Abstract

This submission to the open track of Task 8 at SemEval 2014 seeks to connect the Task to pre-existing, ‘in-house’ parsing systems for the same types of target semantic dependency graphs.

1 Background and Motivation

The three target representations for Task 8 at SemEval 2014, Broad-Coverage Semantic Dependency Parsing (SDP; Oepen et al., 2014), are rooted in language engineering efforts that have been under continuous development for at least the past decade. The gold-standard semantic dependency graphs used for training and testing in the Task result from largely manual annotation, in part re-purposing and adapting resources like the Penn Treebank (PTB; Marcus et al., 1993), PropBank (Palmer et al., 2005), and others. But the groups who prepared the SDP target data have also worked in parallel on automated parsing systems for these representations.

Thus, for each of the target representations, there is a pre-existing parser, often developed in parallel to the creation of the target dependency graphs, viz. (a) for the DM representation, the parser of the hand-engineered LinGO English Resource Grammar (ERG; Flickinger, 2000); (b) for PAS, the Enju parsing system (Miyao, 2006), with its probabilistic HPSG acquired through linguistic projection of the PTB; and (c) for PCEDT, the scenario for English analysis within the Treex framework (Popel and Žabokrtský, 2010), combining data-driven dependency parsing with hand-engineered tectogrammatical conversion. At least for DM and PAS, these parsers have been extensively engineered and applied successfully in a variety of applications, hence represent relevant points of comparison. Through this ‘in-house’ submission (of our ‘own’ parsers to our ‘own’ task), we hope to facilitate the comparison of different approaches submitted to the Task with this pre-existing line of parser engineering.

2 DM: The English Resource Grammar

Semantic dependency graphs in the DM target representation, DELPH-IN MRS-Derived Bi-Lexical Dependencies, stem from a two-step ‘reduction’ (simplification) of the underspecified logical-form meaning representations output natively by the ERG parser, which implements the linguistic framework of Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994). Gold-standard DM training and test data for the Task were derived from the manually annotated DeepBank Treebank (Flickinger et al., 2012), which pairs Sections 00–21 of the venerable PTB Wall Street Journal (WSJ) Corpus with complete ERG-compatible HPSG syntactico-semantic analyses. DeepBank as well as the ERG rely on Minimal Recursion Semantics (MRS; Copestake et al., 2005) for meaning representation, such that the exact same post-processing steps could be applied to the parser outputs as were used in originally reducing the gold-standard MRs from DeepBank into the SDP bi-lexical semantic dependency graphs.

Parsing Setup The ERG parsing system is a hybrid, combining (a) the hand-built, broad-coverage ERG with (b) an efficient chart parser for unification grammars and (c) a conditional probability distribution over candidate analyses. The parser most commonly used with the ERG, called PET (Callmeier, 2002), constructs a complete,
subsumption-based parse forest of partial HPSG derivations (Oepen and Carroll, 2000), and then extracts from the forest n-best lists (in globally correct rank order) of complete analyses according to a discriminative parse ranking model (Zhang et al., 2007). For our experiments, we trained the parse ranker on Sections 00–20 of DeepBank and otherwise used the default, non-pruning development configuration, which is optimized for accuracy. In this setup, ERG parsing on average takes close to ten seconds per sentence.

Post-Parsing Conversion After parsing, MRSs are reduced to DM bi-lexical semantic dependencies in two steps. First, Oepen and Lönning (2006) define a conversion to variable-free Elementary Dependency Structures (EDS), which (a) maps each predication in the MRS logical-form meaning representation to a node in a dependency graph and (b) transforms argument relations represented by shared logical variables into directed dependency links between graph nodes. This first step of the conversion is ‘mildly’ lossy, in that some scope-related information is discarded; the EDS graph, however, will contain the same number of nodes and the same set of argument dependencies as there are predications and semantic role assignments in the original MRS. In particular, the EDS may still reflect non-lexical semantic predications introduced by grammatical constructions like covert quantifiers, nominalization, compounding, or implicit conjunction.

