# Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text

by

Noah Ashton Smith

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2006

# Abstract

This thesis is about estimating probabilistic models to uncover useful hidden structure in data; specifically, we address the problem of discovering syntactic structure in natural language text. We present three new parameter estimation techniques that generalize the standard approach, maximum likelihood estimation, in different ways. **Contrastive estimation** maximizes the conditional probability of the observed data given a "neighborhood" of implicit negative examples. **Skewed deterministic annealing** locally maximizes likelihood using a cautious parameter search strategy that starts with an easier optimization problem than likelihood, and iteratively moves to harder problems, culminating in likelihood. **Structural annealing** is similar, but starts with a heavy bias toward simple *syntactic structures* and gradually relaxes the bias.

Our estimation methods do not make use of annotated examples. We consider their performance in both an *unsupervised* model selection setting, where models trained under different initialization and regularization settings are compared by evaluating the training objective on a small set of unseen, unannotated development data, and *supervised* model selection, where the most accurate model on the development set (now with annotations) is selected. The latter is far superior, but surprisingly few annotated examples are required.

The experimentation focuses on a single dependency grammar induction task, in depth. The aim is to give strong support for the usefulness of the new techniques in one scenario. It must be noted, however, that the task (as defined here and in prior work) is somewhat artificial, and improved performance on this particular *task* is not a direct contribution to the greater field of natural language processing. The real problem the task seeks to simulate—the induction of syntactic structure in natural language text—is certainly of interest to the community, but this thesis does not directly approach the problem of exploiting induced syntax in applications. We also do not attempt any realistic simulation of human language learning, as our newspaper text data do not resemble the data encountered by a child during language acquisition. Further, our iterative learning algorithms assume a fixed batch of data that can be repeatedly accessed, not a long stream of data observed over time in tandem with acquisition. (Of course, the cognitive criticisms apply to virtually all existing learning methods in natural language processing, not just the new ones presented here.) Nonetheless, the novel estimation methods presented *are*, we will argue, better suited to adaptation for real engineering tasks than the maximum likelihood baseline.

Our new methods are shown to achieve significant improvements over maximum likelihood estimation and maximum *a posteriori* estimation, using the EM algorithm, for a state-of-the-art probabilistic model used in dependency grammar induction (Klein and Manning, 2004). The task is to induce dependency trees from part-of-speech tag sequences; we follow standard practice and train and test on sequences of ten tags or fewer. Our results are the best published to date for six languages, with supervised model selection: English (improvement from 41.6% directed attachment accuracy to 66.7%, a 43% relative error rate reduction), German (54.4 → 71.8%, a 38% error reduction), Bulgarian (45.6% → 58.3%, a 23% error reduction), Mandarin (50.0% → 58.0%, a 16% error reduction), Turkish (48.0% → 62.4%, a 28% error reduction, but only 2% error reduction from a left-branching baseline, which gives 61.8%), and Portuguese (42.5% → 71.8%, a 51% error reduction). We also demonstrate the success of contrastive estimation at learning to disambiguate part-of-speech tags (from unannotated English text): 78.0% to 88.7% tagging accuracy on a known-dictionary task (a 49% relative error rate reduction), and 66.5% to 78.4% on a more difficult task with less dictionary knowledge (a 35% error rate reduction).

The experiments presented in this thesis give one of the most thorough explorations to date of unsupervised parameter estimation for models of discrete structures. Two sides of the problem are considered in depth: the choice of objective function to be optimized during training, and the method of optimizing it. We find that both are important in unsupervised learning. Our best results on most of the six languages involve both improved objectives and improved search.

The methods presented in this thesis were originally presented in Smith and Eisner (2004, 2005a,b, 2006). The thesis gives a more thorough exposition, relating the methods to other work, presents more experimental results and error analysis, and directly compares the methods to each other.

*Thesis committee (*readers, †advisor):*

*†Jason Eisner (Assistant Professor, Computer Science, Johns Hopkins University)

Dale Schuurmans (Professor, Computing Science, University of Alberta)

Paul Smolensky (Professor, Cognitive Science, Johns Hopkins University)

*David Yarowsky (Professor, Computer Science, Johns Hopkins University)

*For K.R.T.*

# Acknowledgments

*I would have written a shorter letter, but I did not have the time.*
—attributed to Cicero (106–43 BCE), Blaise Pascal (1623–1662), Mark Twain (1835–1910), and T. S. Eliot (1888–1965)

First, I would like to thank my advisor, Jason Eisner, for many helpful and insightful technical conversations during the course of my graduate work. He usually knew when to let me tread on my own and when to grab me by the ankles and drag me (kicking and screaming) to the True Path,[1] and when to let me go play in the woods. Most importantly he taught me to do well by doing good. If I can emulate in my entire career half of the rhetorical flair, vision, or enthusiasm he's displayed in the course of my Ph.D., then I'll count it a success. Thanks also to Debbie and Talia, who on occasion dined *sans* Jason because of me.

My committee, consisting of David Yarowsky, Paul Smolensky, Dale Schuurmans, and, of course, Jason Eisner, have been supportive beyond the call of duty. They have provided valuable insight and held this work to a high standard. (Any errors that remain are of course my own.) Other current and former members of the CLSP faculty have kindly spent time and shared ideas with me: Bill Byrne, Bob Frank, Keith Hall, Fred Jelinek,[2] Damianos Karakos, Sanjeev Khudanpur,[2] and Zak Shafran. Other professors have contributed to my grad experience by teaching great classes: Yair Amir, Scott Smith, Rao Kosaraju, and Jong-Shi Pang. I thank researchers from other places for their helpful comments and discussions of many kinds at various times over the years: Eric Brill, Eugene Charniak, Michael Collins, Joshua Goodman, Mark Johnson, Rebecca Hwa, Dan Klein, John Lafferty, Chris Manning, Miles Osborne, Dan Roth, and Giorgio Satta; also

[1] This statement is not meant to insinuate that Jason ever inappropriately touched my ankles. He is also not to be blamed for my liberal use of footnotes.

[2] Special thanks to these individuals, who voluntarily read drafts and gave feedback. Any remaining errors are of course my own.

anonymous reviewers on my papers. Deep gratitude especially to Philip Resnik and Dan Melamed, who provided early encouragement, continued mentoring, sunlight, and water.

My work during the academic years from 2001 through 2006 was supported generously by the Fannie and John Hertz Foundation. While I can't predict the future impact of the work in these pages, I am certain that it would have been significantly reduced without the rare opportunity for unfettered exploration that this fellowship has offered. I am especially grateful for my annual progress meetings with Lowell Wood; special thanks also to John Holzrichter, Barbara Nichols, and Linda Kubiak. Research included in this dissertation was also supported (during the summers of 2003, 2005, and 2006) by grant number IIS-0313193 from the National Science Foundation to Jason Eisner.

My student colleagues at Hopkins have made the time fun and the atmosphere constructively critical and always creative—a never-ending brainstorm. Their exceptional intelligence need no mention. Thanks to: Radu "Hans" Florian, Silviu Cucerzan, Rich Wicentowski, Jun Wu, Charles Schafer, Gideon Mann, Shankar Kumar, John Hale, Paola Virga, Ahmed Emami, Peng Xu, Jia Cui, Yonggang Deng, Veera Venkataramani, Woosung Kim, Yu David Liu, Paul Ruhlen (R.I.P.), Elliott Drábek,[3] Reza Curtmola, Sourin Das, Geetu Ambwani, David Smith,[2] Roy Tromble,[2] Arnab Ghoshal, Lambert Mathias, Erin Fitzgerald, Ali Yazgan, Yi Su, Chal Hathaidharm, Sam Carliles, Markus Dreyer,[2] Brock Pytlik, Trish Driscoll, Nguyen Bach, Eric Goldlust, John Blatz, Chris White, Nikesh Garera, Lisa Yung, and visitors David Martinez, Pavel Pečina, Filip Jurciček, and Václav Novák. Comments on drafts of my papers and at my seminar talks were always insightful and helpful. David & Cynthia, Roy & Nicole, and Markus: thanks for the camaraderie and the whiteboard. Thanks generally to the GRO.

In the CS Department, Linda Rorke, Nancy Scheeler, Kris Sisson, Conway Benishek, Erin Guest, and Jamie Lurz have made administrivia as painless as they could. In Barton Hall, Laura Graham and Sue Porterfield (the CLSP dream team) are unrivaled in competence and warmth. To Steve Rifkin, Steve DeBlasio, and Brett Anderson in CS, Jacob Laderman and Eiwe Lingefors in CLSP, and Hans, David, and Charles (*encore une fois*) in the NLP lab: thanks for the cycles.

Many, many friends have given me fond memories of the extra-lab time during my Baltimore years. These include the Book Smattering book group: Rita Turner, Ben Kle-

[3]Special thanks to these individuals, who shared knowledge about languages of experimentation.

mens,[2] and many more; and the Baltimore Argentine tango community; and Jens Nielsen, Anna Goldenburg, Ani Nenkova,[3] and Luis von Ahn Arellano. I especially thank Dan Lake for the many hours he listened to me yammer on about a field he gladly left behind long ago.

Penultimately I thank my family. They've all given me unconditional love all along and knew when to stop asking what I do and when I'll finish doing it and get a job. Thanks and loving appreciation to Kay Dixon, Gene Granger, Maria Smith, Lorie Smith (R.I.P.), and Wayne Smith; also Sue & Stu Thickman, John & Florence Smith (R.I.P.), Betty & Bill Dixon, Gert & Cecil Steinberg (R.I.P.), Pat & Ralph Smith, Cindy & Terry Shea, Debbie Green, and Mark Steinberg & Larissa Blechman.

Finally, Karen Thickman, who has shared grad school with me—every part of the roller coaster, every deadline, every triumph, every obstacle, every chapter, every revision, and every tango—and who has filled these years with laughter, spirited intellectual debate, and true love, is thanked.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

*Grammar is not a set of rules; it is something inherent in the language, and language cannot exist without it. It can be discovered, but not invented.*

—Charlton Laird (1901–1984)

This thesis is about uncovering hidden structure in natural language and other sequence data. It presents three new unsupervised parameter estimation techniques: **contrastive estimation**, **skewed deterministic annealing**, and **structural annealing**. These methods are used to obtain significant improvements over the state-of-the-art, predominant approach to estimation of syntax models from sequences alone (the EM algorithm of Dempster, Laird, and Rubin (1977), presented at length in Chapter 3), applied to six diverse natural languages: German, English, Bulgarian, Mandarin, Turkish, and Portuguese. We achieve the best published performance to date on all six languages. Importantly, these are very general methods that can be applied to a wide variety of unsupervised learning problems.

The experiments presented give one of the most thorough explorations to date of parameter estimation in the unsupervised learning of structured models. Two problems are addressed: what function should be optimized when searching for model parameter values, and how should we optimize it?

Unlike most prior work on statistical language learning, the novelty in this thesis is in the *estimation* methods, rather than in learning the *topology* of the model or in defining new types of models. To set the stage for describing the baseline and novel estimation approaches, a certain amount of historical and technical background are required; otherwise the thesis will not be very exciting to read and its relevance will be lost on the reader. This chapter aims to concisely situate the contributions of the thesis in the context of the larger field of natural language processing (hereafter, NLP) and other approaches to the study of formal and natural languages.

## 1.1 The Empirical Revolution in NLP

A paradigm shift took place in NLP during the 1990s. Prior to that time, approaches to solving natural language problems—including parsing and understanding text—were predominantly *symbolic*. Due largely to the efforts of speech recognition researchers at IBM in the late 1980s and early 1990s, *statistical* or *empirical* methods became prominent. The empirical view, which is assumed throughout this thesis, is that language is a natural phenomenon whose effects are observable in the world as speech and text data. If we want to build a computer program that can do useful things with this kind of data, the best thing to do is to learn from the data.

The first argument for empirical techniques is that they work extremely well, given the right kind and a sufficient quantity of data. Beyond that, a number of obvious facts about natural language—for example, that many sentences are ambiguous, having more than one meaning—are very nicely handled in a probabilistic framework that can tell us not only which analyses are *possible*, but how much more likely one is than another.[1] Some excellent reviews have been written on the *scientific* merits of statistical approaches to language; see Abney (1996) and Pereira (2000).

Empirical methods have all but taken over most problems in natural language processing. Borrowing from the fields of pattern recognition, information theory, and machine learning, many (but not all) researchers now believe that the limiting factor for the successful solution of any particular NLP task is the availability of the right kind of data.

### 1.1.1 Parsing

A frequently discussed example today—and the one of central focus in this thesis—is parsing. The reader is taken to be familiar with the problem of natural language parsing: given a sentence and a grammar, find the sentence's linguistic structure (or derivation).

[1]Of course, "empirical" and "probabilistic" are not equivalent and neither implies the other. Transformation-based learning (Brill and Marcus, 1992), for instance, is empirical but not probabilistic; some parsers have manually-chosen weights that are not empirical (Allen, Miller, Ringger, and Sikorski, 1996).

This problem has many sides to it:

- What constitutes linguistic structure? What representations should be used to describe natural language syntax and/or semantics? Good representations will permit phenomena that do occur in language data and disallow those that do not. This problem is tackled largely by researchers who call themselves linguists or computational linguists.

- What algorithms can be developed to make parsing fast? For example, Eisner and Satta (1999) developed new, asymptotically faster methods for certain kinds of parsing formalisms that we exploit in this thesis.

- What data is required to learn to parse, empirically? One of the major events in the empirical revolution in NLP was the development of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993), a million-word dataset of examples of phrase-structure trees for actual English news text sentences. This thesis attempts to learn to parse from data that does *not* include such trees, but the work here would not be possible without some trees for evaluation.

- What symbolic and/or statistical models are appropriate for natural language syntactic descriptions? For example, a debate in recent years was over the usefulness of *bilexical* features in parsing (Klein and Manning, 2003b; Bikel, 2004).

- What learning methods are appropriate for parsing models? A notable recent advance is the use of large-margin training to learn the parameters of a weighted parsing model (Taskar, Klein, Collins, Koller, and Manning, 2004; McDonald, Crammer, and Pereira, 2005a).

After more than ten years of research, there are several widely-used English statistical parsing systems built using the Penn Treebank (Collins, 1999; Charniak and Johnson, 2005). From a state-of-the-art perspective, parsing illustrates the best and worst of NLP in a nutshell: the empirical revolution was hugely successful at developing better parsers and led to a renewed emphasis on replicability and empirical evaluation of new ideas. Yet there remains a nagging concern. How widely useful are these robust treebank-trained parsers? Are they worth the effort that researchers have put into them?

### 1.1.2 Is Parsing Useful?

Only in startlingly recent years have papers been published that demonstrated quite solidly that statistical parsing—broadly construed—can be very useful in natural language engineering applications. Beneficiary applications include machine translation (Chiang, 2005), information extraction (Ramshaw, Boschee, Bratus, Miller, Stone, Weischedel, and Zamanian, 2001; Viola and Narasimhan, 2005), speech recognition (Chelba and Jelinek, 1998), text correction (Shieber and Tao, 2003), and natural language interfaces to databases (Zettlemoyer and Collins, 2005). Not all systems of these kinds make use of parsers, and—tellingly—the parsing models and methods used in the cited systems are not usually the same as those published in parsing papers describing state-of-the-art parsing results.

In almost all of the papers cited in the last paragraph (exceptions are Ramshaw *et al.* (2001) and Shieber and Tao (2003)), in fact, the parsing model was *designed* for the specific task. In Chiang's and Zettlemoyer and Collins' work, no annotated syntactic structures were used in learning at all. That is, the models of syntactic structure (in Chiang's case, bilingual phrase structure, and in Zettlemoyer and Collins' case, a categorial grammar) were acquired without treebanks. Importantly, these papers achieved state-of-the-art results on their tasks and were immediately recognized with best paper awards at their respective conferences.

In our view, the fact that these two home runs for parsing shunned the now-classic treebank paradigm is no coincidence. Learning to parse from treebanks has its advantages—it is an example of supervised learning, which is very well studied—but it has the limitation of *domain dependence*. A parser trained using state-of-the-art techniques and models from a given treebank tends to be very good at parsing data within that treebank's domain, and less good at parsing data outside the domain. This has spurred an area of research on *parser adaptation* (Hwa, 1999; McClosky, Charniak, and Johnson, 2006b).

Taking the domain problem to the extreme, we have the *language* problem. Treebanks are not available in all languages for which demand for language applications exists, though there are new treebanks in new languages appearing every year; see Buchholz and Marsi (2006). These treebanks, however, are for the most part significantly smaller than the original Penn Treebank, and they still suffer from the domain problem. Beyond that, they are expensive and slow to build, and due to linguistic and treebank-convention differences, models designed for one language must in many cases be carefully altered for use with another language and treebank (Collins, Hajič, Ramshaw, and Tillmann, 1999).

A further limitation of treebanks is that their developers are always forced to choose conventions. How many nonterminal category types, and what kinds? What part-of-speech tags, and how fine-grained? How much structure belongs inside of noun phrases? Is projectivity required in the annotations? Should lexical heads be annotated, and according to which theory? Computational linguistics researchers frequently complain about the conventions chosen for the treebank, and any thoughtful researcher is left to wonder whether some changes to the conventions would improve performance on a given application. Indeed, there is much research focused on *transforming* the Penn Treebank for improved parsing performance, starting with Johnson (1998) and continuing more recently with Matsuzaki, Miyao, and Tsujii (2005); Petrov, Barrett, Thibaux, and Klein (2006); Dreyer and Eisner (2006), who use the treebank with unsupervised learning to improve on it.

How far can unsupervised learning go, and are treebanks even necessary? Almost as long as there have been empirical approaches to parsing, there have been empirical approaches to *unsupervised* parsing—learning about the syntactic structure of a natural language from examples of sentences in the language that have not been (fully) annotated (Carroll and Charniak, 1992; Pereira and Schabes, 1992). Unsupervised parsing would require only unannotated text (though, as we will see, permitting the use of a small set of annotated examples is advantageous in most unsupervised learning settings). Since unannotated text is practically free, the domain- and language-dependence issues vanish (apart from the problem of filtering to get a pure corpus of text in the desired domain and language).

Further, it ought to be much cheaper to change conventions (and formalisms and stochastic models) when applying unsupervised learning methods than when using treebanks: simply redefine the model, implement the appropriate statistical inference and decoding algorithms, and retrain. Taking this idea a step farther, parsing models ought to be trainable to be useful for any given *application*, from unannotated data in any given language and domain, with a minimum of human effort.

## 1.2 Unsupervised Parsing and Grammar Induction

To summarize so far, modeling syntactic structure appears to be useful, but the "right" kind of syntactic structure seems to depend heavily on the beneficiary task. There have been studies comparing different kinds of syntactic structure for their usefulness as linguistic explanations (Schneider, 1998) or on specific tasks (Gildea, 2004), and there has been a great deal of work on designing models that can be learned from unannotated data (Klein and Manning, 2001a, 2004). A tremendous amount of research has gone into the learning—through symbolic or statistical means—of *categorial* information about natural language structure. This includes, for example, learning which rules should go into a grammar (e.g., production rules in a context-free grammar or the topology of a finite-state model of language; see Stolcke and Omohundro (1994) and Chen (1995)).

While many of these techniques fit neatly (or almost neatly) into the parameter estimation scheme espoused in this thesis, some do not. We review the latter here, leaving discussion of more closely related papers to the chapter in which each is most relevant.

### 1.2.1 Non-Model-Based Grammar Induction

Several notable attempts to learn language structure without explicit generative models have used measurable statistics in the text stream to discover structure. For example, Magerman and Marcus (1990) found positions of low pointwise mutual information between word tokens and predicted that these fell between grammatical constituents. Brill and Marcus (1992) compared the distributional behavior of two-tag sequences to the behavior of tags and merged instances of the latter into context-free production rules when they were sufficiently similar to the former. Reminiscent of this work is Yuret (1998), who sought to discover lexical dependency structures by choosing links between words with high mutual information, and the modeling of lexical relationships using raw text described in Paskin (2002). Lin (1999) used mutual information to discover non-compositional compounds in unannotated text.

Also reminiscent of this work is the alignment-based learning (ABL) paradigm of van Zaanen (2001). Van Zaanen's algorithm consists of two passes; first, sentences in a corpus are aligned (every sentence with every other sentence) and common parts are proposed as candidate constituents. The second pass considers the hypotheses and seeks an optimal bracketing of the whole corpus, using ad hoc scoring methods and dynamic programming.

Optionally, a grammar may be extracted from the now-labeled data. Similar in spirit is the EMILE system (Adriaans, 1992), which uses word contexts to infer categorial grammar categories.

More cognitively inspired models with similar information-theoretic criteria (e.g., "redundancy reduction") are described by Solan, Horn, Ruppin, and Edelman (2003). This work is notable for its use of the CHILDES corpus (MacWhinney, 1994), a corpus of transcriptions of parent-to-child speech.

