Parsing with Treebank Grammars: Empirical Bounds, Theoretical Models, and the Structure of the Penn Treebank

Dan Klein and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305-9040
{klein, manning}@cs.stanford.edu

Abstract

This paper presents empirical studies and closely corresponding theoretical models of the performance of a chart parser exhaustively parsing the Penn Treebank with the Treebank’s own CFG grammar. We show how performance is dramatically affected by rule representation and tree transformations, but little by top-down vs. bottom-up strategies. We discuss grammatical saturation, including analysis of the strongly connected components of the phrasal nonterminals in the Treebank, and model how, as sentence length increases, the effective grammar rule size increases as regions of the grammar are unlocked, yielding super-cubic observed time behavior in some configurations.

1 Introduction

This paper originated from examining the empirical performance of an exhaustive active chart parser using an untransformed treebank grammar over the Penn Treebank. Our initial experiments yielded the surprising result that for many configurations empirical parsing speed was super-cubic in the sentence length. This led us to look more closely at the structure of the treebank grammar. The resulting analysis builds on the presentation of Charniak (1996), but extends it by elucidating the structure of non-terminal inter-relationships in the Penn Treebank grammar. On the basis of these studies, we build simple theoretical models which closely predict observed parser performance, and, in particular, explain the originally observed super-cubic behavior.

We used treebank grammars induced directly from the local trees of the entire WSJ section of the Penn Treebank (Marcus et al., 1993) (release 3). For each length and parameter setting, 25 sentences evenly distributed through the treebank were parsed. Since we were parsing sentences from among those from which our grammar was derived, coverage was never an issue. Every sentence parsed had at least one parse – the parse with which it was originally observed.\(^1\)

The sentences were parsed using an implementation of the probabilistic chart-parsing algorithm presented in (Klein and Manning, 2001). In that paper, we present a theoretical analysis showing an \(O(n^3)\) worst-case time bound for exhaustively parsing arbitrary context-free grammars. In what follows, we do not make use of the probabilistic aspects of the grammar or parser.

2 Parameters

The parameters we varied were:

- Tree Transforms: \textbf{NoTransform}, \textbf{NoEmpties}, \textbf{NoUnariesHigh}, and \textbf{NoUnariesLow}.
- Grammar Rule Encodings: \textbf{List}, \textbf{Trie}, or \textbf{Min}
- Rule Introduction: \textbf{TopDown} or \textbf{BottomUp}

The default settings are shown above in bold face.

We do not discuss all possible combinations of these settings. Rather, we take the bottom-up parser using an untransformed grammar with trie rule encodings to be the basic form of the parser. Except where noted, we will discuss how each factor affects this baseline, as most of the effects are orthogonal. When we name a setting, any omitted parameters are assumed to be the defaults.

2.1 Tree Transforms

In all cases, the grammar was directly induced from (transformed) Penn treebank trees. The transforms used are shown in figure 1. For all settings, functional tags and crossreferencing annotations were stripped. For \textbf{NoTransform}, no other modification was made. In particular, empty nodes (represented as \texttt{-NONE-} in the treebank) were turned into rules that generated the empty string (\(\epsilon\)), and there was no collapsing of categories (such as PRT and ADVP) as is often done in parsing work (Collins, 1997, etc.). For \(^1\)Effectively “testing on the training set” would be invalid if we wished to present performance results such as precision and recall, but it is not a problem for the present experiments, which focus solely on the parser load and grammar structure.
Figure 1: Tree Transforms: (a) The raw tree, (b) NoTransform, (c) NoEmpties, (d) NoUnaries-High (e) NoUnaries-Low