment	NN	9	Adv	comment	_	AIM	_
11	.	.	PUNC	0	AuxK	_	_	_	_

Table 3: *Neither they nor Mr. McAlpine could be reached for comment.* Sentence represented in a format similar to Czech CoNLL-ST 2009 (some columns removed to save space).

tween t-nodes that can be somehow resolved via coreference and duplicated nodes, that cannot (in the English zero-anaphora example). We believe that it is quite clear that CoNLL-ST format based on the surface layer units is suitable for relatively simple data that above else do not use additional ("newly established") nodes in trees. Once these are used, the representation should be different.

4.1.2 T-layer as the Starting Point

Starting the CoNLL-ST representation with t-layer does not provide us with such a straightforward representation of the original sentence. On the other hand it provides means to accurately represent significantly more information from t-layer while still keeping everything from a-layer, even though in somewhat modified representation.[3]

If we base the format on t-layer, we take its units and represent them and their associated information just like we do it with token-based CoNLL-ST format currently. The result is illustrated by Table 4. Compare it with Table 3. The example sentence was extracted from PDT-like Prague Czech-English Dependency Treebank (still in development)

[3]Keeping everything from both (or generally *all*) layers is of course possible, but the result would become too complex, as illustrated in Section 4.2.

ID	T-LEMMA	HEAD	SEMREL	A-LEMMA	A-AUX	A-FORM	COREF-TEXT
1	#PersPron	3	PAT.member	they	_	they	-1=4
2	neither_nor	2	CONJ	nor	neither	neither_nor	_
3	mr.	4	RSTR	Mr.	_	Mr.	_
4	mcalpine	2	PAT.member	McAlpine	_	McAlpine	_
5	reach	0	PRED	reach	can_be	could_be_reached	_
6	#Gen	5	ACT	_	_	_	_
7	comment	5	AIM	comment	for	for_comment	_

Table 4: *Neither they nor Mr. McAlpine could be reached for comment.* Sentence represented in a format based on tectogrammatical layer (many columns removed to save space).

by PML-TQ, the query engine for treebanks in PML format. The query is shown in Figure 1.

```
t-node $n4 :=
[ ancestor t-root [  ] ];

t-root
[ descendants() < 10,
  t-node [ coref_text.rf $n4 ],
  t-node [ bbn_tag ~ '.' ],
  t-node [ is_generated = 1 ],
  t-node [ a/aux.rf a-node [  ],
    a/aux.rf a-node [  ] ] ];
```

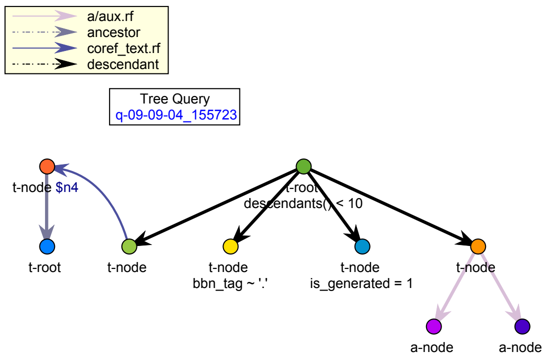

Figure 1: Query in PML-TQ used to find the sentence from Table 3.

It is basically just like the format of CoNLL-ST 2009, except now we do not need the APRED columns. Semantic role labels are in the column SEMREL and their dependency structure is defined by the HEAD column. Most attributes and relations of t-layer can be represented in equally straightforward way, with a few notable exceptions:

Coreference can lead to another sentence. Either relative indices, or sentence IDs are required, as explained in Section 2.

Quotes form subtrees and can be embedded in other quotes. The quotes (sets of nodes) are also typed. The easiest way to represent a set of typed sets is a column containing a list of pairs:

```
1 = typeA | 2 = typeD | .. | N = typeM
```

Bridging Anaphora[4] is a relation between subtrees, possibly in different sentences. It is thus similar to quotes, with two differences: a) the set is identified by its first node (no node is a part of two or more sets). b) There are relations between the sets. So one more column is needed, that contains for each node that represents a set an ID of a node that represents its antecedent. Again, if the antecedent is in a different sentence, it must be marked either with a relative index, or with a sentence ID.