Second, in another conversion step that is not information-preserving, the EDS graphs are further reduced into strictly bi-lexical form, i.e. a set of directed, binary dependency relations holding exclusively between lexical units. This conversion is defined by Ivanova et al. (2012) and seeks to (a) project some aspects of construction semantics onto word-to-word dependencies (for example introducing specific dependency types for compounding or implicit conjunction) and (b) relate the linguistically informed ERG-internal tokenization to the conventions of the PTB. Seeing as both is called the LOGON SVN trunk as of January 2014; see http://moin.delph-in.net/LogonTop for detail.

Conversely, semantically vacuous parts of the original input (e.g. infinitival particles, complementizers, relative pronouns, argument-marking prepositions, auxiliaries, and most punctuation marks) were not represented in the MRS in the first place, hence have no bearing on the conversion.

Adaptations of tokenization encompass splitting ‘multi-word’ ERG tokens (like such as or ad hoc), as well as ‘hiding’ ERG token boundaries at hyphens or slashes (e.g. 77-year-old), which the PTB does not split.

Abstractly similar to the ERG, the annotations of the Enju treebank instantiate the linguistic theory of HPSG. However, the two resources have been developed independently and implementation details are quite different. The most significant difference is that the Enju HPSG treebank is developed by linguistic projection of PTB annotations, and the Enju parser derived from the treebank; conversely, the ERG was predominantly manually crafted, and it was later applied in the DeepBank re-annotation of the WSJ Corpus.

3Adaptations of tokenization encompass splitting ‘multi-word’ ERG tokens (like such as or ad hoc), as well as ‘hiding’ ERG token boundaries at hyphens or slashes (e.g. 77-year-old), which the PTB does not split.

3See http://kmcs.nii.ac.jp/enju/. 4Abstractly similar to the ERG, the annotations of the Enju treebank instantiate the linguistic theory of HPSG. However, the two resources have been developed independently and implementation details are quite different. The most significant difference is that the Enju HPSG treebank is developed by linguistic projection of PTB annotations, and the Enju parser derived from the treebank; conversely, the ERG was predominantly manually crafted, and it was later applied in the DeepBank re-annotation of the WSJ Corpus.

Full details of the predicate–argument structures in the Enju HPSG Treebank, are available in two documents linked from the Enju web site (see above), viz. the Enju Output Specification Manual and the XML Format Documentation.

3Conversely, semantically vacuous parts of the original input (e.g. infinitival particles, complementizers, relative pronouns, argument-marking prepositions, auxiliaries, and most punctuation marks) were not represented in the MRS in the first place, hence have no bearing on the conversion.

3Adaptations of tokenization encompass splitting ‘multi-word’ ERG tokens (like such as or ad hoc), as well as ‘hiding’ ERG token boundaries at hyphens or slashes (e.g. 77-year-old), which the PTB does not split.
** Parsing Setup** Basically we used the publicly available package of the Enju parser ‘as is’ (see the above web site). We did not change default parsing parameters (beam width, etc.) and features. However, the release version of the Enju parser is trained with the HPSG treebank corresponding to the Penn Treebank WSJ Sections 2–21, which includes the test set of the Task (Section 21). Therefore, we re-trained the Enju parser using Sections 0–20, and used this re-trained parser in preparing the PAS semantic dependency graphs in this ensemble submission.

** Post-Parsing Conversion** The dependency format of the Enju parser is almost equivalent to what is provided as the PAS data set in this shared task. Therefore, the post-parsing conversion for the PAS data involves only formatting, viz. (a) format conversion into the tabular file format of the Task; and (b) insertion of dummy relations for punctuation tokens ignored in the output of Enju.