### 1.2.2 The Probabilistic Approach

The greatest recent successes in grammar induction—when success is measured by parsing accuracy on gold-standard data—have used generative probabilistic models and unsupervised parameter estimation techniques, specifically maximum likelihood estimation using the EM algorithm (Klein and Manning, 2002a, 2004). The idea is very simple. Define a probabilistic model that jointly scores syntactic analyses with sentences, then estimate the parameters of the model to optimize some function of the data. Depending on the model, this can often be done without any learning of grammar *rules* at all. Here we simply specify all of the rules possible (having specified the form of the model) and learn the parameters for them.

## 1.3 Related Scientific Work

The focus of this thesis is natural language engineering, but problems of learning languages—by people or machines—have fascinated theoretical computer scientists and linguists since long before the empirical revolution or even the birth of NLP. We review some of the key results here and refer the interested reader to an excellent (if perhaps now dated) review by Lee (1996).

### 1.3.1 Theoretical Computer Science

The first formalization of language learnability was by Gold (1967) who defined *identifiability in the limit*. In Gold's very strict setting, a learner observes only positive examples (well-formed strings) from a language in a known class. The learner receives an infinite stream of examples and is guaranteed to eventually see every string in the language. A language family is identifiable in the limit if there exists a learner that will, eventually, settle on the right language and never again change its hypothesis. The learner need not know when it has reached that point.

Gold proved that no superfinite family (i.e., a family of languages containing all finite languages and at least one infinite language) is identifiable in the limit in this setting. This, of course, includes regular languages and context-free languages. Gold's *second* setting (less commonly cited) involves an "informant" that provides positive and negative examples (strings not in the language), and he shows that superfinite families *are* learnable in the limit in this setting. A later negative result due to Angluin (1990) is that the additional ability to query whether a hypothesis is correct (and get a counterexample if it is not) does not aid learnability.

Gold's learning framework is rather limiting: it does not consider the probabilistic setting. Indeed, if the positive examples are drawn from the distribution of a probabilistic context-free language, then that language *is* learnable in the limit without negative examples (Horning, 1969). This distributional information has been taken to be a kind of negative evidence (Seidenberg and MacDonald, 1999; Lewis and Elman, 2001); we will describe an even stronger form of negative evidence in the probabilistic setting (Chapter 4).

### 1.3.2 Language Acquisition in Humans

Partly bolstered by Gold's theorem (though without explicitly relying on it), very strong voices in theoretical linguistics have argued against the learnability of language by humans (Chomsky, 1965). The Chomskyan theory, in a nutshell, is that the capacity for linguistic competence is innate (not learned) and that language acquisition in children boils down to the setting of a few discrete parameters based on the positive evidence (and no negative evidence, such as corrective feedback) received from parents. Under the assumption that children *do* learn the correct grammar of a language, this is taken to mean that humans have an extensive innate grammatical knowledge.

This claim is controversial, with most of the contention revolving around three issues alluded to above:

- Language and learning are assumed to be purely symbolic (rather than probabilistic or connectionist) constructs.

- The "poverty-of-stimulus" argument, that children receive only positive evidence, has been criticized (Pullum, 1996; Pullum and Scholz, 2002).
- It is not clear that all native speakers of a language "know" the same, fully correct grammar.

Another approach to the question of language learnability comes from psychologists and cognitive scientists who study child language learning empirically. Of course, when a child learns proficiency in her first language, she acquires more than just syntax. For example, the meanings of words (and ontological relationships among those meanings) must also be acquired. Much debate has been given to the primacy of syntax versus lexical semantics in child language learning. Some evidence suggests that knowing syntax helps learning word meanings (the *syntactic* bootstrapping hypothesis; Gleitman, 1990), while nativists argue that syntactic categories are innate and the assignment of words to categories is learned using earlier-acquired semantic knowledge (the *semantic* bootstrapping hypothesis; Pinker, 1994). Other studies suggest that children bootstrap syntactic knowledge from phonological (Morgan, 1986) or prosodic features of the speech they encounter (Saffran, Newport, and Aslin, 1996, *inter alia*). To a coarse approximation, the field of child language learning is focused on two scientific questions: what features do children use to acquire linguistic proficiency, and when do these different features come into play?

While this thesis does not seek to address human language learning at all, the computational contributions may be useful tools to researchers who seek to model these phenomena, computationally and statistically (Brent, 1996). In Chapter 4 we introduce new ways to carry out unsupervised learning of log-linear models, which permit joint modeling of arbitrary features of the data. Importantly, this chapter will present a new way to mathematically formulate *implicit* negative evidence in the probabilistic setting. In Chapter 6 we present a framework for changing the feature set of a model during learning. While our models and data are somewhat different from those used by language acquisition researchers, these ideas are not dissimilar in spirit from the underlying scientific questions of child language acquisition.

## 1.4 Aim of the Thesis

The problem faced in this thesis is distinct from both the formal problem faced by Gold's theorem and the cognitive problem of human language acquisition.

In contrast with Gold, we are not interested in identifying the "correct" language. Our aim is to build a structural classifier that will accurately hypothesize syntactic structure (in our case, a very simple form of syntactic structure) for sentences in a given natural language. We do learn from positive examples, but we are not above building in domain knowledge that will help performance, insofar as this can be done cleanly and with mathematical justification (as in Chapter 4). As it happens, our approach will bear an interesting relationship to negative evidence, exploiting a notion of *implicit* negative evidence to improve the accuracy of the learned model.[2] Further, our methods aim to be theory-independent and applicable to any statistical model for which efficient inference algorithms exist.

In contrast with human language learning, our inputs are text rather than speech. In fact we go farther and assume we even know the correct segmentation into words (for languages like Mandarin) and the syntactic categories (also known as word classes or parts of speech) for the words in the data. We also often assume a small set of annotated examples to select from models learned in slightly different settings, and an additional set for evaluation. Our measure of the quality of learning is not a yes/no question ("was the correct language learned?")—we care about the quality of the structural classifier learned, measured on several different metrics (see Section 2.5). We do note that there is research related to what we present that has the explicit goal of arguing against the innateness hypothesis (Clark, 2001), but we make no claims about cognitive plausibility of the learning methods presented here.

The setting faced by NLP practitioners differs in principle very much from the formal and cognitive work, as well as what we present in this thesis. The case can be made that the World-Wide Web presents a (practically) infinite stream of text data in many languages. This data is not perfect; it contains mistakes (and they are not labeled as such). We do not suggest ways to deal with noisy Web data (Resnik and Smith, 2003) or the problem of identifying which language a sample of text or speech is in (see, for instance, White,

[2]In computational learnability research, "negative evidence" typically refers to examples of strings noted to *not* be in the language. In child language learning research, it most often refers to explicit feedback or corrections from parents when the learning child makes a mistake (Marcus, 1993).

Shafran, and Gauvain, 2006).

The learning framework in this thesis is rather traditional: we assume a finite set of clean examples. We define a parametric model family and consider how we can improve learning by changing the way the parameters of the model are estimated (or "trained") from the data. Parameter estimation is usually described as the *optimization* of a numerical function defined by the data and the form of the model. The most common parameter estimation technique is **maximum likelihood estimation**. Familiar alternatives include maximum *a posteriori*, maximum margin, minimum risk, and minimum error training. In this thesis, we will explore well-motivated, mathematically clean ways to modify the objective function for improved accuracy of the learned model.

**A Possible Criticism** The experimentation in this thesis focuses, for the most part, on a single task. It is conceded that the dependency grammar induction task on which performance improvements are demonstrated is an "Olympic sport" in that it only approximates real world problems faced by the NLP practitioner. The problem is certainly artificial in certain respects. We consider only learning from newspaper text, and only in limited quantities. We assume that the syntactic categories (part-of-speech tags) in the training and test data are correctly disambiguated (except in Section 7.2). We only consider a single, very simple representation of syntactic structure, unlabeled dependency trees (though we evaluate the accuracy of the constituent structures implied by the dependencies in Section 7.3). Our model cannot take advantage of many clearly attested linguistic phenomena (see Section 2.3.3). The learning algorithms tested are not cognitively plausible, in that they assume a batch of unchanging data (rather than a stream) which can be perfectly recollected on each iteration of a parameter-refinement algorithm.

We argue that progress on such tasks is still important for the development of the field when the task is replicable and the progress is substantial, two requirements we have been careful to meet. Indeed, experimentation on standard datasets and task definitions has been widely accepted in the research community as an important force in improving components of NLP. Examples include supervised parsing on the Penn Treebank (Collins, 1999; Charniak and Johnson, 2005, *inter alia*), and organized competitions like SENSEVAL (Edmonds, 2002), the CoNLL shared tasks for text chunking (Tjong Kim Sang and Buchholz, 2000), clause identification (Tjong Kim Sang and Déjean, 2001), named entity recognition (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), semantic role labeling

(Carreras and Màrquez, 2004, 2005), and dependency parsing (Buchholz and Marsi, 2006), and the PASCAL textual entailment challenge (Dagan, Glickman, and Magnini, 2005). To bolster the case for our methods, we include a great deal of experimentation on variations of the task, and also on another task (part-of-speech tagging, Section 4.7).

## 1.5 Road Map

The thesis proceeds as follows. Chapter 2 defines the specific grammar induction problem we seek to solve, the model we will estimate to solve it (due to Klein and Manning (2004)), the relevant algorithms for inference with that model, and the measures used to evaluate performance. An illustrations of the kind of dependency structure sought after in the thesis is shown in Figure 1.1, including gold standard and hypothesized structures for which performance was typical under each method, with error counts.

Chapter 3 describes the standard method for unsupervised grammar induction with probabilistic models: the Expectation-Maximization approach to maximum likelihood estimation from partially-observed examples. Simply put, this approach maximizes:

$$\max_{\text{model}} p(\text{data} \mid \text{model}) \tag{1.1}$$

We present baseline performance results on English data and show how MLE with a prior can improve performance slightly:

$$\max_{\text{model}} p(\text{model}) \cdot p(\text{data} \mid \text{model}) \tag{1.2}$$

EM achieves 41.6% directed attachment accuracy on the English data (62.2% undirected).[3] Two major problems that have been widely noted in the literature (but not fully addressed) are that (a) likelihood is not a good predictor of accuracy and (b) likelihood is a bumpy function that is difficult to optimize because of shallow local optima. Chapters 4 and 5 address (a) and (b), respectively, and Chapter 6 addresses both.

Chapter 4 presents contrastive estimation (Smith and Eisner, 2005a,b), which corresponds to a different objective function for unsupervised learning. Contrastive estimation is really a class of objective functions in which likelihood is one (extreme) example. At the core of the idea is that each observed positive example implicates a set of soft negative

[3]Our accuracy measures are defined in Section 2.5.

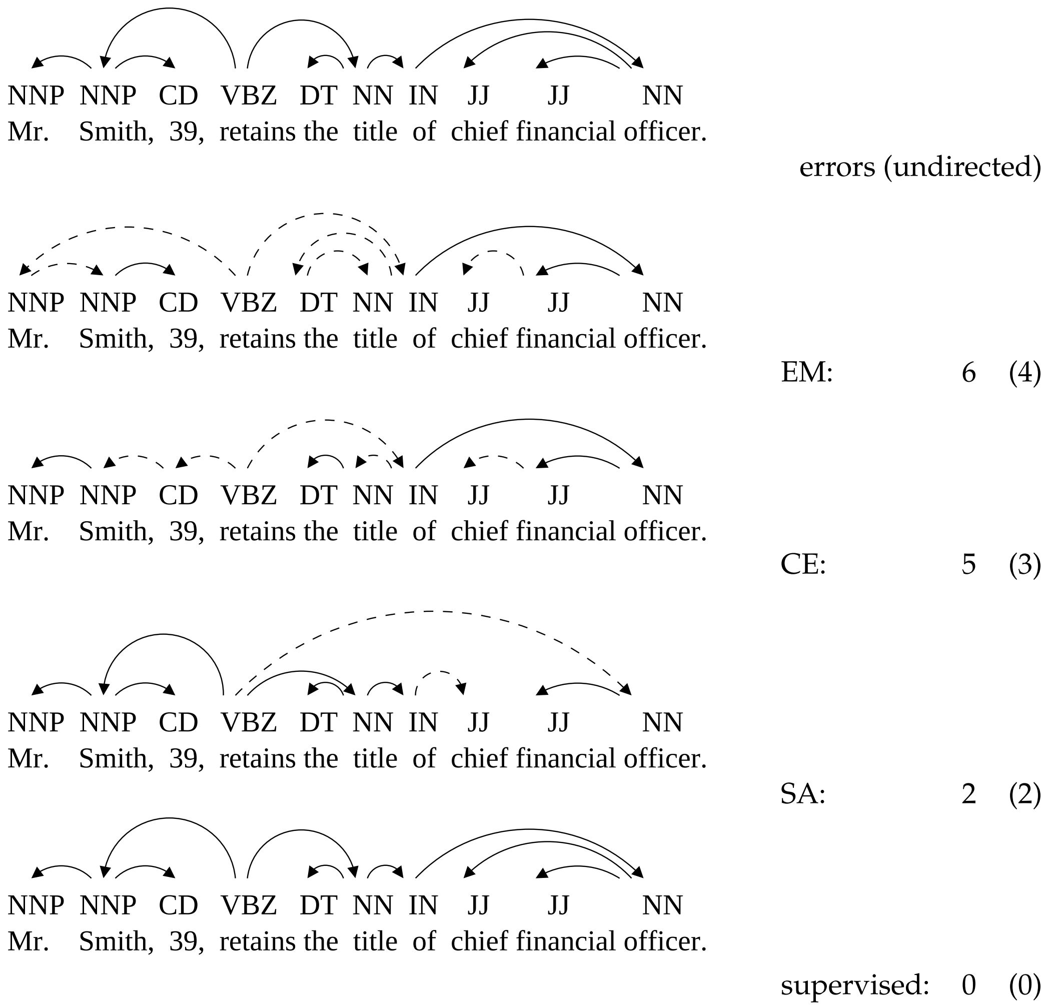

NNP NNP CD VBZ DT NN IN JJ JJ NN
Mr. Smith, 39, retains the title of chief financial officer.
errors (undirected)
NNP NNP CD VBZ DT NN IN JJ JJ NN
Mr. Smith, 39, retains the title of chief financial officer.
EM: 6 (4)
NNP NNP CD VBZ DT NN IN JJ JJ NN
Mr. Smith, 39, retains the title of chief financial officer.
CE: 5 (3)
NNP NNP CD VBZ DT NN IN JJ JJ NN
Mr. Smith, 39, retains the title of chief financial officer.
SA: 2 (2)
NNP NNP CD VBZ DT NN IN JJ JJ NN
Mr. Smith, 39, retains the title of chief financial officer.
supervised: 0 (0)

Figure 1.1: An example sentence from our test set, drawn from the Penn Treebank, with four dependency parses: the gold standard, the baseline method (EM, supervisedly selected), two of our methods (contrastive estimation and structural annealing, supervisedly selected), and a supervised model. This example was chosen because performance on it was typical for each parser (close to the parser's average). The numbers shown are the attachment errors and the undirected attachment errors (defined in Section 2.5). Errors with respect to the gold standard are shown as dashed lines. See Figures 6.14 (page 180) and 9.1 (page 9.1) for more examples.

examples that should (roughly speaking) on average, be less probable than the observed one. It maximizes:

$$\max_{\text{model}} p(\text{model}) \cdot p(D = \text{data} \mid \text{model}, D \in (\text{data} \cup \text{data}')) \tag{1.3}$$

where $\text{data}'$ is perturbed data, taken to be implicitly negative. ($D$ is a random variable over datasets; Equations 1.1 and 1.2 might have referenced it explicitly, writing "$D = \text{data}$" instead of simply "$\text{data}$.") A mathematical formulation is presented, and we show how designing different negative evidence classes affects learning. Substantial improvements over EM are demonstrated on the English parsing task, with some kinds of implicit negative evidence—this chapter achieves 57.6% accuracy (69.0% undirected). CE is also demonstrated to be highly successful (more so than EM) at learning to disambiguate part-of-speech tags in unannotated text. We also present preliminary experiments with a log-linear model of dependency trees that does not fit the stochastic grammar framework well, and show that its performance can be improved using spelling features (also not well-handled by stochastic grammars).

Chapter 5 turns to the problem of parameter search on the bumpy likelihood surface that EM faces. Focusing on the maximum likelihood problem solved (locally) by EM, we show how deterministic annealing (Rose, 1998) can be applied to find better solutions to likelihood (Smith and Eisner, 2004). Deterministic annealing makes use of a sequence of objective functions related to likelihood but less bumpy. It starts with the easiest of these, solving the (very easy) problem

$$\max_{\text{model}} \text{entropy}(p(\text{trees} \mid \text{data}, \text{model})) \tag{1.4}$$

Over time, this function "morphs" into likelihood. DA iteratively uses the solution to the previous problem to initialize parameter search on the next problem, until eventually it is likelihood that is optimized.

We generalize deterministic annealing, presenting skewed deterministic annealing (Smith and Eisner, 2004) as a novel search method that out-performs both EM and deterministic annealing—importantly, though, not because it finds better local optima of the likelihood function. The performance gains in this chapter are relatively small (46.7% accuracy). These results support the case that novel objective functions, not just improved search, hold the key to better grammar induction.

Chapter 6 presents a blended approach, structural annealing (Smith and Eisner, 2006), that does not change the objective as fundamentally as contrastive estimation does, but does impose a bias on the model distribution during learning. The bias under primary consideration will be a preference for string-locality in syntactic relationships. Simply introducing this bias, with the right strength, achieves 45.5% accuracy. If we start out with a very strong bias toward local trees and gradually *anneal* its strength, weakening the bias, we can achieve 66.7% accuracy (73.1% undirected). We also consider a different kind of bias, toward connectedness in syntactic structures, and see a similar, but less pronounced trend.

Chapter 7 compares the various methods under different settings, including training and testing datasets with longer sentences, noisy training data with induced part-of-speech tags, constituent-based evaluation, and random initializers. Structural annealing and contrastive estimation change training in *orthogonal* ways; here we experiment with combining them and see similar patterns of improvement.

The cross-lingual validity of these methods is tested and demonstrated in Chapter 8, where we show that our methods achieve the best published accuracy to date on grammar induction in six languages (Smith and Eisner, 2006). See Table 8.10 (page 212) and Figures 8.2 (page 213) and 8.3 (page 214) for a summary of the improvements by language and a graphical comparison of all methods on all six languages tested. We conclude in Chapter 9. Appendix A gives dynamic programming equations (in the Dyna language) for implementing the inference and decoding methods required for models presented in this thesis. Appendix B gives additional details on the datasets used for experimentation.

Unfortunately, it is impossible to write an entirely self-contained thesis. The reader is assumed to be familiar with probabilistic sequence models like hidden Markov models and stochastic context-free grammars (well-explained in Manning and Schütze, 1999). More recent approaches to estimating these kinds of models (Lafferty, McCallum, and Pereira, 2001) and their log-linear variants are also good to know about. Understanding dynamic programming algorithms will help, but is not essential (a great start is Goodman, 1999). Basic familiarity with the idea of optimizing a continuous function $f : \mathbb{R}^n \to \mathbb{R}$, with or without constraints, and without convexity assumptions, is also required (our favorite reference is Bertsekas, 1995, which may be overly theoretical).

# Chapter 2

# Problem Definition

*[...] each word has a particular and roughly stable likelihood of occurring as argument, or operator, with a given other word, though there are many cases of uncertainty, disagreement among speakers, and change through time.*

—Zellig Harris (1909–1992; Harris, 1988)

We have stated that our goal is to uncover syntactic structure in natural language text through statistical modeling. In this thesis, we will focus almost exclusively on a single probabilistic model of syntactic structure, called here Model A, due to Klein and Manning (2004).[1] In this chapter we motivate and define this model in the context of prior work, noting the algorithms required for inference and decoding and measures used for evaluation.

## 2.1 Dependency Structure

Dependency grammar (Tesnière, 1959; Gaifman, 1965; Hudson, 1984; Mel'čuk, 1988; Fraser, 1989) refers to a variety of formal theories of natural language that focus on the relationships between words in a sentence. At its simplest, dependency structure is one of the most concise ways to describe natural language syntax. Essentially, each word is viewed as the dependent—either an argument or an adjunct—of one other word, with the exception of a single word (usually the matrix verb or matrix auxiliary verb) that is the root of the sentence. An example of a very simple, unlabeled dependency analysis for a

[1]The model is also similar to head-driven parsing models described in in Collins (1999), albeit without lexicalization.

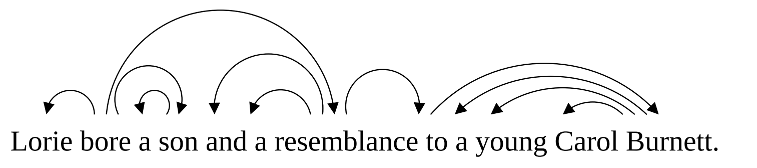

Lorie bore a son and a resemblance to a young Carol Burnett.