Named Entities[5] are contiguous subtrees just like multiword lexemes (only in dependency trees, of course). They can in fact be considered a special type of multiword expressions. They have the same properties, so they are also best identified on t-layer. What makes them more interesting from the point of annotation is their hierarchical structure, for instance a personal name can be composed like this:

$$\text{(title ((first name + middle name) +}$$
$$\text{(last name + maiden name)))}$$

This hierarchy needs to be represented either by lists of list as defined in Section 3, or in the manner used in CoNLL-ST until 2005 (Carreras and Màrquez, 2005) for representation of non-terminals of phrase trees. Neither is very elegant and easily readable.

The most obvious drawback of this representation is that we can not simply represent the surface syntactic structure, since some units, like prepositions or auxiliaries, are not represented by a node.

[4]Not present in published PDT 2.0. Currently being annotated.

[5]Not present in published PDT 2.0, either.

The other drawback is also apparent: even the t-layer itself, with its typed sets of nodes and relations above these sets, requires complex representation. That means also complex documentation, validators, parsers, etc.

4.2 Two or More Layers of Annotation in CoNLL-ST

We have shown that representing several layers of annotation based on different units is not straightforward in the CoNLL-ST format. Either we have to drop some information, or it must be represented in a complex way that is not easy to understand and even harder to validate, parse and use.

For instance, using the examples from Section 4.1.1 and Section 4.1.2 above, let us see both approaches:

- **The lossy approach** has been used in CoNLL-ST 2009. There were only two things from t-layer used: semantic role labels and their dependency structures (argument structures). Even so, the "newly established" nodes were simply dropped and all information on them was lost. See for example the node with `ID = 6` and lemma `#Gen`, stating it is a generic actor, from Table 4 and Figure 2.[6]

  Coreferences have also been considered for CoNLL-ST 2009 (although they have been dropped in the end). Their representation would be again lossy, as described in Section 4.1.1.

  Although lossy, the approach of CoNLL-ST 2009 is also a bit complex and hard to read because of the use of APRED columns but that can be improved by using lists, as we have shown in Section 3.

- **The lossless but complex approach** can be executed in several ways. The easiest would be to use t-layer as the starting point but using lists for A-LEMMA and A-AUX (see Table 4), that would contain DEPREL as values, and then referring to the list members in a SYNHEAD column.

[6]This is true also for Penn Treebank (Marcus et al., 1993) and possibly other treebanks: see the trace `*-38` in Figure 2 corresponding to the newly established t-node `#Gen`.

5 Discussion

We can see that it is *possible* to represent all information from a complex treebank with several layers of annotation in the CoNLL-ST format. But the way to do it is arguably neither simpler, nor very easy to read, and it certainly is not easier to parse or validate, than using XML. So it seems to deny the reason, why the CoNLL-ST format was invented and became popular in the first place.

Yet at the same time after many years of having the PDT 2.0 in a well documented XML format we can say that to our knowledge nobody has ever used the data format in all its complexity. Nobody wants to spend time learning complex data formats and adapting their tools to work with them. If someone really has to use the original XML format, they only parse as little as possible, effectively ignoring much information. Possibly much more than is present in the current, by no means perfect, CoNLL-ST format.

Using a new generic XML annotation standard may be a good solution for some applications but not for others. In the long run, it should be beneficial for annotation tools, query engines, and other "user" applications. When a standardised format is able to represent all the treebanks in the complexity of theories they follow and when there is support of the standard in the annotation tools, there will be treebanks converted into such a standard or produced in it.

On the other hand for the machine learning toolkits and many other "programming" uses of data by researchers, the "table" is the way to go and anything else ends up converted into a table in the end. Thus it is in the interest of data providers to provide a good, standardised "table format", so that their data are used in the research as much as possible and the research is comparable. It is perhaps not crucial to export each and every attribute of the original data into this format, but it is possible to represent quite a bit in a surprisingly simple table. In many respects simpler than in CoNLL-ST 2009.