4 PCEDT: The Treex Parsing Scenario

The *Prague Czech-English Dependency Treebank* (PCEDT; Hajic et al., 2012) is a set of parallel dependency trees over the same WSJ texts from the Penn Treebank, and their Czech translations. Similarly to other treebanks in the Prague family, there are two layers of syntactic annotation: analytical (a-trees) and tectogrammatical (t-trees). Unlike for the other two representations used in the Task, for PCEDT there is no pre-existing parsing system designed to deliver the full scale of annotations of the SDP gold-standard data. The closest available match is a parsing scenario implemented in the Treex natural language processing framework.

** Parsing Setup** Treex (Popel and Žabokrtský, 2010) is a modular, open-source framework originally developed for transfer-based machine translation. It can accomplish any NLP-related task by sequentially applying to the same piece of data various blocks of code. Blocks operate on a common data structure and are chained in scenarios.

Some early experiments with scenarios for tectogrammatical analysis of English were described by (Klimeš, 2007). It is of interest that they report

---

7The Enju parser ignores tokens tagged as ‘.’; while the PAS representation includes them with dummy relations; thus, missing periods are inserted in post-processing by comparison to the original PTB token sequence.


---

Figure 1: PCEDT asserts two copies of the token *regulate* (shown here as ‘regulate’ and ‘ε’ and underlined). Projecting t-nodes onto the original tokens, required by the SDP data format, means that the ε node will be merged with *regulate*. The edges going to and from ε will now lead to and from *regulate* (see the dotted arcs), which results in a cycle. To get rid of the cycle, we skip ε and connect directly its children, as shown in the final SDP graph below the sentence.

The F1 score of assigning *functors* (dependency labels in PCEDT terminology) of 70.3%; however, their results are not directly comparable to ours.

Due to the modular nature of Treex, there are various conceivable scenarios to get the t-tree of a sentence. We use the default scenario that consists of 48 blocks: two initial blocks (reading the input), one final block (writing the output), two A2N blocks (named entity recognition), twelve W2A blocks (dependency parsing at the analytical layer) and 31 A2T and T2T blocks (creating the t-tree based on the a-tree).

Most blocks are highly specialized in one particular subtask (e.g. there is a block just to make sure that quotation marks are attached to the root of the quoted subtree). A few blocks are responsible for the bulk of the work. The a-tree is constructed by a block that contains the MST Parser (McDonald et al., 2005), trained on the CoNLL 2007 English data (Nivre et al., 2007), i.e. Sections 2–11 of the PTB, converted to dependencies. The annotation style of CoNLL 2007 differs from PCEDT 2.0, and thus the unlabeled attachment score of the analytical parser is only 66%.

Obviously we could expect better results if we retrained the MST Parser directly on the PCEDT a-trees, and on the whole training data. The only reason why we did not do so was lack of time. Our results thus really demonstrate what is available “off-the-shelf”; on the other hand, the PCEDT component of our system fails to set any “upper bound” of output quality, as it definitely is not bet-
Fungucrription is done heuristically, based on POS tags and function words. The primary focus of the scenario was on functors that could help machine translation, thus it only generated 25 different labels (of the total set of 65 labels in the SDP gold-standard data) and left about 12% of all nodes without functors.

The system was able to output the following functors (ordered in the descending order of their frequency in the system output): RSTR, PAT, ACT, CONJ.m, APP, MAPP, LOCA, TVHEN, DISJM, BE, RHEM, PREC, ACMP, MEANS, ADVS.m, CPM, EXT, DIR3, CAUS, COND, TSN, REG, DIR2, CNCS, TILL. The system’s precision peaks at ACT relations (78%), while the most frequent error type (besides labelless dependencies) is a falsely proposed RSTR relation. Both ACT and RSTR are among the most frequent dependency types in PCEDT.