Figure 2.1: A projective, unlabeled dependency tree. The root is *bore*. The handling of the conjunction *and* is according to one of many conventions; English conjunctions are a classic difficulty for dependency grammars.

zeugmatic sentence is shown in Figure 2.1. In this figure the arrows point from parents to their children.

### 2.1.1 Why Dependencies?

Before we formalize the notion of a dependency tree and present Model A, which assigns probabilities to dependency trees, it is worth commenting on the choice of the formalism. An important trend in the statistical parsing literature of the past decade has been the use of *lexicalized* formalisms such as lexicalized tree adjoining grammar (Schabes, 1992), lexicalized context-free grammars (Collins, 1996), and other, less grammar-driven models based on lexical features (Magerman, 1995).[2] Indeed, many modern linguistic theories emphasize lexical relationships, as well (Pollard and Sag, 1994; Steedman, 2000; Bresnan, 2001). The Penn Treebank does not include dependency annotations, though Collins (1999) provides a set of rules for augmenting the phrase-structure trees with dependencies.

Eisner (1996) explored a "bare bones" approach to dependency grammar as a "clean testbed" for exploring the usefulness of lexical affinity relationships in parsing. Klein and Manning (2004) also adopted such a model, mainly because they—as we do here—were trying to find syntactic structure without annotated examples and wanted the simplest, most plausibly learnable kind of structure imaginable. Their dependency model is unlexicalized, modeling only the relationships between syntactic categories (parts of speech) and the valency of each tag type. These simple models and representations are arguably useful on their own (see below), but one can also imagine using them as a starting point for learning grammars in more complicated formalisms.

A second motivation for probabilistic dependency grammars is that they are a very

[2]There has more recently been a counter-trend that questions the importance of the role of lexicalization in parsing (Klein and Manning, 2003b).

natural generalization of $n$-gram models used in language modeling for speech recognition and other applications (Jelinek, 1997). Dependency models have often been presented as a more linguistically-motivated model of natural language in which, given a previous sequence of words, the next word is predicted not by $n-1$ most *recent* words (as in an $n$-gram model), but rather by words to which it has some syntactic relationship (Lafferty, Sleator, and Temperley, 1992; Chelba and Jelinek, 1998). Eisner and Smith (2005) considered the spectrum of models between bigram models and bilexical dependency models by imposing and varying constraints on the distance between two words in a dependency relationship, a theme we will return to in the context of structural annealing (Chapter 6).

Recent work on *supervised* dependency parsing has been quite successful at integrating state-of-the-art machine learning techniques for training with rich feature sets (see, for example, Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Nivre and Nilsson, 2005; McDonald, Pereira, Ribarov, and Hajič, 2005b). The NLP community appears to be very interested in dependency models, as evidenced by the recent CoNLL shared task, involving dependency parsing in thirteen languages (Buchholz and Marsi, 2006). Dependency parsing has recently been used in relation extraction (Culotta and Sorensen, 2004), machine translation (Ding and Palmer, 2005), synonym generation (Shinyama, Sekine, Sudo, and Grishman, 2002), inference rule discovery (Lin and Pantel, 2001), and hypernym discovery (Snow, Jurafsky, and Ng, 2004). Current work in NLP on semantic processing, including the problems of semantic role labeling (Carreras and Màrquez, 2004, 2005) and textual entailment (Dagan *et al.*, 2005), very often makes use of dependency analyses (e.g., Cohn and Blunsom, 2005; Raina, Haghighi, Cox, Finkel, Michels, Toutanova, MacCartney, de Marneffe, Manning, and Ng, 2005).

### 2.1.2 Formal Description

Let $\Sigma$ be a finite set of symbols. Let $\mathbf{x} = \langle x_1, ..., x_n \rangle$ be a sequence in $\Sigma^*$. Then a dependency tree is a connected, directed, acyclic graph on the nodes in $\mathbf{x}$. The single source of the graph is called the root. If there is an edge from $x_i$ to $x_j$, then we say $x_i$ is the parent of $x_j$ and $x_j$ is the child of $x_i$. An example is given in Figure 2.1.

In this thesis we add the constraint (not empirically supported for all natural languages; see McDonald *et al.* (2005b)) that the trees be **projective**. Informally, this means no links cross when all edges are drawn on the same side of the nodes (which are laid out

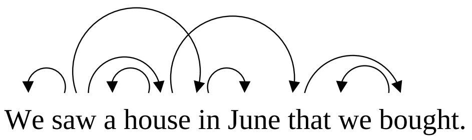


Figure 2.2: A non-projective dependency tree.

in sequence), and no links "cover" the root of the tree; see Figure 2.2. More formally, projective dependency trees obey a linear precedence constraint: if two subtrees are disjoint, then all nodes in one yield linearly precede all nodes in the other yield. This constraint makes dependency grammars that are defined using tree-local rules an instance of *lexicalized context-free grammars*. The proof is omitted (see Gaifman, 1965); an example of the correspondence between the derivations structures is shown in Figure 2.3, and we discuss the relationship further in Section 2.3.1.

In our notation, $\mathbf{y}$ denotes a dependency tree, and $\mathcal{Y}_{\mathbf{x}}$ denotes the set of possible dependency trees consistent with a sequence $\mathbf{x}$ (that is, trees that obey the constraints laid out above). We use $\mathbf{y} : \{0, 1, ..., |\mathbf{x}|\} \rightarrow 2^{\{1,2,...,|\mathbf{x}|\}}$ to refer to the mapping of nodes to their children; $\mathbf{y}(0)$ contains a single node (the root).

Variants of dependency trees that we will touch upon in this thesis include a formalism where $\mathbf{y}(0)$ can contain more than one node ("vine grammar;" Eisner and Smith, 2005), and a formalism where the edges are undirected (similar to link grammar; Sleator and Temperley, 1993).[3] More linguistically interesting dependency grammars can be defined that include labels on the edges and less local constraints on configurations in the tree. Successful learning of our model might be helpful in learning such grammars.

## 2.2 Probabilistic Models over Discrete Structures

A probabilistic model is a way of assigning probability mass to a set of events.[4] Formally, let $\mathcal{W}$ be a set of discrete outcomes (possibly infinite, for example, $\Sigma^*$ or dependency

[3]Link grammar structures need not be trees (cycles are allowed). They consist of undirected, planar, connected graphs with labeled edges. Link grammars are also lexical by design, unlike our models.

[4]In this thesis, we deal only with discrete events, but the idea of course generalizes to continuous sets of events, too.

structures over $\Sigma^*$.). A model $p$ is a function $\mathcal{W} \to [0, 1]$ such that:

$$\sum_{\mathbf{w} \in \mathcal{W}} p(\mathbf{w}) \leq 1 \tag{2.1}$$

A consistent model is one where the above holds with equality. Models that assign probability mass to ill-formed structures that are not in $\mathcal{W}$ are sometimes called *deficient*. (In this thesis, unless otherwise noted, $\mathcal{W} \overset{\text{def}}{=} \{\langle \mathbf{x}, \mathbf{y} \rangle : \mathbf{x} \in \Sigma^*, \mathbf{y} \in \mathcal{Y}_{\mathbf{x}}\}$.)

### 2.2.1 Two Parameterizations of Models over Infinite Structures

If $|\mathcal{W}|$ is finite, we can define a model by listing the events in $\mathcal{W}$ paired with their probabilities. This is called a **multinomial distribution** and underlies many stochastic models over discrete sets. For example, stochastic context-free grammars are defined *using* one multinomial distribution over production rules *for each* nonterminal type.

Many classes of models—including stochastic CFGs—assign probability in a different way that does not involve an enumeration of $\mathcal{W}$. For large $\mathcal{W}$, enumeration would be inefficient (and it is probably very difficult to accurately estimate the probabilities from data). For *infinite* $\mathcal{W}$—like $\Sigma^*$—enumeration is not possible in finite time or space.

The standard approach is to define the probability of a structure $\mathbf{w}$ as a product of probabilities corresponding to steps taken in a stochastic process that "generates" $\mathbf{w}$. Each step is independent of the others (corresponding to *independence assumptions* about parts of $\mathbf{w}$ with respect to other parts of $\mathbf{w}$). Stochastic CFGs and hidden Markov models are two examples of stochastic models based on multinomial distributions. In the remainder of the thesis, we use the term **stochastic model** to refer to multinomial-based models that correspond to stochastic processes.[5]

Another kind of model is a **log-linear** model. Like stochastic models, these define a probability by multiplying together $\mathbb{R}$-valued factors for different pieces of structure. Unlike stochastic models, these **weights** need not be interpretable as probabilities or steps in a stochastic process; though they can be. (If they are, we have a stochastic model, which is just a special case.) To make sure that Equation 2.1 is satisfied, log-linear models divide the product of weights by a constant factor.

[5]Sometimes these are called *generative* models, but in our view that name conflates the model's parameterization and estimation procedure.

**Notation.** Both kinds of models are **parameterized** by the probability or weight factors. We use $\vec{\lambda}$ to refer to the set of stochastic model probabilities in a given model or the set of log-linear weights. We use $\vec{\theta}$ to denote the *logarithms* of those weights. When we write $\lambda$ or $\theta$ with a subscript, we are referring to a specific step's or feature's weight. We use $p_{\vec{\theta}}$ and $p_{\vec{\lambda}}$ equivalently to mean the model parameterized by $\vec{\theta} = \log \vec{\lambda}$; it will be clear from the context whether the model is stochastic or just log-linear.

### 2.2.2 Weighted Dependency Grammars

In the most general form, a weighted dependency grammar defines not only the set of possible trees $\mathcal{Y}_{\mathbf{x}}$ for every sentence $\mathbf{x} \in \Sigma^*$ (a set that may be empty for some $\mathbf{x}$), but also a $\mathbb{R}$-valued score for each tree. This is the central idea behind statistical parsing, which we motivated in Section 1.1.1. Given a weighted grammar, we can pick the highest-scoring parse tree for a sentence, or, more generally, rank possible parses by their scores and define distributions over trees.

Having argued that dependency grammar is a compelling tool for modeling natural language syntax, we note that the parameter estimation methods presented in this thesis are in principle applicable to a much broader class of models, including other kinds of weighted grammars. Smith and Eisner (2005a), for example, applied contrastive estimation (Chapter 4) to a hidden Markov model.

## 2.3 Model A

We now define the probabilistic model used in experiments throughout the thesis, which we call Model A. Model A is due to (Klein and Manning, 2004), who called it the "dependency model with valence." It is a highly simplified model of parent-child syntactic relationships between part-of-speech tags and valency of heads (i.e., number of children a head has). In Model A, $\Sigma$ is a finite set of tag types. The hidden structures modeled by Model A are unlabeled, directed, projective trees over finite tag sequences in $\Sigma^*$. It is not crucial to understand the workings of the model to understand the (rather general) estimation techniques defined in the thesis.

However, it is important to note that the *choice* of model topology (in our case, the grammar rules or features of a tree) has important implications for the accuracy of the learned parameters. Indeed, an important contribution of Klein and Manning (2004) (capitalized on heavily in this thesis) is the extension of earlier statistical dependency models for unsupervised learning (e.g., Carroll and Charniak, 1992) with the notion of head valency. An alternative path, not taken here, would be to explore alternative feature choices for grammar induction, both independently and in combination with the novel estimation procedures presented in this thesis.

There are two ways to think about Model A. First we present it as a stochastic branching process (a stochastic context-free grammar). Then we describe it as a log-linear model of specific features of the sequence $\mathbf{x}$ and the tree $\mathbf{y}$. We then relate it to other models in the literature and suggest some ways it could be enhanced in future work.

### 2.3.1 Stochastic Process (Model A as a SCFG)

Assuming that the reader is familiar with stochastic lexicalized context free grammars, we first present Model A in that form (though it lexicalizes to part-of-speech tags, not words). What do CFGs have to do with dependency grammars? Under the projectivity assumption, dependency grammars produce context-free languages, and the transformation is rather simple. The derivations of the stochastic CFG we describe next are in a one-to-one correspondence with those of Model A (see Figure 2.3).[6]

In the following rule schemata, $x$ and $x'$ are taken to be terminal symbols (part-of-speech tags) in $\Sigma$. Upper-case letters denote nonterminals. The $\lambda$s on the left correspond to the parameters of the model, which are nonnegative weights (probabilities). In our models, these rules are instantiated for all tags $x$ and $x'$ (where referenced), both $\in \Sigma$ (so they should all be understood under universal quantification, which is not written explicitly, for clarity).

| corresponding weight | context-free rule schema |
|---:|:---|
| $\lambda_{\mathbf{root}(x)}$ | $S \rightarrow N[x]$ |
| $\lambda_{\mathbf{stop}(x,\text{right},\mathbf{false})}$ | $N[x] \rightarrow L_0[x]$ |
| $\lambda_{\mathbf{continue}(x,\text{right},\mathbf{false})}$ | $N[x] \rightarrow R_{c_0}[x]$ |
| $\lambda_{\mathbf{child}(x,\text{right},x')}$ | $R_{c_0}[x] \rightarrow R[x]\ N[x']$ |

[6]Because the correspondence is one-to-one, there is no concern over deficiency. SCFGs are not deficient, and neither is Model A.

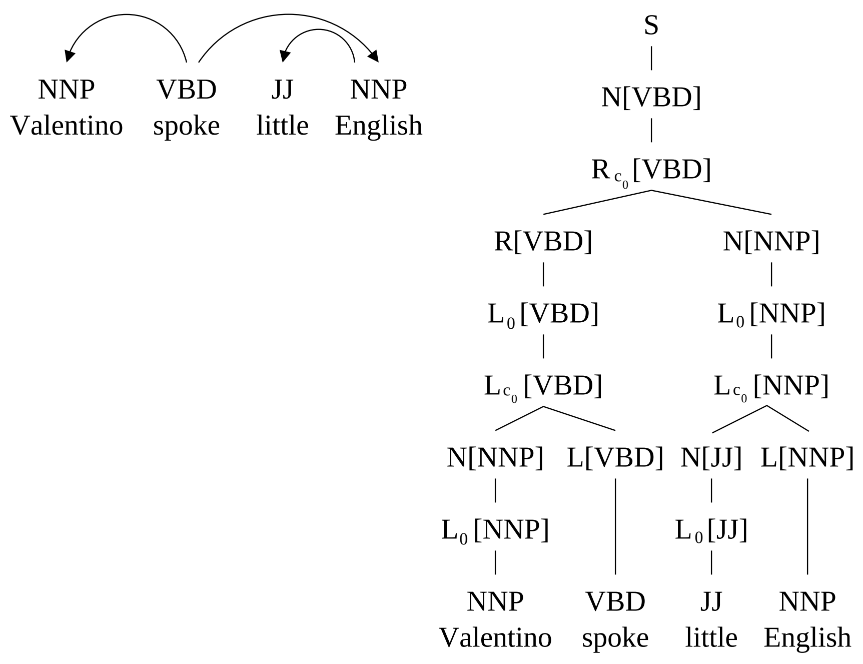
$$\begin{aligned}
\lambda_{\mathbf{stop}(x,\text{right},\mathbf{true})} &\quad R[x] \rightarrow L_0[x] \\
\lambda_{\mathbf{continue}(x,\text{right},\mathbf{true})} &\quad R[x] \rightarrow R_c[x] \\
\lambda_{\mathbf{child}(x,\text{right},x')} &\quad R_c[x] \rightarrow R[x]\ N[x'] \\
\lambda_{\mathbf{stop}(x,\text{left},\mathbf{false})} &\quad L_0[x] \rightarrow x \\
\lambda_{\mathbf{continue}(x,\text{left},\mathbf{false})} &\quad L_0[x] \rightarrow L_{c_0}[x] \\
\lambda_{\mathbf{child}(x,\text{left},x')} &\quad L_{c_0}[x] \rightarrow N[x']\ L[x] \\
\lambda_{\mathbf{stop}(x,\text{left},\mathbf{true})} &\quad L[x] \rightarrow x \\
\lambda_{\mathbf{continue}(x,\text{left},\mathbf{true})} &\quad L[x] \rightarrow L_c[x] \\
\lambda_{\mathbf{child}(x,\text{left},x')} &\quad L_c[x] \rightarrow N[x']\ L[x]
\end{aligned} \tag{2.2}$$

The production rules above may be somewhat confusing. In plain English, a tag $x$ first decides (stochastically) whether it has any children on the right. If so, it generates the first one, then repeatedly decides whether there are any more and, if so, generates the next one. The children are generated from the most distant inward.[7] The same process is then applied on the left of the head. Each child generated (on the left or right) then recursively generates its own set of children. Figure 2.3 gives an example of a dependency structure in both formats.

To see how the production rules relate to the process described, consider that the nonterminals do not provide any information that the dependency tree does not provide. They are merely there to keep track of the state of a head tag during its child-generation process. $N[x]$, for example, means that the token $x$ has not generated any children yet. $R_{c_0}[x]$ means that $x$ will generate a first child on the next production, and $R_c[x]$ means $x$ will generate a non-first child. $L_0[x]$ signifies that all the right children of $x$ have been generated, and now the (possibly empty) sequence of left children will be generated. It is not important that these rules generate the right children first, then the left children—that was an arbitrary choice for exposition—but it is important that they are generated independently of each other, given the head.

Note that the $\lambda$s (probabilities in the SCFG) do not refer to the nonterminals at all. The nonterminals are, however, helpful in telling us what constraints to place on the $\lambda$s to make the model into a valid distribution over dependency trees. For a weighted CFG to

[7]In the present parameterization, this makes no difference. Klein and Manning (2004) described the model as generating children from the head *outward*, but we opt for the opposite because it makes the CFG exposition clearer.



$$\begin{gathered}
\lambda_{\textbf{root}}(\text{VBD}) \\
\cdot\lambda_{\textbf{continue}}(\text{VBD,right,}\textbf{false}) \\
\cdot\lambda_{\textbf{child}}(\text{VBD,right,NNP}) \\
\cdot\lambda_{\textbf{stop}}(\text{VBD,right,}\textbf{true}) \cdot \lambda_{\textbf{stop}}(\text{NNP,right,}\textbf{false}) \\
\cdot\lambda_{\textbf{continue}}(\text{VBD,left,}\textbf{false}) \cdot \lambda_{\textbf{continue}}(\text{NNP,left,}\textbf{false}) \\
\cdot\lambda_{\textbf{child}}(\text{VBD,left,NNP}) \cdot \lambda_{\textbf{child}}(\text{NNP,left,JJ}) \\
\cdot\lambda_{\textbf{stop}}(\text{NNP,right,}\textbf{false}) \cdot \lambda_{\textbf{stop}}(\text{VBD,left,}\textbf{true}) \cdot \lambda_{\textbf{stop}}(\text{JJ,right,}\textbf{false}) \cdot \lambda_{\textbf{stop}}(\text{JJ,left,}\textbf{false}) \\
\cdot\lambda_{\textbf{stop}}(\text{NNP,left,}\textbf{false}) \qquad \cdot \lambda_{\textbf{stop}}(\text{NNP,left,}\textbf{true})
\end{gathered}$$

Figure 2.3: A dependency tree (left) and its representation as a context-free derivation tree under Model A (right). Note that binary production rules are in a one-to-one correspondence with dependencies. No $R_c[\cdot]$ or $L_c[\cdot]$ nonterminals appear because no word has more than one child. The expression at the bottom gives the score of the tree. Each $\lambda$ corresponds to one production rule, and they laid out in a pattern similar to the context-free derivation tree.

be a consistent *stochastic* CFG, the weights for rules that share a left-hand side nonterminal must be nonnegative and must sum up to one. This gives:

$$\begin{aligned}
\forall \lambda_\bullet, \quad & \lambda_\bullet \geq 0 && (2.3)\\
& \sum_{x \in \Sigma} \lambda_{\mathbf{root}(x)} = 1 \\
\forall x \in \Sigma, \forall d \in \{\text{left}, \text{right}\}, \forall b \in \{\mathbf{true}, \mathbf{false}\}, \quad & \lambda_{\mathbf{stop}(x,d,b)} + \lambda_{\mathbf{continue}(x,d,b)} = 1 \\
\forall x \in \Sigma, \forall d \in \{\text{left}, \text{right}\}, \quad & \sum_{x' \in \Sigma} \lambda_{\mathbf{child}(x,d,x')} = 1
\end{aligned}$$

Note that in the SCFG, some distributions are tied; though $R_{c_0}[x]$ and $R_c[x]$ are distinct in their roles within the process, they have identical distributions over right-hand sides (similarly $L_{c_0}[x]$ and $L_c[x]$). Because we have written the parameters, $\lambda$, as variables independently of the rules they are tied to, the constraints above are sufficient. In this thesis, we may refer in general to a partition of $\mathcal{R}$, by which we mean non-overlapping subsets of the rules in $\mathcal{R}$ (whose union is $\mathcal{R}$) such that there is one subset for each multinomial distribution; these are denoted $\mathcal{R}_i$, which will never take any other meaning here.

