in the treebanks consists of 36 clauses and the 0B1B2B3B4B5B6 clause chart is a chart with the highest number of layers of embedding.

	0	0B1	0B0	0B1B0	0B1B2
Rel. freq.	50.1	14.5	8.0	3.6	2.5
PDT train	93.6	95.8	92.9	92.9	95.9
PDT dtest	85.7	88.2	82.3	81.7	90.0
PDT etest	85.4	88.0	83.4	82.0	88.1
CAC 2.0	84.1	85.7	81.0	79.7	87.3

Table 2: Relative frequency of the five most frequent clause charts in PDT 3.0 and CAC 2.0 (*Rel. freq.*) and the unlabeled attachment score of MST evaluated on the particular subsets PDT train, PDT dtest, PDT etest, CAC 2.0.

5 Methods and Experiments

We present a method for improving dependency parsing of long sentences. In particular, we formulate an algorithm for parsing the two most frequent clause structures, namely coordinated clauses 0B0 and governing and dependent clauses 0B1. The other types of clause structures are processed as usual using full-scale parsing. The experiments exploit an existing dependency parser trained on complete sentences, namely the MST parser – see Section 3 for details.

5.1 Parsing Coordinated Clauses

Given the clause chart representation, we can recognize coordinated clauses in sentences in a straightforward way. Thus, we consider neighboring coordinated clauses C_1, C_2, ..., C_n on the same layer ($n > 1$) and we propose the following parsing strategy that we call *clause chart parsing* (CCP):

1. Using MST parse C_1, C_2, ..., C_n individually to get dependency trees T_1, T_2, ..., T_n with the r_1, r_2, ..., r_n root nodes, respectively.
2. Create a sequence $S = r_1\ B_{1,2}\ r_2\ B_{2,3}\ \ldots r_n$ where $B_{i,i+1}$ is a boundary between C_i and C_{i+1}.
3. Using MST parse the sequence S to get a dependency tree T_S.
4. Build a final dependency tree so that the trees $T_1, \ldots, T_n$ become subtree of T_S.

For illustration, we assume the sentence *John loves Mary and Linda hates Peter*. The sentence consists of two coordinated clauses C_1 = {*John loves Mary*}, C_2 = {*Linda hates Peter*} and one clause boundary $B_{1,2}$ = {*and*}. Therefore, the clause chart of the sentence is 0B0. In Step 1, C_1 and C_2 are parsed to get T_1 and T_2 with the root nodes r_1 = *loves* and r_2 = *hates*, resp. In Step 2, the sequence S = *loves and hates* is created. In Step 3, S is parsed to get T_S and, finally, in Step 4, T_1 and T_2 become subtrees of T_S.

We evaluated the proposed parsing strategy only on the sentences having the 0B0 clause chart, i.e., on the subsets of the treebank datasets. Table 3 presents the unlabeled attachment score achieved for

- full-scale parsing, i.e., parsing complete sentences using MST (*FS*)
- parsing individual clauses instead of parsing complete sentences, i.e., MST performance is measured on individual clauses (*Clauses*)
- full-scale parsing using the *CCP* strategy

We observe that parsing performance measured on complete sentences is the highest when parsing individual clauses. Using the CCP method we achieved an average 1.36% improvement in UAS.

Data	Sent	FS	Clauses	CCP
PDT dtest	319	82.28	86.87	84.80
PDT etest	352	83.43	87.16	84.67
CAC 2.0	2,272	80.96	84.69	82.34

Table 3: Parsing evaluation on the 0B0 sentences of three different parsing strategies: full-scale parsing (*FS*) using MST, parsing individual clauses (*Clauses*), and full-scale parsing using CCP (*CCP*).

5.2 Parsing governing and dependent clauses

Table 4 presents the unlabeled attachment score achieved for full-scale parsing and parsing individual clauses.

We observe almost no improvement when parsing individual clauses. Also, we observe that the parser performance on the 0B1 sentences is significantly higher than the parser performance on the