Post-Parsing Conversion Once the t-tree has been constructed, it is converted to the PCEDT target representation of the Task, using the same conversion code that was used to prepare the gold-standard SDP data.10

SDP graphs are defined over surface tokens but the set of nodes of a t-tree need not correspond one-to-one to the set of tokens. For example, there are no t-nodes for punctuation and function words (except in coordination); these tokens are rendered as semantically vacuous in SDP, i.e. they do not participate in edges. On the other hand, t-trees can contain generated nodes, which represent elided words and do not correspond to any surface token. Most generated nodes are leaves and, thus, can simply be omitted from the SDP graphs. Other generated nodes are copies of normal nodes and they are linked to the same token to which the source node is mapped. As a result, one token can appear at several different positions in the tree; if we project these occurrences into one node, the graph will contain cycles. We decided to remove all generated nodes causing cycles. Their children are attached to their parents and inherit the functor of the generated node (Figure 1). The conversion procedure also removes cycles caused by more fine-grained tokenization of the t-layer.

Furthermore, t-trees use technical edges to capture paratactic constructions where the relations are not ‘true’ dependencies. The conversion procedure extracts true dependency relations: Each conjunct is linked to the parent or to a shared child of the coordination. In addition, there are also links from the conjunction to the conjuncts and they are labeled CONJ.member. These links preserve the paratactic structure (which can even be nested) and the type of coordination. See Figure 2 for an example.

5 Results and Reflections

Seeing as our ‘in-house’ parsers are not directly trained on the semantic dependency graphs provided for the Task, but rather are built from additional linguistic resources, we submitted results from the parsing pipelines sketched in Sections 2 to 4 above to the open SDP track. Table 1 summarizes parser performance in terms of labeled and unlabeled F1 (LF and UF)11 and full-sentence exact match (LM and UM), comparing to the best-performing submission (dubbed Priberam; ?, ?) to this track. Judging by the official SDP evaluation metric, average labeled F1 over the three representations, our ensemble ranked last among six participating teams; in terms of unlabeled average F1, the ‘in-house’ submission achieved the fourth rank.

As explained in the task description (Oepen et al., 2014), parts of the WSJ Corpus were excluded from the SDP training and testing data because of gaps in the DeepBank and Enju treebanks, and
to exclude cyclic dependency graphs, which can sometimes arise in the DM and PCEDT conversions. For these reasons, one has to allow for the possibility that the testing data is positively biased towards our ensemble members. But even with this caveat, it seems fair to observe that the ERG and Enju parsers both are very competitive for the DM and PAS target representations, respectively, specifically so when judged in exact match scores. A possible explanation for these results lies in the depth of grammatical information available to these parsers, where DM or PAS semantic dependency graphs are merely a simplified view on the complete underlying HPSG analyses. These parsers have performed well in earlier contrastive evaluation too (Miyao et al., 2007; Bender et al., 2011; Ivanova et al., 2013; inter alios).

Results for the Treebank English parsing scenario, on the other hand, show that this ensemble member is not fine-tuned for the PCEDT target representation; due to the reasons mentioned above, its performance even falls behind the shared task baseline. As is evident from the comparison of labeled vs. unlabeled $F_1$ scores, (a) the PCEDT parser is comparatively stronger at recovering semantic dependency structure than at assigning labels, and (b) about the same appears to be the case for the best-performing Priberam system (on this target representation).

**Acknowledgements**

This project has been supported by the infrastructural funding by the Ministry of Education, Youth and Sports of the Czech Republic (CEP ID LM2010013).

---

<table>
<thead>
<tr>
<th></th>
<th>DM</th>
<th>PAS</th>
<th>PCEDT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LF</td>
<td>UM</td>
<td>LF</td>
</tr>
<tr>
<td>Priberam</td>
<td>8916</td>
<td>2685</td>
<td>9176</td>
</tr>
<tr>
<td>In-House</td>
<td>9246</td>
<td>4807</td>
<td>9206</td>
</tr>
<tr>
<td>Priberam</td>
<td>9032</td>
<td>.2990</td>
<td>9281</td>
</tr>
<tr>
<td>In-House</td>
<td>9349</td>
<td>.5230</td>
<td>9317</td>
</tr>
</tbody>
</table>

Table 1: End-to-end ‘in-house’ parsing results.

---

References


of Korea.