Given a sequence $\mathbf{x}$ and a dependency tree $\mathbf{y}$, we can succinctly write the probability under Model A of $\langle \mathbf{x}, \mathbf{y} \rangle$ using a recursive equation and a predicate:

$$p_{\vec{\lambda}}(\mathbf{x}, \mathbf{y}) = P(x_r; \vec{\lambda}, \mathbf{y}) \lambda_{\mathbf{start}(x_r)} \quad (2.4)$$

$$\begin{aligned}
P(x_i; \vec{\lambda}, \mathbf{y}) = & \left( \prod_{j \in \mathbf{y}(i) : j < i} \lambda_{\mathbf{continue}(x_i, \text{left}, f(j))} \lambda_{\mathbf{child}(x_i, \text{left}, x_j)} \right) \lambda_{\mathbf{stop}(x_i, \text{left}, [\mathbf{y}(i) > 0])} \\
& \times \left( \prod_{j \in \mathbf{y}(i) : j > i} \lambda_{\mathbf{continue}(x_i, \text{right}, f(j))} \lambda_{\mathbf{child}(x_i, \text{right}, x_j)} \right) \lambda_{\mathbf{stop}(x_i, \text{right}, [\mathbf{y}(i) > 0])}
\end{aligned} \quad (2.5)$$

$$f(j) = \begin{cases} \text{true if } j \text{ is the nearest child on the left or right to its parent} \\ \text{false otherwise} \end{cases} \quad (2.6)$$

For example, the tree in Figure 2.3 gets the probability obtained by multiplying together the $\lambda$ factors shown (also in the figure).

### 2.3.2 Features, and Model A as a Log-Linear Model

Talking about the model as a stochastic process is only part of the picture. What assumptions does it make, and what features does it really capture? We can treat Model A as a log-linear (or exponential) model where each $\lambda$ corresponds to a piece of local structure in the tree. For each piece of structure, we multiply in its weight.

More formally, let $\mathcal{R}$ be a set of structural features. In Model A, these include the following; note that all instances of $\mathrm{left}$ can be replaced with $\mathrm{right}$ as well, and the features are instantiated for all $x, x' \in \Sigma$:

- $\mathbf{root}(x)$: $x \in \Sigma$ is the root of $\mathbf{y}$.
- $\mathbf{child}(x, \mathrm{left}, x')$: $x \in \Sigma$ has $x' \in \Sigma$ as a left child.
- $\mathbf{stop}(x, \mathrm{left}, \mathbf{false})$: $x \in \Sigma$ has no left children.
- $\mathbf{continue}(x, \mathrm{left}, \mathbf{false})$: $x \in \Sigma$ has at least one left child.
- $\mathbf{stop}(x, \mathrm{left}, \mathbf{true})$: $x \in \Sigma$ has only one left child.
- $\mathbf{continue}(x, \mathrm{left}, \mathbf{true})$: $x \in \Sigma$ has a non-first left child.

For each feature $\mathbf{r} \in \mathcal{R}$, let $f_{\mathbf{r}}(\mathbf{x}, \mathbf{y})$ be the number of times the feature occurs in the structure $\langle \mathbf{x}, \mathbf{y} \rangle$. Then we define the log-linear **score** of $\mathbf{x}$ and $\mathbf{y}$ as:

$$\ddot{u}_{\vec{\lambda}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \prod_{\mathbf{r} \in \mathcal{R}} \lambda_{\mathbf{r}}^{f_{\mathbf{r}}(\mathbf{x}, \mathbf{y})} \tag{2.7}$$

where each $\lambda$ is constrained only to be nonnegative. The double-dot notation is a reminder, throughout the thesis, that the model is log-linear. It is often easier to talk in terms of completely unconstrained $\mathbb{R}$-valued parameters, in which case we let $\theta_{\mathbf{r}} = \log \lambda_{\mathbf{r}}$ and write:

$$\ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \exp \sum_{\mathbf{r} \in \mathcal{R}} \theta_{\mathbf{r}} \cdot f_{\mathbf{r}}(\mathbf{x}, \mathbf{y}) \tag{2.8}$$

The subscript of $\ddot{u}$ is overloaded, but we will always use $\vec{\lambda}$ for the multiplicative weights and $\vec{\theta}$ for the additive, log-domain weights.

To turn these scores $\ddot{u}$ into a probability distribution, we normalize by the sum of scores of all productions:

$$\ddot{p}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \frac{\ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{x}' \in \Sigma^*} \sum_{\mathbf{y}' \in \mathcal{Y}_{\mathbf{x}'}} \ddot{u}_{\vec{\theta}}(\mathbf{x}', \mathbf{y}')} = \frac{\ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{w} \in \mathcal{W}} \ddot{u}_{\vec{\theta}}(\mathbf{w})} \tag{2.9}$$

When summing over log-linear scores, we use the shorthand $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ to denote the summation of scores over the set $\mathcal{W}$.

$$\ddot{p}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) = \frac{\ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}{\ddot{Z}_{\vec{\theta}}(\mathcal{W})} \tag{2.10}$$

The stochastic version of Model A, given in Section 2.3.1, constrains the parameters in such a way that $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ is always equal to 1.[8] The denominator is deceptively simple; there are some problems it presents that we will discuss in Chapter 3. For example, if $\mathcal{W}$ is of infinite size (as it is here), $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ may not be finite, and the probability distribution is not well-formed. We will return to this matter in Section 3.5.1.

### 2.3.3 Limitations of Model A

Given its feature set, Model A cannot capture some syntactic phenomena:

- Effects among tags that share a parent. For example, ditransitive verbs (like *give*) should take one direct object and one indirect object; once such a verb has generated the direct object, it should either generate an indirect object or possibly stop.

- Lexical subcategorization effects. The verb *give* is likely to take a direct object, but the verb *rain* is less likely to do so. Model A treats them both equivalently because they are both verbs.

- Lexical selection effects. I am likely to *eat* things like *bagels*, but not *tables*. To Model A, *bagels* and *tables* are both plural nouns.

- Effects between children and their grandparents. For example, in the sentence *I like to chase cats*, the embedded sentence is not well-formed on its own (it has an infinitive verb and no subject). Since the verb *chase* is a child of *like*, the distribution over its left children should disfavor a subject and strongly prefer the infinitive marker *to*.

- Locality effects. Short dependencies are much more common than long ones; distance between parents and children is not explicitly modeled as a feature in Model A. We will return to this idea in Chapter 6.

[8]It is possible for the denominator to be less than one, if the Model is not "tight" (Chi and Geman, 1998). For stochastic context-free grammars, the tightness property is a matter of formal concern (Booth and Thompson, 1973; Wetherell, 1980) but not of great practical concern. Chi and Geman (1998) showed that maximum likelihood estimation of SCFGs from annotated or unannotated examples always results in tight models. The result was extended by Nederhof and Satta (2006).

If we treat Model A as a log-linear model, there is no mathematical problem with incorporating features to try to account for the above phenomena. Features relating the relevant bits of structure can be thrown into the mix and given weights. This is the beauty of log-linear models, but it does not come for free. Computing the normalized probability of a tree may become tricky, involving some pattern matching, but, more critically, summing and maximizing over trees can become computationally expensive and perhaps even intractable—though all of the effects above can be treated in models that are tractable. Speaking very generally, the more structurally local that features are, the less expensive they are to incorporate into the model, giving a dynamic programming algorithm with fewer items and ways to combine items (see Section 2.4).

### 2.3.4 Related Models

We described Model A as a stochastic or log-linear lexicalized context-free grammar. It can also be described as a split head automaton grammar. Head automaton grammar (Alshawi, 1996) is a context-free formalism in which each terminal symbol generates a sequence of children *outward* from the head. Split head automaton grammar (Eisner, 1997; Eisner and Satta, 1999) adds the constraint that the left children are conditionally independent of the right children, given the head. Model A meets this constraint, as we have noted, which is advantageous for computational reasons in inference and decoding (Section 2.4 below). Our SCFG exposition of Model A actually generates more distant children first and proceeds *inward* to the head, but this is not important here, since the children are generated completely independently of each other and identically regardless of their relative positions.

As pointed out by Klein and Manning (2004), Model A is very similar to other models used in experimental work in unsupervised dependency grammar induction (Carroll and Charniak, 1992; Yuret, 1998; Paskin, 2002). Carroll and Charniak's and Yuret's models are similar to Model A in their modeling of associations between elements of $\Sigma$, with directionality (Carroll and Charniak modeled tag sequences, like Model A, Yuret modeled words). Yuret's model is not formulated as a generative process but involves similar features. Paskin (2002) presents a model with similar features in which the graph $\mathbf{y}$ is generated first, chosen randomly and uniformly from among possible trees given the length of the sentence, then the words are filled in. All of those models lack the valency features present in Model A. Models commonly used in *supervised* lexicalized or dependency parsing include far more interesting feature sets to score productions (Collins, 1999) or edges (McDonald *et al.*, 2005a).

A formally more powerful version of Model A might allow **non-projective** trees. McDonald *et al.* (2005b) described a weighted grammar that includes local relationships between parents and children, and trained it using large-margin techniques. Their model was highly effective for English and for Czech (the treebank for which includes non-projective trees). Decoding with their model (finding the best tree for a sentence; see Section 2.4) is done using a maximum spanning tree algorithm in $O(n^2)$ runtime for an $n$-length sequence. Unfortunately, exact inference algorithms have not been developed for computing expectations over the non-projective trees.[9] Adding features that do not correspond simply to edges in the graph (like Model A's valence features) makes decoding less straightforward (and even intractable, depending on the locality of the features; see McDonald and Pereira, 2006).

## 2.4 Inference and Decoding

We have seen how Model A assigns probabilities to dependency trees. This is not particularly useful on its own. To train Model A, apply it to parsing, and evaluate its quality, we need efficient algorithms for **inference** (computing distributions over parse trees, given sentences) and **decoding** (choosing a tree, given a sentence). We turn to these tasks next; in the next section we consider how to evaluate Model A as a parser.

The quantities we require are listed below. The solution to each, for Model A and many weighted grammar formalisms in general, is a semiring dynamic programming algorithm. Readers unfamiliar with dynamic programming as a general method for parsing are referred to Goodman (1999), Eisner, Goldlust, and Smith (2004), and Eisner, Goldlust, and Smith (2005).

1. For a given sequence $\mathbf{x} \in \Sigma^*$, what is the most probable hidden structure:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} p_{\vec{\lambda}}(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})$$

[9] Such an algorithm is likely to exist and to be tractable, given that there is a polynomial-time algorithm for *counting* spanning trees (Kirchhoff, 1847). Counting and score-summing algorithms are often closely related.

This is called *maximum a posteriori* (MAP) decoding (not to be confused with MAP estimation in Chapter 3). Although alternative decoding methods exist (Goodman, 1996), MAP is the most common and the one we will use throughout the thesis. This is solved by a "Viterbi" dynamic programming algorithm using the $\langle \max, \times \rangle$ semiring with a backtrace.[10]

2. For a given sequence $\mathbf{x} \in \Sigma^*$, what is its total score under the model, summing over parse trees:
$$\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) = \ddot{Z}_{\vec{\theta}}(\{\mathbf{x}\} \times \mathcal{Y}_{\mathbf{x}})$$
This is solved by the same algorithm as the MAP decoding problem, but with the $\langle +, \times \rangle$ semiring. This is known as an Inside algorithm.

3. For a given sequence $\mathbf{x} \in \Sigma^*$ and a given model feature $\mathbf{r} \in \mathcal{R}$, what is the expected value of the feature function $f_{\mathbf{r}}$ over trees compatible with $\mathbf{x}$, under the model:
$$\mathbf{E}_{p_{\vec{\lambda}}(\mathbf{Y}|\mathbf{x})}\left[f_{\mathbf{r}}(\mathbf{x}, \mathbf{Y})\right] = \mathbf{E}_{\ddot{p}_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[f_{\mathbf{r}}(\mathbf{x}, \mathbf{Y})\right]$$
This is solved by the Inside algorithm followed by a related algorithm derivable from the Inside algorithm, known as the Outside algorithm.

The semiring dynamic programming algorithm we use for Model A is due to Eisner and Satta (1999), slightly specialized.[11] Its runtime is cubic in the length of the sequence.

The implementation of these follows the general form described in Eisner *et al.* (2005). It is not necessary to deeply understand the algorithms or their implementation to understand the thesis. The algorithm for Model A is given in Dyna-style pseudocode, in a semiring-independent form, in Appendix A.

One important point about these dynamic programming algorithms is that they are exactly the same for the SCFG and log-linear parameterizations of Model A. Recall that the normalizing factor $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ is the only difference between the two parameterizations. Because this is a constant with respect to $\mathbf{x}$ and $\mathbf{y}$, it does not affect the maximization done

[10] The term "Viterbi" comes from the original such algorithm, for hidden Markov models, due to Viterbi (1967).

[11] Because Model A is an SCFG, one could use the semiring versions of Earley's algorithm (Earley, 1970) or, since the grammar contains only binary and unary rules, a slight adaptation of the CKY algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970). The fact that Model A is lexicalized makes this approach markedly less efficient, as pointed out by Eisner and Satta (1999). We instead use their cubic-time algorithm.

by the Viterbi algorithm. The Inside algorithm can be described as computing $\ddot{Z}_{\vec{\theta}}(\{\mathbf{x}\} \times \mathcal{Y}_{\mathbf{x}})$ (a sum of unnormalized scores, $\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})$), which is divided out to compute feature expectations. We will use $\ddot{Z}_{\vec{\theta}}(\mathbf{x})$ to denote this sum of scores.

## 2.5 Evaluation of Dependency Parsers

Another advantage of dependency parsing is that it is extremely easy to evaluate given a gold standard (Lin, 1995). Let $\mathbf{y}^*$ be a gold-standard tree (from a treebank or a treebank augmented with lexical heads using head rules) for sentence $\mathbf{x}$. Let $\mathbf{y}$ be our hypothesis, obtained typically by decoding $\mathbf{x}$ under a particular model (here, usually Model A with the parameters $\vec{\lambda}$ trained using some estimation method).

Informally, the accuracy of a dependency parse hypothesized by a parser (in our case, after decoding under a given model) is the fraction of attachments it gets correct. Below we formally define four variations on this theme.

**Directed accuracy** is the fraction of words in $\mathbf{x}$ that are correctly attached to the appropriate parent word:

$$accuracy_{directed}(\mathbf{y}; \mathbf{x}, \mathbf{y}^*) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}(j), i \in \mathbf{y}^*(j) \\ 0 & \text{otherwise} \end{cases}}{n-1} \tag{2.11}$$

The $n-1$ factor in the denominator corresponds to the total number of attachments possible in an $n$-word sentence, which is one per word, excluding the root. **Undirected accuracy** is a less stringent measure that counts an attachment as correct even if the parent and child roles are reversed:

$$accuracy_{undirected}(\mathbf{y}; \mathbf{x}, \mathbf{y}^*) \stackrel{\text{def}}{=} \frac{\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}(j), (i \in \mathbf{y}^*(j) \text{ or } j \in \mathbf{y}^*(i)) \\ 0 & \text{otherwise} \end{cases}}{n-1} \tag{2.12}$$

Sometimes gold standard treebanks define disconnected trees that have more than one root. (This is also possible for parsers; these are known as *partial* parsers.) In that case, we can define **precision** and **recall** measures:

$$precision(\mathbf{y}; \mathbf{x}, \mathbf{y}^*) \stackrel{\text{def}}{=} \frac{\displaystyle\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}(j), i \in \mathbf{y}^*(j) \\ 0 & \text{otherwise} \end{cases}}{\displaystyle\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}(j) \\ 0 & \text{otherwise} \end{cases}} \tag{2.13}$$

$$recall(\mathbf{y}; \mathbf{x}, \mathbf{y}^*) \stackrel{\text{def}}{=} \frac{\displaystyle\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}(j), i \in \mathbf{y}^*(j) \\ 0 & \text{otherwise} \end{cases}}{\displaystyle\sum_{i=1}^{n} \begin{cases} 1 & \text{if } \exists j \neq 0 : i \in \mathbf{y}^*(j) \\ 0 & \text{otherwise} \end{cases}} \tag{2.14}$$

(The above are directed precision and recall measures; undirected precision and recall can be defined in the obvious way.) When this is required, we will report the harmonic mean of precision and recall, known as the $F_1$ measure; this will not happen until Chapter 8, where we evaluate on treebanks that have this property. When both $\mathbf{y}$ and $\mathbf{y}^*$ are connected dependency trees, each with a single root, directed accuracy, precision, recall, and $F_1$ are equal.

In Section 7.3 we will evaluate our models with PARSEVAL scores (Black, Abney, Flickenger, Gdaniec, Grishman, Harrison, Hindle, Ingria, Jelinek, Klavans, Liberman, Marcus, Roukos, Santorini, and Strzalkowski, 1991), a community standard for evaluating phrase-structure parses against a gold-standard. Another alternative would be to consider the accuracy of dependencies between *content words* only (or specific classes of content words), taking the transitive attachment relationships between such words into account and ignoring other words. For example, in the dependency structure *sat on the chair*, with *sat* $\rightarrow$ *on*, *on* $\rightarrow$ *chair*, *the* $\leftarrow$ *chair*, the words *on* and *the* would be ignored, and the transitive dependency (grandparent relationship) between *sat* and *chair* would be treated as an attachment. This style of evaluation was suggested by Lin (1995).

In this thesis, we will *micro-average* all accuracy measures, summing up the numerators and the denominators separately over an entire corpus of examples before taking the ratio. This gives a per-dependency accuracy measure rather than a per-sentence average accuracy ("macro-averaging") that would inflate our performance numbers because most parsers tend to be more accurate on shorter sentences.

# Chapter 3

# Likelihood and EM

*What most experimenters take for granted before they begin their experiments is infinitely more interesting than any results to which their experiments lead.*

—Norbert Weiner (1894–1964)

In the last chapter we described a stochastic model, Model A, that we would like to apply to the task of learning syntactic structure from a set of unparsed sequences. Learning in this setting involves choosing values for the weights $\vec{\lambda}$ in the model. The standard approach, which is very widely used, is to apply partial-data maximum likelihood estimation (MLE), and the most common computational solution to MLE is the Expectation-Maximization (EM) algorithm. As a baseline method, we review MLE/EM and describe how Model A performs when estimated this way. We also describe an important alternative, *maximum a posteriori* (MAP) estimation, and show how it performs.

This chapter does not present many novel ideas or results; it presents the state-of-the-art baseline on the problem tackled by the remaining chapters. (The idea of explicitly supervised model selection may be novel, and the evaluation here goes beyond prior work with the model we call Model A.) This baseline, EM training of Model A, is a thorough replication of work described in Klein and Manning (2004), with additional commentary, evaluation, and error analysis. A recurring theme will be that maximizing likelihood does not in practice equate to maximizing accuracy.

## 3.1 Partial-Data Maximum Likelihood Estimation

Let $\vec{\mathbf{x}}^{\mathrm{t}} = \left\langle \mathbf{x}_1^{\mathrm{t}}, \mathbf{x}_2^{\mathrm{t}}, ..., \mathbf{x}_{|\vec{\mathbf{x}}^{\mathrm{t}}|}^{\mathrm{t}} \right\rangle$ be an unannotated training dataset.[1] We assume that there are hidden structures $\mathbf{y}_i^{\mathrm{t}}$ for each $\mathbf{x}_i^{\mathrm{t}}$, but we do not know what those hidden structures are. In this thesis, $\mathbf{x}_i^{\mathrm{t}}$ is a sequence of part-of-speech tags and $\mathbf{y}_i^{\mathrm{t}}$ is an unlabeled dependency tree. Recall that $\mathcal{W}$ is used to denote the set of all sequences in $\Sigma^*$ paired with all of their possible hidden structures.