6 Conclusion

We argue that while the linguistic annotation standards like LAF might be useful, they have

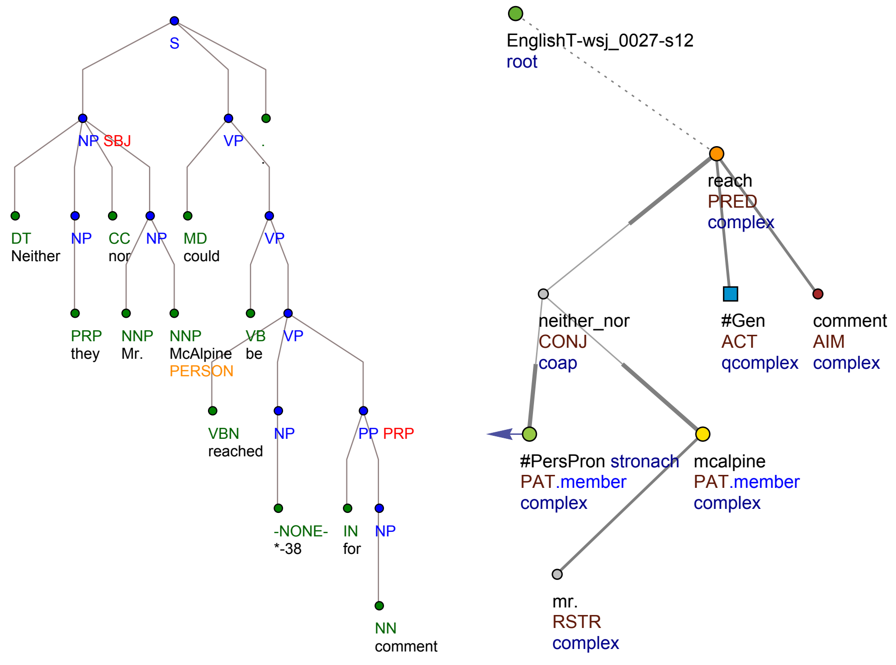

Figure 2: Phrasal and tectogrammatical tree of the English sentence from Tables 3 and 4.

a different use than CoNLL-ST format. We show why CoNLL-ST is the format of choice of the research community.

We point to its weakest points and propose simple enhancements that make it more readable, easier to parse and allow to represent more complex annotations. We demonstrate representation of these complex annotations on several examples.

The idea that complex linguistic annotation can be fully represented in a generic simple data format is naïve. To make the data simple, one must sacrifice complexity, losing some details or pieces of information (e.g. the case of PDT data at CoNLL-ST 2009). But this is not necessarily a bad thing, as long as the data format evolves in details to suite the needs of target applications as has been the case of CoNLL-ST format.

We have hopefully shown that the CoNLL-ST format is highly suited for its original purpose: simplified representation of tree structures with flat attributes for machine-learning toolkits. And with small additions that do not compromise its simplicity (notably the comment character, header line, and sentence identifiers) it can well incorporate additional relations, even between sentences, and be also more readable and flexible. May it serve many future tasks.

Acknowledgement

This paper was written with the support of post-doc grant 406/10/P193 of Grant Agency of the Academy of Science of the Czech Republic, grant FP7-ICT-2007-3-231720 (EuroMatrix Plus) of the Seventh Framework Programme of the European Union, grant 1ET201120505 of Grant Agency of the Academy of Science of the Czech Republic, and grant GAUK 4307/2009 of the Grant Agency of Charles University in Prague.

References

[Carreras and Màrquez2005] Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics. 4.1.2

[Dickinson and Ragheb2009] Markus Dickinson and Marwa Ragheb. 2009. Dependency annotation for learner corpora. In *Proceedings of the Eighth International Workshop on Treebanks* *and Linguistic Theories (TLT8)*, pages 59–70, Milan, December. EDUCatt. 1

[Hajič et al.2009] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi 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-2009), June 4-5*, Boulder, Colorado, USA. 1

[Hajič et al.2006] Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. 2006. Prague Dependency Treebank 2.0. CD-ROM. Linguistic Data Consortium. 4.1

[Ide and Romary2004] Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Natural Language Engineering*, 10(3-4):211–225. 1

[Marcus et al.1993] 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. 1, 6

[Meyers et al.2004] A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The Nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics. 1

[Mikulová et al.2006] Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. 2006. Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep. 4.1

[Palmer et al.2005] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, March. 1