Given a parameterized model family $p_{\vec{\theta}}$ that defines a distribution over $\mathcal{W}$, the **maximum likelihood estimate** given the partially observed dataset, is:

$$\vec{\theta}_{\mathrm{MLE}} \stackrel{\mathrm{def}}{=} \operatorname*{argmax}_{\vec{\theta}} p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}}) = \operatorname*{argmax}_{\vec{\theta}} \sum_{\vec{\mathbf{y}}} p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}}, \vec{\mathbf{y}}) \tag{3.1}$$

Noting that $p_{\vec{\theta}}$ defines a distribution over $\mathcal{W}$, not $\mathcal{W}^*$, we add an independence assumption among the examples in $\vec{\mathbf{x}}^{\mathrm{t}}$: we assume that they were sampled independently from the same underlying distribution.

$$\vec{\theta}_{\mathrm{MLE}} = \operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{|\vec{\mathbf{x}}^{\mathrm{t}}|} p_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}) = \operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{|\vec{\mathbf{x}}^{\mathrm{t}}|} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} p_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y}) \tag{3.2}$$

Let $\tilde{p}$ be the empirical distribution over $\Sigma^*$ (that is, $\tilde{p}(\mathbf{x})$ is the relative frequency of $\mathbf{x}$ in the training set $\vec{\mathbf{x}}^{\mathrm{t}}$). Then MLE is equivalent to minimizing the cross-entropy:

$$\vec{\theta}_{\mathrm{MLE}} = \operatorname*{argmin}_{\vec{\theta}} \mathbf{H}_C \left( p_{\vec{\theta}} \middle\| \tilde{p} \right) = \operatorname*{argmin}_{\vec{\theta}} - \sum_{\mathbf{x} \in \Sigma^*} \tilde{p}(\mathbf{x}) \log \underbrace{p_{\vec{\theta}}(\mathbf{x})}_{\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} p_{\vec{\theta}}(\mathbf{x}, \mathbf{y})} \tag{3.3}$$

The algebraic derivation is straightforward and omitted.

Maximum likelihood estimation has the attractive property that it unifies supervised and unsupervised learning. If no variables are hidden in the training data (supervised learning; there is no hidden structure $\mathbf{y}^{\mathrm{t}}$), then the same objective—$p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}})$—can be used. Notwithstanding the fact that MLE (with any model) is no longer the state-of-the-art supervised learning approach, a common framework for supervised and unsupervised learning offers the advantage of straightforward combination and gradation between the two. For example, Merialdo (1994) combined unlabeled data and labeled data into a single objective function.[2] Pereira and Schabes (1992) estimated a model of phrase structure using a

[1] The $\mathrm{t}$ superscript reminds us that we are looking at training data.

[2] In fact, Merialdo only used the labeled data to initialize unsupervised training of the model.

partially-bracketed corpus. In this thesis, we only ever train on unannotated examples, though we often use some annotated examples to choose among models trained using the same paradigm but with slightly different settings.

### 3.1.1 Experiments

We next describe some experimental baselines and a simple randomized method for choosing model parameters. The experimental setup will remain essentially unchanged until Chapter 8.

**Corpus** Following Klein and Manning (2002a) and Klein and Manning (2004), our training data consist of sentences, stripped of punctuation, of length $\leq 10$ from the Wall Street Journal Penn Treebank (Marcus *et al.*, 1993). Words are replaced by their (gold-standard) part-of-speech tags. Our training set is 5,301 sentences; we use an additional 531 sentences for development data and an additional 530 as test data. $\Sigma$ consists of 35 tags, defined by the Penn Treebank. The gold-standard dependencies come from the rules of Collins (1999).

**Baselines** Two obvious baseline methods are to attach each tag as a child of its immediate right or left neighbor. The former, ATTACH-RIGHT achieves 39.5% directed accuracy; ATTACH-LEFT achieves 22.6% directed accuracy. The two behave identically on the undirected accuracy measure, achieving 62.1%. In this chapter, very few methods will match the performance of ATTACH-RIGHT.[3] We noted in Section 2.5 that it is possible to compute attachment recall and precision; this corpus and parsing model are such that the two are identical, since every sentence is annotated with a single connected tree in both the gold standard and the hypothesis. Therefore recall, precision, and accuracy (whether the three are directed or undirected) are equal.

We will see shortly that partial-data MLE can be carried out using the EM algorithm to *optimize* the likelihood function. First, however, we consider an alternative, very simple

[3]The accuracy scores reported by Klein and Manning (2004) differ somewhat from ours. We suggest four reasons for the (minor) discrepancy. First, our accuracy measure does not count the correct (incorrect) choice for the root of the sentence for (against) a parser, to avoid doubly penalizing a wrong choice of root word. (Incorrectly choosing the root necessarily implies at least one other error in the tree.) In a complete tree, getting the root wrong implies at least one other error. Second, Klein used the entire dataset for both training and testing; we separate out training, development, and testing data. Third, there may be slight differences in the application of Collins' head rules, which in some cases are not unambiguous. Fourth, our implementation of their initializer in EM training is not exactly identical to theirs.

randomized method for approximating partial-data MLE.

The method is to repeatedly sample randomly (and uniformly) from the space of valid models. Assuming that our model is a stochastic grammar whose parameters are multinomial probability distributions—true of the stochastic CFG form of Model A—this can be done by picking points on simplices of the right numbers of dimensions, one point *for each* multinomial distribution in the model. Rubin (1981) and Smith and Tromble (2004) (with a slight correction) provide a method for efficient uniform sampling from this set. In our case, we pick points uniformly from the set that satisfy the constraints in Equations 2.3.

We generated 100 models this way. The distribution of their accuracy scores (directed and undirected) is shown in Figure 3.1.1. Among these models, different selection strategies are possible:

- The model with the highest likelihood on the training data. This model achieved 21.9% directed accuracy and 38.1% undirected accuracy on test data.
- The model with the highest accuracy on an annotated development dataset. This model achieved 31.9% directed accuracy and 45.4% undirected accuracy on test data.

Already we see that higher likelihood on the training data does not imply higher accuracy. Note also that neither model is as good as ATTACH-RIGHT.

The joint distribution among these 100 $\vec{\theta}$s over test-data accuracy and training-data log-likelihood is shown in Figure 3.1.1. Note that the two scores show no strong correlation ($r = 0.12$), and that models with high likelihood tend to have a wide range of accuracies. Perhaps a correlation *would* emerge if we had more models with higher likelihood. We will see in Figure 3.7 that if likelihood is *much* higher (indeed, locally optimal), accuracy tends to be high, but not well correlated with likelihood. We do not show undirected accuracy here (see Figure 3.7); its correlation with likelihood is even smaller ($r = 0.05$). This is one of the major problems with partial-data MLE: likelihood does not predict accuracy with any consistency.

## 3.2 Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm is a general technique for optimizing the likelihood function $p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}})$ (and some other, related functions, as we will see). It was

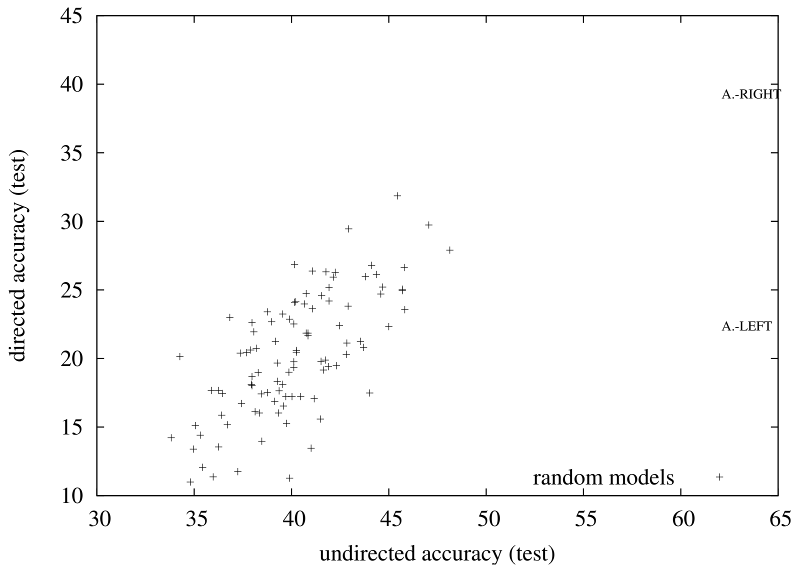


Figure 3.1: Accuracy on test data (directed *vs.* undirected): 100 random models. The ATTACH-LEFT and ATTACH-RIGHT baselines are also plotted.

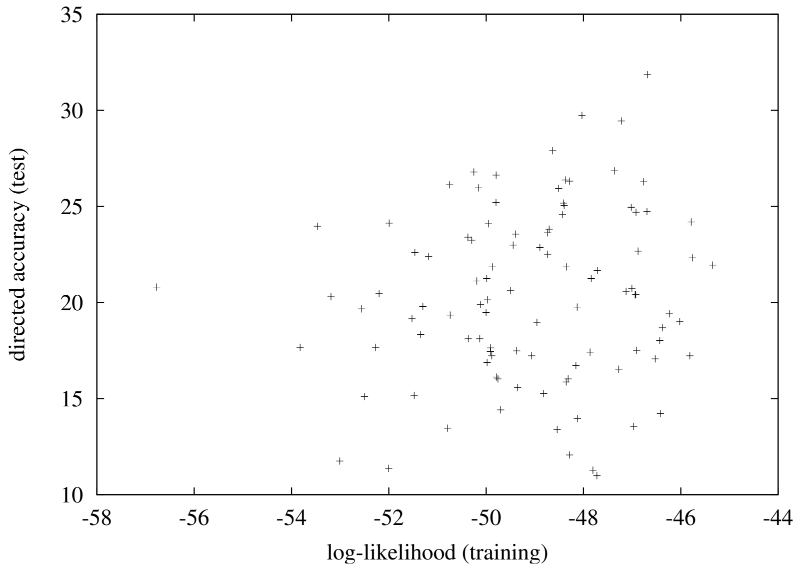


Figure 3.2: Accuracy on test data *vs.* log-likelihood on training data: 100 random models.

**EXPECTATION-MAXIMIZATION:**

1. Initialize $\vec{\theta}^{(0)}$ and let $i \leftarrow 0$.

2. (E step) For each observed $\mathbf{x}^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$ and each possible value $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}$, let

$$q(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \leftarrow p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) = \frac{p_{\vec{\theta}^{(i)}}(\mathbf{x}^{\mathrm{t}}, \mathbf{y})}{\sum_{\mathbf{y}' \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}} p_{\vec{\theta}^{(i)}}(\mathbf{x}^{\mathrm{t}}, \mathbf{y}')} \tag{3.4}$$

3. (M step) Let

$$\vec{\theta}^{(i+1)} \leftarrow \operatorname*{argmax}_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y} \mid \mathbf{X})} \left[ \log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y}) \right] \tag{3.5}$$

   This is equivalent to MLE from complete data, where $q$ is taken to be the conditional distribution over $\mathbf{Y}$ for each $\mathbf{x}^{\mathrm{t}}$. In the complete data case, $q$ would be replaced by the empirical distribution over $\mathbf{Y}$ given $\mathbf{x}$.

4. If $\vec{\theta}^{(i+1)} \approx \vec{\theta}^{(i)}$ then stop; otherwise let $i \leftarrow i + 1$ and go to step 2.

Figure 3.3: The Expectation-Maximization algorithm for locally optimizing likelihood with partially-observed data. This is the discrete data case.

introduced by Dempster *et al.* (1977) as a generalization of the Baum-Welch algorithm for estimating the parameters of a hidden Markov model (Baum, 1972). EM has been widely studied (Neal and Hinton, 1998; Riezler, 1999). EM is a self-scaling hillclimbing method in the space of $\vec{\theta}$ that finds a convergent sequence of parameter estimates $\left\langle \vec{\theta}^{(0)}, \vec{\theta}^{(1)}, ... \right\rangle$ such that $p_{\vec{\theta}^{(t+1)}}(\vec{\mathbf{x}}^{\mathrm{t}}) \geq p_{\vec{\theta}^{(t)}}(\vec{\mathbf{x}}^{\mathrm{t}})$. When EM converges, and $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)}$, then a stationary point of the likelihood has been found.

Here we give a brief review of the EM algorithm.[4] We have opted for an intuitive exposition with enough formulas to allow concreteness and clarity, but we omit proofs and derivations (with one exception). Readers wishing more detail are advised to see tutorials by Berger (1998), Bilmes (1997), and Collins (1997b). Unfortunately the notation varies greatly in different expositions; here we aim for consistency with the rest of the thesis. The EM algorithm is given in Figure 3.3.

EM feels a bit circular; the current parameters are used to get an estimate of the pos-

[4]We stay with the discrete random variable case, because it is more applicable to the problems at hand.

terior $q$ over hidden structures (value of $\mathbf{Y}$), then the posterior is used to reestimate the parameters. How is it that EM effects an improvement in the likelihood?

### 3.2.1 Demonstration that EM Iterations Increase $-\mathbf{H}_C$

There are many ways to think about EM; the most appropriate at this juncture is to think about the mapping from $\vec{\theta}^{(t)}$ to $\vec{\theta}^{(t+1)}$. (In Chapter 5 we will present another way to think about EM: coordinate ascent on $\vec{\theta}$ and $q$ to optimize a related function.) When picking the next estimate of $\vec{\theta}$ during an M step, EM optimizes a function, the expectation $\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X},\mathbf{Y})\right]$.[5] This function is often called an "auxiliary function," and we will write it $Q_{\vec{\theta}^{(t)}}(\vec{\theta})$. The subscript $\vec{\theta}^{(t)}$ is a reminder that $Q$ depends on $q$ (a posterior), which was chosen using $\vec{\theta}^{(t)}$.

What does this function $Q_{\vec{\theta}^{(t)}}$ have to do with the original log-likelihood function to be maximized ($-\mathbf{H}_C\left(p_{\vec{\theta}^{(t+1)}} \middle\| \tilde{p}\right)$)? Consider their *difference*, keeping in mind that this is a function of $\vec{\theta}$, which is what EM is optimizing:

$$
\begin{aligned}
&-\mathbf{H}_C\left(p_{\vec{\theta}} \middle\| \tilde{p}\right) - Q_{\vec{\theta}^{(t)}}(\vec{\theta}) \qquad (3.6)\\
&= -\mathbf{H}_C\left(p_{\vec{\theta}} \middle\| \tilde{p}\right) - \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X},\mathbf{Y})\right]\\
&= \sum_{\mathbf{x}\in\Sigma^*}\tilde{p}(\mathbf{x})\log\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}p_{\vec{\theta}}(\mathbf{x},\mathbf{y}) - \sum_{\mathbf{x}\in\Sigma^*}\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}\tilde{p}(\mathbf{x})q(\mathbf{y}\mid\mathbf{x})\log p_{\vec{\theta}}(\mathbf{x},\mathbf{y})\\
&= \sum_{\mathbf{x}\in\Sigma^*}\tilde{p}(\mathbf{x})\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}q(\mathbf{y}\mid\mathbf{x})\log\sum_{\mathbf{y}'\in\mathcal{Y}_\mathbf{x}}p_{\vec{\theta}}(\mathbf{x},\mathbf{y}') - \sum_{\mathbf{x}\in\Sigma^*}\tilde{p}(\mathbf{x})\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}q(\mathbf{y}\mid\mathbf{x})\log p_{\vec{\theta}}(\mathbf{x},\mathbf{y}) \qquad (3.7)
\end{aligned}
$$

The last step above simply sums over $\mathbf{y}$ inside the outer sum; $\mathbf{y}$ does not appear inside the logarithm, so this move just makes the first term look more similar to the second.

$$
\begin{aligned}
&= \sum_{\mathbf{x}\in\Sigma^*}\tilde{p}(\mathbf{x})\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}q(\mathbf{y}\mid\mathbf{x})\left(\log\sum_{\mathbf{y}'\in\mathcal{Y}_\mathbf{x}}p_{\vec{\theta}}(\mathbf{x},\mathbf{y}') - \log p_{\vec{\theta}}(\mathbf{x},\mathbf{y})\right)\\
&= -\sum_{\mathbf{x}\in\Sigma^*}\tilde{p}(\mathbf{x})\sum_{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}q(\mathbf{y}\mid\mathbf{x})\log\frac{p_{\vec{\theta}}(\mathbf{x},\mathbf{y})}{\sum_{\mathbf{y}'\in\mathcal{Y}_\mathbf{x}}p_{\vec{\theta}}(\mathbf{x},\mathbf{y}')}
\end{aligned}
$$

[5] The reader may notice that this equates to minimizing the cross-entropy $\mathbf{H}_C\left(p_{\vec{\theta}} \middle\| \tilde{p}\cdot q\right)$.

$$
\begin{aligned}
&= -\sum_{\mathbf{x}\in\Sigma^*} \tilde{p}(\mathbf{x}) \sum_{\mathbf{y}\in\mathcal{Y}_{\mathbf{x}}} q(\mathbf{y} \mid \mathbf{x}) \log p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) \\
&= -\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] \qquad (3.8)
\end{aligned}
$$

We can therefore write:

$$
-\mathbf{H}_C\left(p_{\vec{\theta}} \,\middle\|\, \tilde{p}\right) = \overbrace{\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})\right]}^{\text{term 1}} \overbrace{-\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right]}^{\text{term 2}} \qquad (3.9)
$$

Our goal is to show that $-\mathbf{H}_C\left(p_{\vec{\theta}^{(t+1)}} \,\middle\|\, \tilde{p}\right) \geq -\mathbf{H}_C\left(p_{\vec{\theta}^{(t)}} \,\middle\|\, \tilde{p}\right)$. To do this, we will show that an iteration of EM increases both terms in Equation 3.9, a sufficient condition for increasing their sum. Term 1 is simple: by definition of the M step, $\vec{\theta}^{(t+1)}$ maximizes term 1, and therefore for all $\vec{\theta}$, including the previous estimate $\vec{\theta}^{(t)}$,

$$
\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}^{(t+1)}}(\mathbf{X}, \mathbf{Y})\right] \geq \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})\right] \qquad (3.10)
$$

Turning to term 2 in Equation 3.9, note that

$$
\begin{aligned}
&\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}^{(t)}}(\mathbf{Y} \mid \mathbf{X})\right] - \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}^{(t+1)}}(\mathbf{Y} \mid \mathbf{X})\right] \\
&\quad= \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log q(\mathbf{Y} \mid \mathbf{X})\right] - \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}^{(t+1)}}(\mathbf{Y} \mid \mathbf{X})\right] \\
&\quad= \mathbf{E}_{\tilde{p}(\mathbf{X})} \left[\mathbf{D}\left(q \,\middle\|\, p_{\vec{\theta}^{(t+1)}}\right)\right] \\
&\quad\geq 0 \qquad (3.11)
\end{aligned}
$$

(since the Kullback-Leibler divergence can never be negative). Therefore term 2 is also improved by an EM iteration. Minka (1998) also discusses EM as a lower bounding method.

### 3.2.2 Relation to Gibbs Sampling

Gibbs sampling is a general, randomized technique for generating a sample from a joint probability distribution of more than one random variable. Such a sample can be used to approximate the joint distribution or to compute expected values under the joint distribution. Casella and George (1992) provide a good starting point for understanding Gibbs sampling.

We can think of our model as including three random variables: $\vec{\mathbf{\Theta}}$ (the parameters), $\vec{\mathbf{X}}$ (the observed data), and $\vec{\mathbf{Y}}$ (the hidden structure). Gibbs sampling would proceed by first locking $\vec{\mathbf{X}}$ to be $\vec{\mathbf{x}}^{\mathrm{t}}$, the observed training data. We then alternate between sampling a value of $\vec{\mathbf{Y}}$ given our current view of $\vec{\mathbf{\Theta}}$ (and $\vec{\mathbf{X}}$) and sampling a new value of $\vec{\mathbf{\Theta}}$ given our current view of $\vec{\mathbf{Y}}$ (and $\vec{\mathbf{X}}$). At each step, $\vec{\mathbf{Y}}$ and $\vec{\mathbf{\Theta}}$ are each set to a specific value. The effect, over time, is to sample from the joint distribution over $\vec{\mathbf{Y}}$ and $\vec{\mathbf{\Theta}}$ given $\vec{\mathbf{X}}$. The computational steps required for this kind of sampling are quite expensive, and Gibbs sampling is likely to require a very long runtime. (Sampling $\mathbf{Y}$ given $\mathbf{x}$ and $\vec{\theta}$ can be done, for weighted grammar models like Model A, by running the Inside algorithm and sampling derivation paths from the chart.)

EM can be described as a deterministic version of Gibbs sampling. Instead of sampling a value for $\vec{\mathbf{Y}}$, we infer the entire distribution over $\vec{\mathbf{Y}}$ given the current estimate of $\vec{\mathbf{\Theta}}$ and $\vec{\mathbf{x}}^{t}$—this is the E step. Instead of sampling a value for $\vec{\mathbf{\Theta}}$, we take the maximum likelihood estimate (M step).

### 3.2.3 Practical Concerns

The choice of the initial model $\vec{\theta}^{(0)}$ matters tremendously, because EM is a hillclimbing method and the likelihood surface it aims to climb typically has many shallow local optima.[6] De Marcken (1995a) describes the bumpiness problem in detail and with illustrations, for a simple model with a small number of parameters. In practice, initialization (step 1) is typically done in a "clever" way that brings some prior knowledge to bear on the problem.

Numerous papers can be found addressing the convergence properties of EM (Dempster *et al.*, 1977; Wu, 1983; Redner and Walker, 1984; Xu and Jordan, 1996; Jordan and Xu, 1996). When numerical optimization is carried out in software, floating point errors can occur. For this reason, when an algorithm like EM is "near" convergence, tiny changes in the objective function should not be trusted. In practice, the changes in the parameters and the objective function tend to vanish over time (without reaching zero). To keep runtime manageable, we stop optimization of the function $f$ (here, likelihood) when the relative change in the objective function falls below some small value $\epsilon$:

$$\frac{f\left(\vec{\theta}^{(t)}\right) - f\left(\vec{\theta}^{(t-1)}\right)}{f\left(\vec{\theta}^{(t-1)}\right)} < \epsilon \tag{3.12}$$

[6]Charniak (1993) gives anecdotal evidence for a weighted grammar induction problem in which hundreds of initializers each led to a different local optimum under EM search. We will see a similar effect in Section 3.3.2.

## 3.3 EM for Stochastic Grammars

Using EM, we should be able to take an instance of Model A (with specified parameter values) and find a new parameter estimate that is at least as "good" on the likelihood function. Running this to convergence will lead us to a local optimum on the likelihood surface.

Here we outline the computational steps required for efficient EM with stochastic grammars like Model A. Importantly, we make use of Model A's Inside algorithm (Section 2.4). Inside algorithms were arguably invented in service of EM (Baum, 1972; Baker, 1979; Jelinek, Lafferty, and Mercer, 1991).[7]

The important point is that the E step as shown in Figure 3.3 is computationally intractable, since there may be exponentially many values of $\mathbf{y}$ for each $\mathbf{x}_i^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$. Representing the distribution $q$ directly, however, is not necessary. In order to carry out the M step, we only need the *sufficient statistics* to carry out complete-data MLE under the joint distribution $\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y} \mid \mathbf{X})$. For stochastic grammars, these statistics are the expected counts under $\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y} \mid \mathbf{X})$ of each of the grammar rules, computable by the Inside-Outside algorithm. EM for a stochastic grammar is defined in Figure 3.4.

Notice that the Inside and Outside algorithms are carried out on each example in the training set on each iteration. While EM is guaranteed to converge, it may take a very long time to do so, partly because of the computational expense of the dynamic programming.

### 3.3.1 Experiments

In Section 3.1.1 we described how choosing the MLE $\vec{\theta}$ from among a randomly-generated set of $\vec{\theta}$s for Model A performed on parsing English data. Here we use EM to *optimize* the estimate and show markedly improved performance.

**Initializers:** The model, $\vec{\theta}$, consists of 2,765 probabilities, including $|\Sigma| = 35$ **root** probabilities, $8|\Sigma| = 280$ **stop/continue** probabilities, and $2|\Sigma|^2 = 2,450$ **child** probabilities.[8] We make use of three initializers, each consisting of a single special E step followed by an M step. None of these defines a *joint* distribution; they simply provide unnormalized

[7]The paper by Baum is believed to be the first description of an EM algorithm. Dempster *et al.* (1977) gave a more general form.

[8]The number of degrees of freedom is $|\Sigma| - 1$ (for **root**), plus $4|\Sigma|$ (for **stop**), plus $2|\Sigma|(|\Sigma| - 1)$ (for **child**), totaling 2,554.

```
Expectation-Maximization (stochastic grammars):

1. Initialize $\vec{\theta}^{(0)}$ and let $i \leftarrow 0$.

2. (E step) For each rule $\mathbf{r} \in \mathcal{R}$ (the rules of the grammar) let $c_{\mathbf{r}} \leftarrow 0$. For each example
   $\mathbf{x}^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$:

     • Run the relevant Inside and Outside algorithms to compute $p_{\vec{\theta}^{(i)}}(\mathbf{x})$ and (for all
       $\mathbf{r} \in \mathcal{R}$) let $c_{\mathbf{r}} \leftarrow c_{\mathbf{r}} + \mathbf{E}_{p_{\vec{\theta}^{(i)}}(\mathbf{Y}|\mathbf{x}^{\mathrm{t}})}\left[f_{\mathbf{r}}(\mathbf{x}^{\mathrm{t}}, \mathbf{Y})\right]$.

   $\vec{c}$ now contains sufficient statistics for the parameters of the model.

3. (M step) Compute $\vec{\theta}^{(i+1)}$ by normalizing $\vec{c}$. That is, if $\mathbf{r}$ is in the partition $\mathcal{R}_i$ of the
   grammar rules, let $\theta_{\mathbf{r}} \leftarrow \dfrac{c_{\mathbf{r}}}{\sum_{\mathbf{r}' \in \mathcal{R}_i} c_{\mathbf{r}'}}$. This is the maximum likelihood estimate given the
   sufficient statistics.

4. If $\vec{\theta}^{(i+1)} \approx \vec{\theta}^{(i)}$ then stop; otherwise let $i \leftarrow i + 1$ and go to step 2.
```

Figure 3.4: EM for stochastic grammars.

scores over all parses for a given $\mathbf{x}^{\mathrm{t}}$, which are then normalized (implicitly by running the Inside-Outside algorithm) to give the posterior $q$.

**Zero** The special E step is the usual Inside-Outside pass, but with all $\lambda_i = 1$. Note that this is not a valid stochastic grammar (because Equations 2.3 are not satisfied), but the posterior $q$ computed indirectly *is* a valid posterior; in fact it is the uniform posterior over hidden dependency trees for each sentence, with the unnormalized scores given by:[9]

$$\forall \mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}, u(\mathbf{x}^{\mathrm{t}}, \mathbf{y}) = 1 \tag{3.13}$$

We call this the Zero initializer because the parameters in the logarithmic domain are all set to zero.

**K&M** Recall that for dependency trees, we use $\mathbf{y}(i)$ to denote the set of children of the $i$th word in the sentence. We start with a special E step that involves no dynamic programming, defining the posterior as follows for each example $\mathbf{x}^{\mathrm{t}}$:

$$\begin{aligned} p(x_i \text{ is the root}) &= \frac{1}{|\mathbf{x}^{\mathrm{t}}|} \\ p(j \in \mathbf{y}(i)) &= Z(|\mathbf{x}^{\mathrm{t}}|, j)\frac{1}{|i-j|} \end{aligned}$$

where $Z(n, j)$ is an appropriate constant for the $j$th word in an $n$-length sentence, equal to $(1 - |\mathbf{x}^{\mathrm{t}}|^{-1}) \Big/ \sum_{i \in \{1,\ldots,j-1,j+1,\ldots,|\mathbf{x}|\}} |i-j|^{-1}$. The unnormalized score, then, is simply:

$$\forall \mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}, u(\mathbf{x}^{\mathrm{t}}, \mathbf{y}) = \prod_{i=1}^{|\mathbf{x}^{\mathrm{t}}|} \prod_{j \in \mathbf{y}(i)} \frac{1}{|i-j|} \tag{3.14}$$

This is roughly the initializer used by Klein and Manning (2004).

**Local** The special E step scores trees as follows, using a variant of the Inside-Outside algorithm:

$$\forall \mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}, u(\mathbf{x}^{\mathrm{t}}, \mathbf{y}) = \prod_{i=1}^{|\mathbf{x}^{\mathrm{t}}|} \prod_{j \in \mathbf{y}(i)} \left(1 + \frac{1}{|i-j|}\right)$$

[9]Interestingly, $\vec{\theta}^{(0)}$ need not correspond to a feasible point in the log-probability simplex for the given stochastic grammar. It is perfectly reasonable to initialize with any real vector, since such a vector has an interpretation as a log-linear model, and expected values of rule counts can be computed under that model. Under the M step, those counts will lead to a feasible estimate. For some models (none in this thesis), $\mathcal{Y}_{\mathbf{x}}$ is not finite, and this kind of initialization could lead to divergence of $\ddot{Z}_{\vec{\theta}}(\mathbf{x})$.

Similar to K&M, this initializer favors local attachments, but in a softer way. As in the case of Zero, the Inside-Outside computation will give a valid posterior over dependency trees even though the above expression does not have a normalizer.

**Convergence:** See Equation 3.12; here we set $\epsilon = 10^{-10}$ and note performance at possible earlier $\epsilon$ values. Beyond this experiment, we will always use the stopping criterion that either $\epsilon = 10^{-5}$ or 100 iterations have passed, whichever comes first.

Table 3.1 shows, for each of the three initializers, how many iterations it takes to get to each convergence criterion $\epsilon$, the state of the cross-entropy $\mathbf{H}_C$ in bits, and the directed and undirected attachment accuracies at that stage.

To summarize Table 3.1, likelihood does not predict either kind of accuracy with any kind of consistency. Before training, the Zero initializer performs best—it is the favored contestant.[10] But as EM iterations improve its cross-entropy, its directed accuracy falls, and its undirected accuracy fluctuates. Even though Zero appeared to be the best initializer, it did not find the deepest valley among the three initializers on the cross-entropy surface (equivalently, the highest mountain on the likelihood surface).

The K&M initializer achieved by far the best accuracy. Recall how similar it was to the Local initializer, which performed very poorly. This highlights how precarious the situation is with regard to choosing an initializer: two initializers that appear to pick up on the same basic idea may lead to very different results.

In all cases, iterations past $\epsilon = 10^{-5}$ show severely diminishing returns on cross-entropy, given the number of iterations that must be run to achieve successively tighter convergence requirements. (E.g., running to $\epsilon = 10^{-10}$ is more than ten times as slow.) For the remainder of the thesis, we will use $\epsilon = 10^{-5}$ as the stopping criterion for all objective functions and datasets.

A more detailed error analysis of a slightly improved version of this training method follows in Section 3.4.1.

### 3.3.2 Multiple Random Initializations

Before describing the next set of experiments, we describe an important method used throughout the thesis wherever a choice must be made among many close but distinct

[10]If ties are broken randomly in decoding, then this initializer equates to a random classifier that uniformly picks from among the structures in $\mathcal{Y}_{\mathbf{x}}$, given $\mathbf{x}$.

| convergence criterion $\epsilon$ | Zero | | | | K&M | | | | Local | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | test accuracy | | | | test accuracy | | | | test accuracy | |
| | iterations needed | $\mathbf{H}_C$ | directed | undirected | iterations needed | $\mathbf{H}_C$ | directed | undirected | iterations needed | $\mathbf{H}_C$ | directed | undirected |
| ATTACH-RIGHT | | | | | | | | | 0 | – | 39.5 | 62.1 |
| – | 0 | 30.60 | **31.0** | 55.4 | 0 | 30.95 | 32.2 | 53.2 | 0 | 30.64 | 27.4 | 54.3 |
| $10^{-1}$ | 1 | 29.39 | 27.2 | 54.5 | 1 | 29.61 | 36.0 | 55.3 | 1 | 29.32 | **28.2** | 54.9 |
| $10^{-2}$ | 7 | 26.81 | 24.5 | 58.0 | 9 | 25.84 | 38.7 | 59.8 | 7 | 26.83 | 24.6 | 58.1 |
| $10^{-3}$ | 15 | 26.23 | 23.8 | **59.3** | 16 | 25.41 | 40.9 | 61.2 | 14 | 26.25 | 23.8 | **59.2** |
| $10^{-4}$ | 36 | 26.08 | 22.5 | 58.6 | 35 | 25.18 | **41.9** | **62.3** | 36 | 26.08 | 22.6 | 58.9 |
| $10^{-5}$ | 49 | 26.07 | 22.7 | 58.8 | 62 | 25.16 | 41.7 | 62.1 | 49 | 26.07 | 22.8 | 58.9 |
| $10^{-6}$ | 109 | 26.07 | 22.8 | 59.0 | 107 | 25.15 | 41.5 | 62.1 | 75 | 26.07 | 22.8 | 58.9 |
| $10^{-7}$ | 219 | 26.07 | 22.8 | 59.1 | 156 | 25.15 | 41.6 | 62.1 | 207 | 26.07 | 22.9 | **59.2** |
| $10^{-8}$ | 330 | 26.07 | 22.6 | 58.6 | 316 | 25.15 | 41.6 | 62.1 | 322 | 26.07 | 22.8 | 58.9 |
| $10^{-9}$ | 367 | 26.07 | 22.7 | 58.9 | 579 | 25.15 | 41.6 | 62.2 | 397 | 26.07 | 22.7 | 58.9 |
| $10^{-10}$ | 511 | 26.07 | 22.7 | 59.0 | 850 | 25.15 | 41.6 | 62.2 | 665 | 26.07 | 22.7 | 58.9 |

Table 3.1: Model A trained on English data using EM, with three different initializers. $\mathbf{H}_C$ is the cross-entropy (Equation 3.3) on the training data, in bits, which is minimized locally by EM. Boldface marks the best performance in each column. Notice that continued training to closer convergence does not imply improved accuracy!

variations on a training algorithm.

### Model Selection

We can view a deterministic numerical optimization algorithm like EM as a mapping $opt : \mathbb{R}^n \rightarrow \mathbb{R}^n$ from initial point estimates to local optima of the objective function. Instead of starting with a single initializer, we might start with many different ones, chosen randomly (as in Section 3.1.1) or from some engineered set. If $\mathcal{T}$ is the set of initializers under consideration, We then choose:

$$\vec{\theta}^* \leftarrow \max_{\vec{\theta} \in \mathcal{T}} f(opt(\vec{\theta})) \tag{3.15}$$

where $f$ is some **selection** function. In this thesis, we will apply two kinds of model selection. **Supervised model selection** sets $f$ to be the directed attachment accuracy of a model on the development data. **Unsupervised model selection** uses likelihood of the unannotated development data (computed with help from the Inside-Outside algorithm). We will use these approaches throughout the thesis (the former is far more successful, though we will see that very little annotated data is needed for the level of performance obtained by supervised selection). They have the important property—for research and practical use—that no retraining needs to be done if we wish to change the selection function $f$. If a different objective criterion were defined for an application (like machine translation), we could measure the goodness of each model using that application and pick the appropriate model from the set.[11]

Obviously, as $|\mathcal{T}|$ increases, the quality (measured by $f$) of the estimate will increase. This can become very computationally expensive, since iterative numerical optimization algorithms like EM can have long runtimes. One option might be to start training with many different initial models synchronously and strategically allocate computational resources to the different instantiated runs of the algorithm based on the current quality of each iterate. Different strategies are imaginable (see, for example, Gomes and Selman, 2001). In the following section, we simply apply EM to each of the 100 randomly-generated

[11]It is natural to ask whether there are better ways to use annotated examples than simply for selection among purely unsupervisedly trained models. We will address this question directly, for completeness, in Section 3.4.2. Supervised model selection in this thesis works on the assumption that human annotators' syntactic choices are likely to improve task performance—it is still up for debate whether those choices are optimal or even appropriate for any given NLP task.

models from Section 3.1.1, one at a time. After the training of the $k$th model, we apply model selection among the $k$ models trained so far.

### 3.3.3 Experiments

Figure 3.5 shows performance as a function of the number of EM iterations. The supervised model selection method obtains 63.4% directed accuracy (71.7% undirected), but requires 3,144 total iterations of EM (39 random restarts). Unsupervised model selection does not perform nearly as well and does not improve monotonically with the number of random restarts.

When we consider the 100 random models used as initializers and the 100 EM-trained models, we see a marked difference in their accuracy (Figure 3.6). EM training does consistently improve accuracy on random models (though not on all models—recall Table 3.1); see Figure 3.8, which includes Zero, K&M, and Local.

Figure 3.7 shows that there is small variation in the training data log-likelihood after EM training, and (as before) no clear relationship between likelihood and accuracy. The evidence in this figure is particularly damning of partial-data MLE: note that initialization has little effect on the likelihood of the learned model, but accuracy is still very much spread out. The fact that there are so many different learned models is further evidence that the likelihood function has many, many local optima.

### 3.3.4 Variants

For completeness, we now describe an approximate method (Viterbi EM) and an incremental variant. We have not applied these variants to Model A; they are generally viewed as approximations to be used when E steps are too computationally expensive (statistical machine translation is a typical example; Brown, Cocke, Della Pietra, Della Pietra, Jelinek, Lafferty, Mercer, and Roossin, 1990; Brown, Pietra, Pietra, and Mercer, 1993), though there are theoretical justifications (Neal and Hinton, 1998).

#### Winner-take-all or "Viterbi" EM

Maximizing ("Viterbi") dynamic programs often are less expensive in practice to run than their summing (Inside) counterparts. For some problems involving scores of discrete structures, like matchings on weighted bipartite graphs (Melamed, 2000; Smith, 2002) and

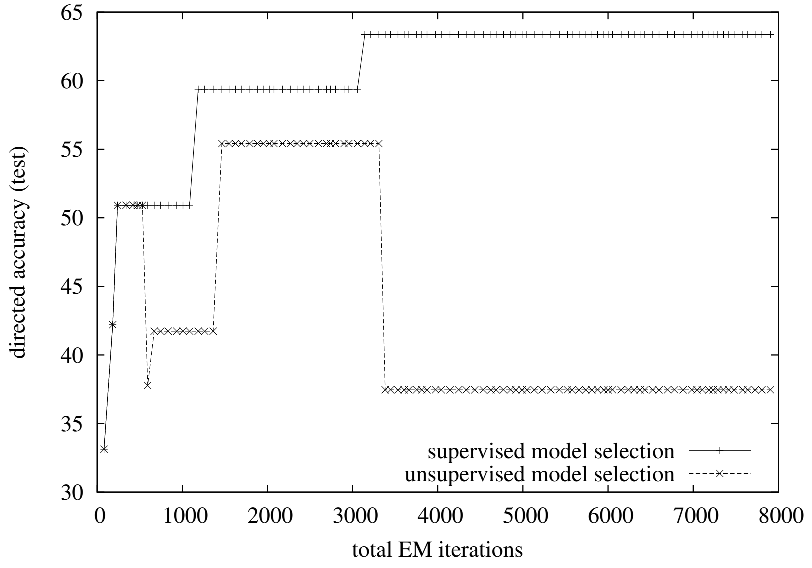


Figure 3.5: EM with multiple random restarts, run independently in an arbitrary order. The $k$th mark in a plot represents selection over the first $k$ models. After each successive model is trained, we plot the test-set accuracy of the trained model with the best accuracy on development data ("supervised") or the one with the best partial-data likelihood on development data ("unsupervised").

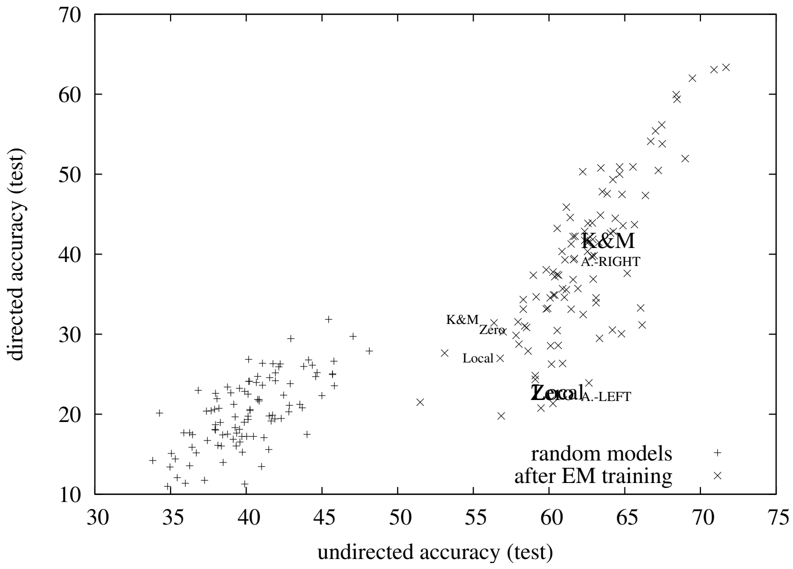


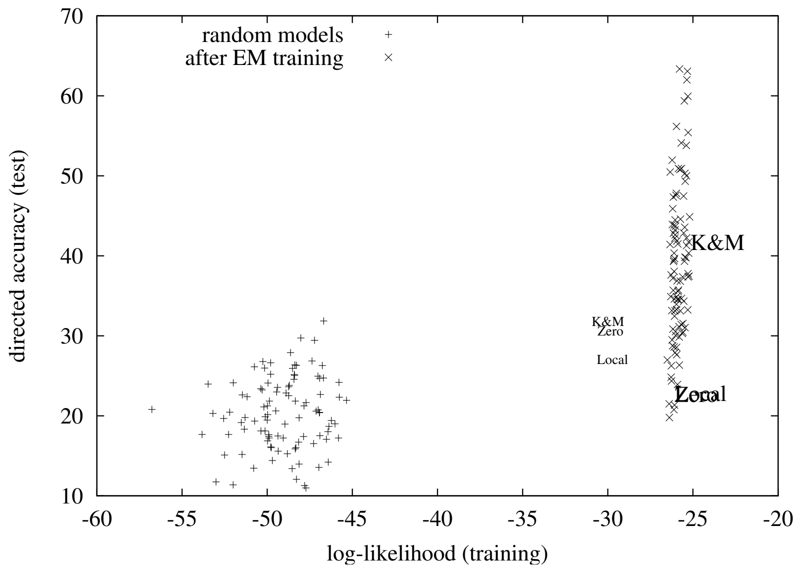
Figure 3.6: Accuracy on test data (directed *vs.* undirected): 100 random models (and our three initializers) before and after EM training. Small type is before training, and large is after—note that Zero and Local are at essentially the same point in this plot after training. The ATTACH-LEFT and ATTACH-RIGHT baselines are also plotted.

spanning trees on weighted graphs (McDonald *et al.*, 2005b), maximization algorithms are known and efficient, but summing algorithms are not. Even in cases where polynomial Inside algorithms are known, the Viterbi variant usually runs faster in practice because heuristic search methods can be employed (Charniak, Goldwater, and Johnson, 1998; Caraballo and Charniak, 1998; Klein and Manning, 2003c).

The "Viterbi" variant of EM replaces the Inside-Outside algorithm with a Viterbi algorithm in the E step. This amounts to an approximation of the posterior probability distribution by its mode:

$$p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \approx \begin{cases} 1 & \text{if } \mathbf{y} = \operatorname{argmax}_{\mathbf{y}'} p_{\vec{\theta}^{(i)}}(\mathbf{y}' \mid \mathbf{x}^{\mathrm{t}}) \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

In practice, Viterbi EM equates to iteratively labeling the data (using the Bayes decision rule and the current model), then re-estimating the model on the labeled data using full-data maximum likelihood estimation. The algorithm is given in Figure 3.9.



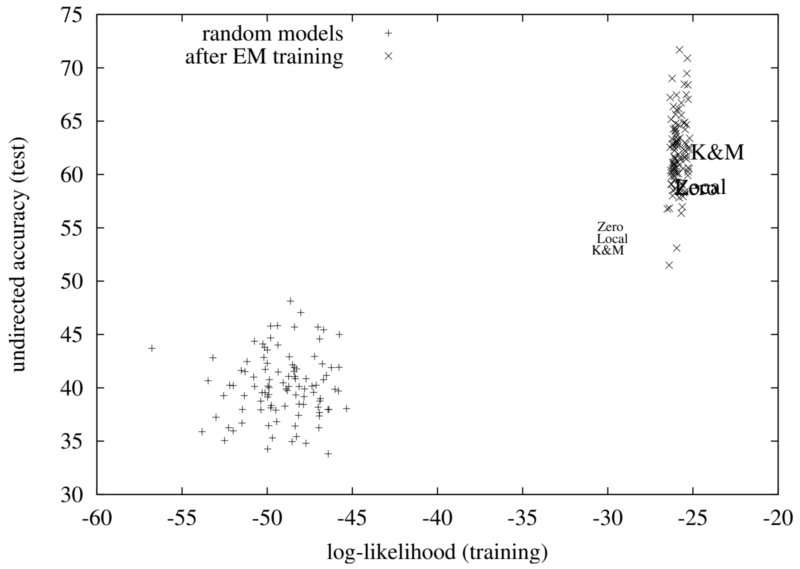


Figure 3.7: Accuracy on test data *vs.* log-likelihood on training data: 100 random models (and our three initializers) before and after EM training. The plot above shows directed accuracy, the plot below undirected. Small labels correspond to our initializers before training, large labels to after training. Zero and Local are at essentially the same point in this plot. The correlation (computed on random models only) between likelihood and directed accuracy increases from $r = 0.12$ to $r = 0.40$; the correlation between likelihood and undirected accuracy increases from $r = 0.05$ to $r = 0.23$. Even among local maxima, there is not a strong correlation between likelihood and accuracy.

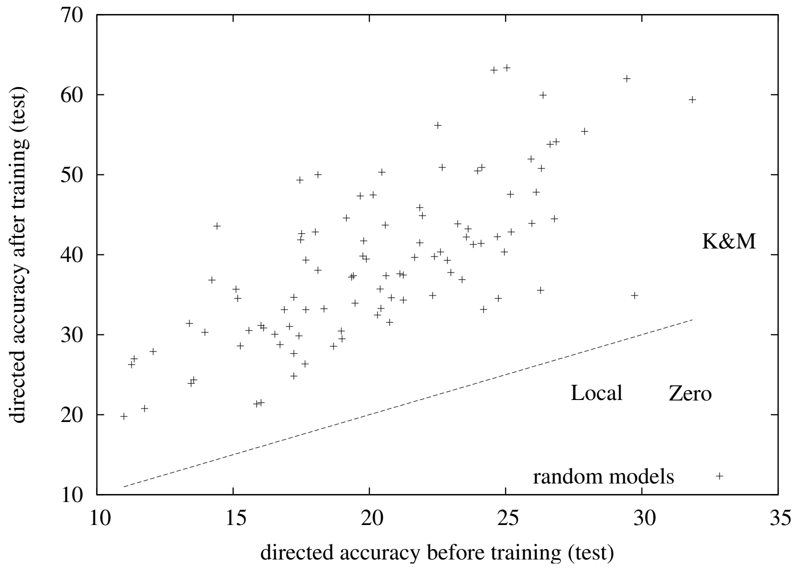


Figure 3.8: Directed accuracy on test data: 100 random models after EM training *vs.* before EM training. The text labels correspond to the more carefully designed initializers. The line is $y = x$.

VITERBI-EM:

1. Initialize $\vec{\theta}^{(0)}$ and let $i \leftarrow 0$.
2. (E step) For each $\mathbf{x}^{\mathrm{t}} \in \vec{\mathbf{x}}^{\mathrm{t}}$, let $best(\mathbf{x}^{\mathrm{t}}) = \mathrm{argmax}_{\mathbf{y}}\, p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}})$, and let the distribution $q(best(\mathbf{x}^{\mathrm{t}}) \mid \mathbf{x}^{\mathrm{t}}) \leftarrow 1$ (and 0 for all $\mathbf{y} \neq best(\mathbf{x}^{\mathrm{t}})$). Typically in practice ties are broken arbitrarily, though one might also divide $q$ among all of the top equally-ranked $\mathbf{y}$s.
3. (M step) Let $\vec{\theta}^{(i+1)} \leftarrow \mathrm{argmax}_{\vec{\theta}}\, \mathbf{E}_{\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})\right]$. This is equivalent to assuming that the labels selected in the E step are correct, and applying maximum likelihood estimation (from complete data).
4. If $\vec{\theta}^{(i+1)} \approx \vec{\theta}^{(i)}$ then stop; otherwise let $i \leftarrow i + 1$ and go to step 2.

Figure 3.9: Viterbi EM.

#### Incremental EM

Another variant of EM is **incremental** EM. Instead of computing sufficient statistics over the whole training dataset before running an M step, run an M step after updating the statistics for each example. Neal and Hinton (1998) describe how this is mathematically justified; here we note that it may have the benefit of faster training.

## 3.4 Maximum *a Posteriori* Estimation

Maximum likelihood estimates are known to be prone to "over-fit" the training data. That is, the model that best fits the training data (in terms of likelihood) may not generalize well to unseen examples.

A classical example of this is an $m$th order Markov chain ($(m + 1)$-gram model) over words. Suppose $m = 2$; a trigram model trained on a finite dataset and then used to compute the probability of a new sentence will typically assign probability 0, because some trigram will appear in the new sentence that was never seen in training. Under the MLE, that trigram has probability 0 (more precisely, $p(x_3 \mid x_1, x_2) = 0$). The term *smoothing* is used to describe a technique—mathematically clean or not—for making the probability estimate less brittle and allowing events not seen in training to receive some probability.

Many smoothing methods can be described as maximum *a posteriori* (MAP) estimation methods. Instead of maximizing likelihood (Equation 3.1), MAP estimates:

$$\vec{\theta}_{\mathrm{MAP}} = \operatorname*{argmax}_{\vec{\theta}} \pi(\vec{\theta}) p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}}) = \operatorname*{argmax}_{\vec{\theta}} \log \pi(\vec{\theta}) + \log p_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}}) \tag{3.17}$$

where $\pi$ is a **prior** density over parameters. The field of Bayesian statistics has a lot to say about priors (we recommend MacKay, 2003). For engineering purposes, priors can present some computational challenges, which we avoid by choosing a family of priors that is simple and fits neatly into EM training.

#### Dirichlet Priors

An example for multinomial-based models (including stochastic grammars) is so-called "add-$\lambda$" smoothing, which adds to the (expected) count of each rule $\mathbf{r} \in \mathcal{R}$ a fixed value $\lambda > 0$ before renormalizing. This is equivalent to MAP estimation with a special symmetric case of a Dirichlet prior over $\vec{\theta}$. The form of the Dirichlet follows. Let $\mathcal{E}$ be a finite set of events, and suppose we wish to define a probability distribution over multinomial distributions over $\mathcal{E}$.[12] The Dirichlet is parameterized by a vector $\vec{\alpha} \in \mathbb{R}_{\geq 0}$ and defines:

$$\pi_{\text{Dirichlet}}(\vec{\lambda}; \vec{\alpha}) = \frac{1}{\mathbf{B}(\alpha)} \prod_{\mathbf{e} \in \mathcal{E}} \lambda_{\mathbf{e}}^{\alpha_{\mathbf{e}} - 1} \tag{3.18}$$

where $\mathbf{B}$ is the Beta function; this term simply normalizes so the density integrates to 1 over the probability simplex.

In practice, MAP estimation of a multinomial with any Dirichlet prior simply requires adding the pseudo-count $\alpha_{\mathbf{e}} - 1$ to the (expected) count (i.e., sufficient statistic) for $\mathbf{e}$. So, to carry out MAP estimation with stochastic grammars, we modify the E step slightly and add $\alpha_{\mathbf{e}} - 1$ to the expected count of rule event $\mathbf{e}$. Add-$\lambda$ is the special case where $\alpha_{\mathbf{e}} = \lambda + 1$ for all $\mathbf{e} \in \mathcal{E}$.

Interestingly, values of $\alpha_{\mathbf{e}}$ that are less than one also give a valid Dirichlet prior. The effect is *sparsity*: many of the model parameters are 0 in the MAP estimate (Mark Johnson, personal communication). Such a prior, however, may require a more complicated M step than simple renormalization of counts.

### 3.4.1 Experiments

We applied MAP estimation, using EM with the symmetric Dirichlet (add-$\lambda$), to the same weighted grammar induction problem. Results are presented in Table 3.2. Notice that, within initializers, the best trial for training set accuracy is less heavily smoothed (smaller $\lambda$) than the best trial for *test* set accuracy. For better generalization to test data, a stronger prior is required.

Note that cross-entropy lessens with the strength of the prior. The prior essentially adds a term to the objective function (see Equation 3.17), so that fitting the training data (low cross-entropy) becomes relatively less important as $\lambda$ increases. It is also interesting to note that convergence occurs more quickly (as a general trend) with more smoothing.

When exploring many different training settings (here, there are 3 initializers $\times$ 7 values of $\lambda$), it is appropriate to apply model selection. This is done as described in Section 3.3.2, above.[13] In addition to unsupervised and supervised model selection, we re-

[12]Remember that valid multinomials are $\vec{\lambda} : \mathcal{E} \to \mathbf{R}_{\geq 0}$ such that $\sum_{\mathbf{e} \in \mathcal{E}} \lambda(\mathbf{e}) = 1$.

[13]We did not—here or elsewhere in the thesis—apply model selection over the randomly-initialized models. The rationale for this is that training 100 randomly initialized models is extremely computationally expensive, especially when other hyperparameters are to be considered. Instead, we stick to a few values for each hyper-

port **oracle** model selection, which refers to the model that performs best on the test data $\vec{\mathbf{x}}^{\tau}$. In this setting, both unsupervised and supervised model selection result in the same choice, which is only slightly worse than oracle model selection (see the last three lines of Table 3.2). This model, initialized by K&M and trained with $\lambda = 10^{-2/3}$, will serve as a baseline in the experiments in the next few chapters, where we consider how to change the objective function to improve accuracy. It achieves 41.6% directed accuracy (not shown in Table 3.2) and 62.2% undirected accuracy.

### 3.4.2 Supervised and Semisupervised Training

Our use of annotated data for the selection among models trained with different hyperparameters (so far, the initializer $\vec{\theta}^{(0)}$ and the prior parameter $\lambda$) had little effect in the experiment of Section 3.4.1—supervised and unsupervised model selection resulted in the same choice (K&M initializer and $\lambda = 10^{-2/3}$). Later, we will see that supervised model selection greatly outperforms unsupervised selection for some other training methods.

First, as an upper bound, we note that training Model A using supervised maximum *a posteriori* estimation on the training set (i.e., assume the trees are observed) and model selection across $\lambda$ achieves 82.5% directed accuracy on the test data. If we use the development dataset *alone* to estimate Model A (with MLE), we can achieve 81.1% accuracy. Figure 3.10 shows how this changes if the annotated development dataset is decreased in size; 56% accuracy can be achieved with only five annotated examples (shown in the same figure).

This is fairly damning for our paradigm of unsupervised estimation with supervised model selection, which (so far) has achieved only 41.6% accuracy (diminished with a smaller development dataset; see Figure 3.10).

It might be expected that semisupervised training—that is, learning from both the annotated and the unannotated examples—could outperform either method. One reasonable technique is to train the model supervisedly (using MLE or MAP, for example) on the annotated data, then initialize EM training with that model. Here we initialized training with a supervised MAP estimate (symmetric $\lambda = 1$ Dirichlet prior). Strikingly, using the unannotated data in this way does not improve the quality of the model over training with the annotated development dataset alone (see the first two curves in Figure 3.10), unless only

parameter, then take the cross-product over those settings.

| | Zero | | | | | | K&M | | | | | | Local | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | directed accuracy | | undirected accuracy | | | | directed accuracy | | undirected accuracy | | | | directed accuracy | | undirected accuracy | |
| $\log_{10} \lambda$ | iterations | $\mathbf{H}_C$ | train | test | train | test | iterations | $\mathbf{H}_C$ | train | test | train | test | iterations | $\mathbf{H}_C$ | train | test | train | test |
| Attach-Right | | | | | | | | | | | | | 0 | – | 38.7 | 39.5 | 62.7 | 62.1 |
| $-\infty$ | 49 | **26.07** | 22.2 | 22.7 | 58.5 | 58.8 | 62 | **25.16** | **43.6** | 41.7 | 63.1 | 62.2 | 49 | **26.07** | 22.2 | 22.8 | 58.5 | 58.9 |
| $-2/3$ | 43 | 26.16 | 22.4 | 23.0 | 58.4 | 59.3 | 49 | 25.24 | 43.3 | 41.6 | **63.1** | 62.2 | 43 | 26.16 | 22.4 | 22.8 | 58.4 | 58.8 |
| $-1/3$ | 43 | 26.24 | 22.8 | 23.0 | 58.6 | 59.4 | 49 | 25.33 | 43.4 | 41.9 | **63.1** | 62.2 | 43 | 26.24 | 22.8 | 23.0 | **58.7** | 59.2 |
| 0 | 76 | 26.33 | 21.4 | 21.8 | **58.9** | 59.9 | 54 | 25.46 | 42.3 | 41.6 | 62.6 | 62.2 | 74 | 26.38 | 22.8 | 23.0 | 58.6 | 59.2 |
| $1/3$ | 43 | 26.62 | 21.5 | 22.0 | 58.7 | **60.0** | 40 | 25.77 | 42.4 | **42.0** | 62.5 | **62.3** | 41 | 26.67 | 22.9 | 23.0 | 58.6 | 59.2 |
| $2/3$ | 43 | 27.14 | 22.1 | 22.4 | 58.2 | 59.5 | 51 | 26.22 | 40.2 | 39.9 | 61.5 | 61.5 | 47 | 27.10 | 21.2 | 21.9 | 58.2 | **59.7** |
| 1 | **38** | 27.99 | **23.6** | **23.8** | 57.5 | 58.9 | **28** | 27.51 | 36.8 | 36.5 | 57.0 | 57.7 | **35** | 27.98 | **24.2** | **24.4** | 57.9 | 59.4 |
| unsupervised model selection: K&M, $\lambda = 10^{-2/3}$ | | | | | | | | | | | | | 49 | 25.24 | 43.3 | 41.6 | 63.1 | 62.2 |
| supervised model selection: K&M, $\lambda = 10^{-2/3}$ | | | | | | | | | | | | | 49 | 25.24 | 43.3 | 41.6 | 63.1 | 62.2 |
| oracle model selection: K&M, $\lambda = 10^{1/3}$ | | | | | | | | | | | | | 40 | 25.77 | 42.4 | 42.0 | 62.5 | 62.3 |

Table 3.2: Model A trained on English data using EM, with three different initializers and different $\lambda$s for add-$\lambda$ smoothing (Dirichlet prior). We show directed and undirected performance on training and test data. Note that $\infty$ is the no-smoothing, $\lambda = 0$ model. Boldface marks the best trial in each column, highlighting the trend (most differences within columns are *not* significant). Model selection here was across $\vec{\theta}^{(0)}$ and $\lambda$.

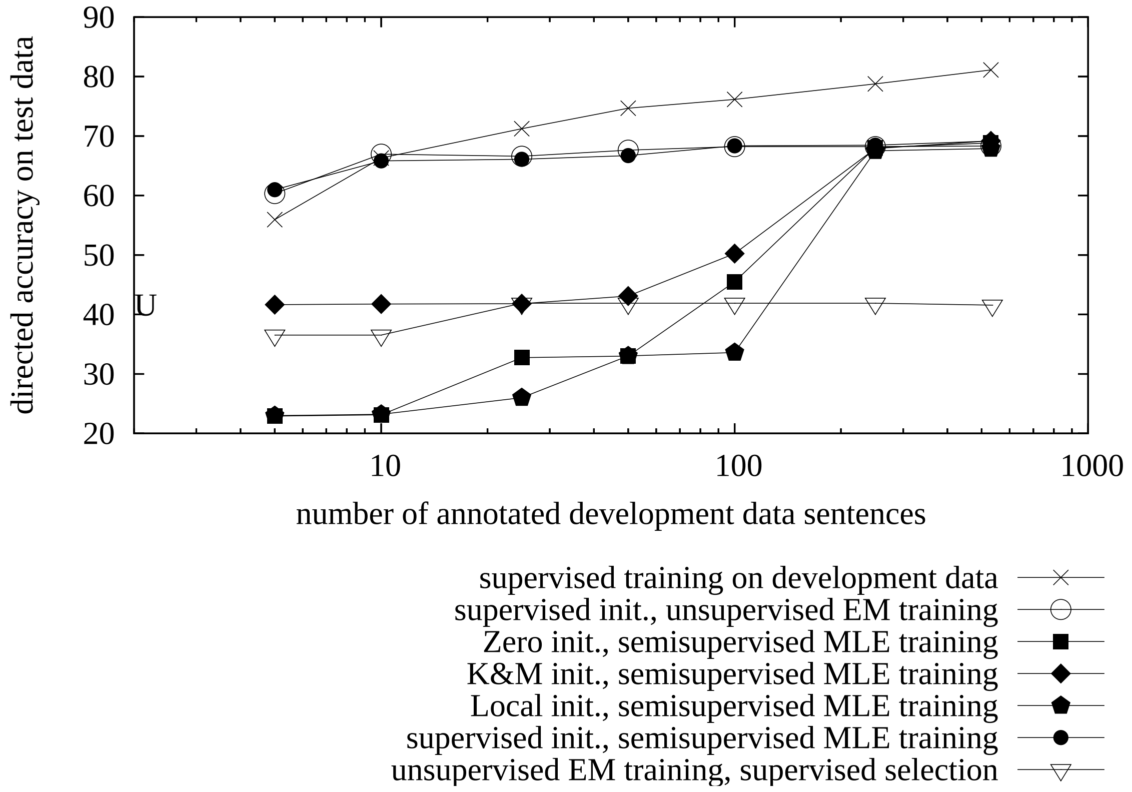


Figure 3.10: Different uses of the development dataset, as the size of the development dataset changes. "Supervised" training refers to using (only) the development dataset to estimate Model A (with add-1 smoothing). "Supervised initialization" refers to using the supervised MLE model trained from the development dataset to initialize EM training on unlabeled data. Merialdo (1994) essentially compared curves like these first two, for a different task. "EM with supervised selection" shows how the experiments in Section 3.3.3 would change if the 531-sentence annotated development dataset were diminished in size (from the plot, we see that very little would change); the "U" marks the performance of unsupervised selection and can be seen as the limit of the curve as the value of $x \to 0$. The "semisupervised" curves show training from different initializers, where the E-counts (on each E step) are augmented with supervised counts from the development dataset—this is equivalent to defining a Dirichlet prior (from the annotated data) for MAP/EM training on the unannotated data.

a tiny number (10 or fewer) of annotated examples are available. This is easy to explain: after the first iteration, the supervised initializer is forgotten, and the unlabeled data have complete control over all future iterations. Similar results were found for part-of-speech tagging by Merialdo (1994).

Another reasonable technique for semisupervised estimation is maximum likelihood on the full dataset (here, unannotated training data plus annotated development data). This can be achieved by running EM on the union of the annotated and unannotated datasets. (This is equivalent to using the sufficient statistics from the annotated data to define a Dirichlet prior—see Section 3.4)—then carrying out MAP/EM training on the unannotated training dataset.) This approach lets the annotated data influence learning throughout EM, not just at the beginning. Figure 3.10 shows how this performs for four different initializers (Zero, K&M, Local, and the supervised initializer suggested above)—for the K&M initializer the curve is the same or better as unsupervised estimation, but for Zero and Local at least 100 annotated examples are needed to outperform our unsupervised training/supervised selection method. The supervised initializer and mixed-data MLE is on par with the same initializer used with MLE on only the unlabeled data.

It is not difficult to imagine more clever ways of combining the labeled and unlabeled data. The annotated examples might be weighted more heavily, or their influence might be gradually reduced (indeed, we will explore such a possibility in Section 5.3.3). While such an exploration is clearly important, it is beyond the scope of this thesis and is left to future work.

To sum up this section, our unsupervised estimation/supervised selection paradigm is not the best way to use annotated training data to construct accurate models. Consistent with much prior work, our attempts to *combine* unannotated and annotated data have not even matched the performance of training with tiny amounts of annotated data alone. Considerably more sophisticated semisupervised learning techniques have been developed, and could be applied to that problem (e.g., Ando and Zhang, 2005).

Importantly, the stated goal of the work in this thesis is to learn *without* supervision. We are therefore faced with a choice: (a) we might forgo the use of annotated data completely (even for model selection), (b) we might use extrinsic tasks to select models, or (c) we might use annotated data for model selection as a proxy for such tasks. We follow (a) and (c). Our best reported results will be achieved using annotated data for model selection, and this is arguably a methodological weakness.

We offer two defenses here before leaving the matter to lie. First, unsupervised learning in NLP nearly always reports performance in comparison to a gold standard test dataset. In parsing, for example, it simply makes sense to evaluate an unsupervised parser against a gold standard. When such results are reported in publication, typically model selection is glossed over (see, e.g., Klein and Manning (2004); counterexamples include Smith and Eisner (2005a,b, 2006)). The full publication record over the past two decades show improvements on related grammar induction tasks, and each paper takes into account lessons learned in the previous one. It is arguable that recent work on unsupervised approaches to this problem (and likely others) has involved implicit supervised model selection over time—possibly on the test datasets—through the biases imposed by knowing prior work, and possibly by selective reporting of results.[14] Indeed, the methodology in this thesis is distinct from prior work in making explicit the use of development data (how much, whether annotated, and when it is used). Arguably all research in this area, then, is using supervised model selection—it is only now that we are saying so. This argument from the *status quo* is not meant to denigrate prior work. Indeed, we believe this kind of gold-standard evaluation is highly appropriate for unsupervised learning, at least at present (see Section 1.4).

The second defense comes from a larger point already made about appropriate hidden structure. Ideally, we should evaluate unsupervised learning methods for parsing in the context of several applications that use parsers (option b, above). That kind of evaluation offers the advantage of the researcher being *agnostic* about precisely which trees (or distributions over trees) should be discovered in the data. While we, personally, are agnostic about whether Treebank trees (or dependency versions of them) are ideal or even appropriate for any given task, this is the best approximation we know of to date for a real application that accepts parsers modularly. An important line of future work, then, is to explore the use of parsers in such settings. As suggested by results in Chiang (2005) and Zettlemoyer and Collins (2005), among others, it is unclear whether gold standard parsing accuracy is a good predictor of system performance. Since test-beds of this kind (b) are unavailable, we opt for (c), which uses annotated data as a proxy for the task of interest.

In the context of contrastive estimation (Chapter 4) and structural annealing (Chapter 6), we will briefly note the performance of semisupervised training in comparison

[14]This may only hold for English, on which most unsupervised syntax learning experiments have been performed.

to our unsupervised estimation/supervised selection approach (Sections 4.6.6, 5.3.3, and 6.2.3).

### 3.4.3 Error Analysis

Table 3.3 breaks down the performance of the selected MAP/EM model by undirected, unordered link types, showing undirected precision and recall for the most common link types in the hypothesized and gold-standard annotations. The model does relatively well with common noun-phrase internal links: determiner/noun (DT/NN), adjective/noun (JJ/NN), and proper noun compounds (NNP/NNP).

A major problem is the failure to connect nouns to the tags they hold syntactic/semantic relationships to, including prepositions (IN/NN) and verbs (NN/VBD, NN/VBZ). This is the result of the model having learned that determiners are the parents of nouns, rather than the reverse. It has been pointed out (in particular by Klein and Manning (2004)) that there is a valid linguistic argument for making determiners the heads of noun phrases, and the convention of making the noun the head is not necessarily the best choice (Abney, 1987). The point is well taken; notice that undirected attachment measures do not *directly* penalize the reversal of the determiner-noun link, but such a mistake does prevent the noun from attaching as a child to the appropriate preposition or verb. At best, the determiner (rather than the noun) will attach to the preposition or verb, and undirected accuracy *will* penalize that error. Notice the over-hypothesizing of preposition/determiner (DT/IN) and verb/determiner (DT/VBD, DT/VBZ) links. Another problem is the under-hypothesizing of adverb-verb links (RB/VBZ, RB/VBD).

Table 3.4 breaks undirected and directed precision and recall down by the length of the attachment—that is, the string distance between a tag and its hypothesized or true parent tag. Each column of the table corresponds to a given distance between words, and only word pairs of that distance are counted in the numerators and denominators for precision and recall scores in that column. The table tells us that the model is more accurate (both in precision and recall) on shorter dependencies. The smaller numbers break down the dependencies by direction of the child (left or right). We see that the model tends to attach tags as adjacent right children too frequently, for example. (This is consistent with the DT $\rightarrow$ NN problem mentioned above.) Notice (by comparing the top and bottom rows, of counts) that the overall distribution of dependency lengths is approximately correct.

| tag types | | count in hyp. | precision | recall | count in gold |
|---|---|---|---|---|---|
| DT | NN | 231 | 94.4 | 95.6 | 228 |
| NNP | NNP | 207 | 60.9 | 70.4 | 179 |
| IN | NN | 73 | 89.0 | 50.8 | 128 |
| NN | VBD | 23 | 100.0 | 19.2 | 120 |
| JJ | NN | 116 | 79.3 | 78.0 | 118 |
| NN | VBZ | 16 | 100.0 | 14.7 | 109 |
| JJ | NNS | 62 | 98.4 | 75.3 | 81 |
| IN | NNS | 61 | 75.4 | 62.2 | 74 |
| NNS | VBD | 51 | 90.2 | 67.6 | 68 |
| NN | NN | 59 | 84.7 | 75.8 | 66 |
| NN | NNS | 42 | 88.1 | 60.7 | 61 |
| IN | VBD | 36 | 83.3 | 50.8 | 59 |
| RB | VBD | 2 | 100.0 | 3.6 | 56 |
| NN | NNP | 35 | 74.3 | 46.4 | 56 |
| NNS | VBP | 31 | 93.5 | 56.9 | 51 |
| NNP | VBD | 40 | 37.5 | 31.3 | 48 |
| CD | CD | 54 | 83.3 | 97.8 | 46 |
| RB | VBZ | 5 | 60.0 | 6.8 | 44 |
| NNP | VBZ | 40 | 37.5 | 34.1 | 44 |
| CD | NN | 49 | 73.5 | 83.7 | 43 |
| PRP | VBD | 40 | 87.5 | 85.4 | 41 |
| CD | IN | 72 | 38.9 | 70.0 | 40 |
| TO | VBD | 42 | 92.9 | 100.0 | 39 |
| PRP | VBZ | 45 | 86.7 | 100.0 | 39 |
| DT | NNS | 42 | 88.1 | 94.9 | 39 |
| MD | VB | 38 | 97.4 | 97.4 | 38 |
| TO | VB | 34 | 100.0 | 100.0 | 34 |
| CD | TO | 65 | 26.2 | 51.5 | 33 |
| CD | NNS | 33 | 78.8 | 78.8 | 33 |
| IN | NNP | 38 | 60.5 | 71.9 | 32 |
| IN | VBN | 32 | 84.4 | 93.1 | 29 |
| IN | VBZ | 33 | 42.4 | 73.7 | 19 |
| DT | VBZ | 78 | 16.7 | 100.0 | 13 |
| DT | IN | 82 | 4.9 | 100.0 | 4 |
| DT | VBD | 74 | 4.1 | 100.0 | 3 |

Table 3.3: Undirected precision and recall of the K&M, $\lambda = 10^{-2/3}$ trial of MAP/EM (the selected trial), by the unordered tag-pair type. Note that attachments in either direction and with the tags in either order are counted together. Tag-pair types with a count $\geq 30$ in the hypothesized annotation or the gold-standard are listed.

MAP/EM:

| length: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| hyp. count: | 2222 | 547 | 237 | 85 | 41 | 17 | 7 | 1 | 1 |
| | 838 1384 | 233 314 | 109 128 | 47 38 | 24 17 | 12 5 | 6 1 | 0 1 | 1 0 |
| precision: | 68.1 | 56.9 | 39.7 | 32.9 | 26.8 | 23.5 | 14.3 | 0.0 | 100.0 |
| | 64.1 34.5 | 19.3 47.8 | 14.7 43.0 | 19.1 36.8 | 12.5 23.5 | 0.0 20.0 | 0.0 0.0 | – 0.0 | 100.0 – |
| recall: | 77.2 | 44.7 | 33.1 | 24.1 | 20.0 | 13.3 | 8.3 | 0.0 | 100.0 |
| | 43.1 66.9 | 14.4 39.2 | 15.1 30.9 | 18.0 21.2 | 8.8 19.0 | 0.0 6.3 | 0.0 – | 0.0 – | 100.0 – |
| gold count: | 1960 | 696 | 284 | 116 | 55 | 30 | 12 | 4 | 1 |
| | 1246 714 | 313 383 | 106 178 | 50 66 | 34 21 | 14 16 | 11 1 | 3 1 | 1 0 |

ATTACH-RIGHT:

| length: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| hyp. count: | 3158 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3158 0 | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 | 0 0 |
| precision: | 62.1 | – | – | – | – | – | – | – | – |
| | 39.5 – | – – | – – | – – | – – | – – | – – | – – | – – |
| recall: | 100.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | 100.0 – | 0.0 – | 0.0 – | 0.0 – | 0.0 – | 0.0 – | 0.0 – | 0.0 – | 0.0 – |
| gold count: | 1960 | 696 | 284 | 116 | 55 | 30 | 12 | 4 | 1 |
| | 1246 714 | 313 383 | 106 178 | 50 66 | 34 21 | 14 16 | 11 1 | 3 1 | 1 0 |

Table 3.4: Precision and recall by string distance between parent and child, for the MAP/EM trial with $\lambda = 10^{-2/3}$, initialized with K&M (the selected trial). ATTACH-RIGHT is also shown. To be clear, a given column's precision and recall scores count only attachments (hypothesized or gold-standard) between words of a certain distance apart. The large-typeface numbers show the undirected attachment precision and recall of dependencies at different lengths. Small-typeface numbers give the directed attachment precision and recall for left children and right children, by length.

**Path Analysis**

Later in the thesis, we will treat the MAP/EM condition as a baseline against which other methods will be compared. While directed and undirected accuracy are perfectly reasonable ways to measure the quality of a dependency model, there is the nagging question of how much to penalize a model for getting directionality wrong.

Consider this simple example. Suppose we have the subsequence "*foams*/VBZ *at*/IN *the*/DT *mouth*/NN." The gold standard would probably posit that *foams*/VBZ is the parent of *at*/IN, *at*/IN is the parent of *mouth*/NN, and *mouth*/NN is the parent of *the*/DT. If the model wrongly makes *the* the parent of *mouth*, but correctly attaches the phrase (*the*/DT *mouth*/NN) as a child of *at*—by making *the* the child of *at*—then even the undirected attachment loss will still be worse by a point, since the gold standard does not posit a dependency in either direction between *at*/IN and *the*/DT.

We suggest a novel solution to this issue.[15] Rather than a simple 0–1 loss for attachments, we consider the tree distance from each tag to its correct parent.[16] Using a simple search algorithm, we can compute the length of the *directed* path from each tag $x_i$ to its gold-standard parent,[17] and we can also compute the length of the *undirected* path.[18] The former is said to have infinite length if it does not exist. In the case of the latter, which is always finite, we can compute a micro-average (by tags in the test corpus) and a macro-average (average over sentences of the mean path child-parent path length in the sentence). For the selected MAP/EM model, these statistics are given in Table 3.5.

## 3.5 EM with Log-Linear Grammars

This section considers EM for log-linear grammars. In this thesis we will not carry out EM directly on log-linear grammars with unconstrained weights, for computational reasons and because for Model A it is equivalent to EM on the *stochastic* grammar (as we will demonstrate in Section 3.5.1).

EM for weighted (log-linear) grammars is more computationally demanding than for

[15]Paths through parse trees have been measured for other purposes (Bunescu and Mooney, 2005; Swanson and Gordon, 2006).

[16]For the purposes of this measure, the root tag attaches to a special wall symbol, denoted \$ and placed by convention at the left side of the sentence.

[17]This path is unique in a directed tree, if it exists.

[18]This path always exists and is unique in a connected tree.

| path length | Attach-Right % of tag tokens undirected | Attach-Right % of tag tokens directed | MAP/EM % of tag tokens undirected | MAP/EM % of tag tokens directed |
|---|---|---|---|---|
| 1 | 55.3 | 36.0 | 64.9 | 47.3 |
| 2 | 20.0 | 9.6 | 25.2 | 21.1 |
| 3 | 9.8 | 5.0 | 7.1 | 5.8 |
| 4 | 5.3 | 3.6 | 2.0 | 1.3 |
| 5 | 3.1 | 2.5 | 0.7 | 0.5 |
| 6 | 2.8 | 2.4 | 0.0 | 0.0 |
| 7 | 1.8 | 1.8 | 0.0 | 0.0 |
| 8 | 1.1 | 1.1 | 0.0 | 0.0 |
| 9 | 0.6 | 0.6 | 0.0 | 0.0 |
| $\infty$ | – | 37.4 | – | 23.9 |

Table 3.5: Path analysis of the selected MAP/EM model. The macro-averaged undirected path length in hypothesized trees between a word and its gold-standard parent is 1.31; for Attach-Right it is 1.50. Note that attachment accuracy is *not* identical to the first line because here we count paths to the special wall symbol at the left side of the sentence (for these purposes, just another node in the tree), whereas our usual accuracy measures do not count attachments to the wall.

stochastic grammars. As in the stochastic case, dynamic programming (Inside and Outside) algorithms obviate the explicit representation of $q$. The sufficient statistics for the log-linear model are identical to those for the multinomial model in the case where the log-linear features $\vec{f}$ correspond exactly to the counts of grammar rules $\mathcal{R}$ in a derivation.[19]

The added expense comes in the M step. For log-linear models, there is no closed form for the complete-data maximum likelihood estimate. Finding the MLE requires solving a convex optimization problem:[20]

$$\max_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})} \left[\log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})\right] \equiv \min_{\vec{\theta}} \mathbf{H}_C \left(p_{\vec{\theta}} \middle\| \tilde{p} \cdot q\right) \tag{3.19}$$

Many methods exist (e.g., Benson and Moré, 2001), but they can take a long time to converge. We would prefer to avoid this doubly-nested optimization problem. One way to do this (not applied here), introduced by Dempster *et al.* (1977) and applied by Riezler (1999), is a *generalized* EM (GEM) algorithm. The idea is simple: on the M step, instead of *maximizing* Equation 3.19, with respect to $\vec{\theta}$, simply choose $\vec{\theta}^{(i+1)}$ such that $\mathbf{H}_C \left(p_{\vec{\theta}^{(i+1)}} \middle\| \tilde{p} \cdot q\right) \leq \mathbf{H}_C \left(p_{\vec{\theta}^{(i)}} \middle\| \tilde{p} \cdot q\right)$. In the case of a log-linear model, a single step down the cross-entropy hill (in the gradient direction and of the appropriate size, usually found by a line search, e.g., Armijo (1966)) satisfies this requirement. Another option is a single iteration of an iterative scaling algorithm (Darroch and Ratcliff, 1972). This eliminates the double loop.

The ability to compute the objective function value and gradient permits the application of standard numerical optimization techniques like gradient ascent, conjugate gradient, and quasi-Newton methods. While not necessarily simple to compute, the general

[19]In the more general case, with arbitrary features, we require the expected value of each feature's count. If these are no more coarse-grained than the grammar rules, then obtaining these expectations is not costly and can be done by bookkeeping in the dynamic program.

[20] The unconstrained optimization problem for *fully-observed-data* MLE with log-linear models turns out to be the dual of a constrained maximum entropy problem on a given feature set $\mathcal{F}$, namely:

$$\begin{array}{ll} \max_{\vec{\theta}} & \mathbf{H}\left(p_{\vec{\theta}}\right) \\ \text{such that} & \forall f \in \mathcal{F}, \mathbf{E}_{p_{\vec{\theta}}(\mathbf{W})}\left[f(\mathbf{W})\right] = \mathbf{E}_{\tilde{p}(\mathbf{W})}\left[f(\mathbf{W})\right] \end{array}$$

This is why log-linear models are often called "maximum entropy" models. The solution to the maximum entropy problem above is always the log-linear model that maximizes likelihood. See Ratnaparkhi (1997); Berger (1998) for a thorough discussion.

form of the gradient is:

$$\nabla_{\vec{\theta}}(-\mathbf{H}_C\left(p_{\vec{\theta}} \middle\| \tilde{p}\cdot q\right)) = \overbrace{\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\vec{f}(\mathbf{X},\mathbf{Y})\right]}^{\text{constant w.r.t. }\vec{\theta}\text{; sufficient statistics}} - \mathbf{E}_{p_{\vec{\theta}}(\mathbf{X},\mathbf{Y})}\left[\vec{f}(\mathbf{X},\mathbf{Y})\right] \quad (3.20)$$

Interestingly, the second derivative $\partial^2/\partial\theta_i\partial\theta_j$ is equivalent to a difference in *covariances*; a proof is omitted since we do not make use of second derivatives in this thesis. While GEM methods are not applied in this work, the form of the gradient is similar to a gradient we *do* require (in the next chapter).

### 3.5.1 Convergence of $\ddot{Z}_{\vec{\theta}}$

A problem still remains for weighted grammars. Note that we have assumed that $\vec{\theta}$ can take any value in $\mathbb{R}^n$ (for $n$ features). The trouble is that for some values of $\vec{\theta}$, the partition function $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ does not converge to a finite value. To see why not, consider the following log-linear CFG that generates sequences in $\mathcal{W} = x^+$:

$$\begin{array}{ll} \lambda_1 & S \rightarrow x \\ \lambda_2 & S \rightarrow S\ S \end{array}$$

If $\lambda_1 = \lambda_2 = 1$, then each of the infinitely many derivations receives score $\ddot{u}_{\vec{\lambda}} = 1$. Their summation, $\ddot{Z}_{\vec{\lambda}}(x^+)$, is infinite.

In this case the model is not well-formed, and the expected values of the features are undefined. Indeed, the solution to an M step (for example) cannot have this form; likelihood for any finite dataset approaches zero as $\ddot{Z}_{\vec{\lambda}}(\mathcal{W})$ approaches $+\infty$. The difficulty is that, while the maximum likelihood *solution* (on a given M step) will always have a convergent $\ddot{Z}_{\vec{\lambda}}(\mathcal{W})$, *finding* that solution may involve moving to infeasible $\vec{\lambda}$.

In the language of numerical optimization, there is a *constraint* on $\vec{\theta}$ ($\vec{\lambda}$), and we have a constrained optimization problem:

$$\begin{array}{rl} \max\limits_{\vec{\theta}} & \log \ddot{p}_{\vec{\theta}}(\vec{\mathbf{x}}^{\mathrm{t}}) \\ \text{such that} & \ddot{Z}_{\vec{\theta}}(\mathcal{W}) < \infty \end{array} \quad (3.21)$$

This kind of constraint expression, "$k(\vec{\theta})$ is finite," is not to be found in standard texts on constrained numerical optimization (Bertsekas, 1995; Nocedal and Wright, 1999). More typically, we see "$k(\vec{\theta}) = 0$" or "$k(\vec{\theta}) \leq 0$," possibly with $k$ a multivariate function and $\vec{\theta}$ including the variables of interest and perhaps additional variables to be optimized over. To convert the problem into the more standard optimization setting, we seek a sufficient (and ideally necessary) condition in the right form that forces $\ddot{Z}_{\vec{\theta}}(\mathcal{W}) < \infty$. Ideally this condition will restrict $\vec{\theta}$ to a closed, convex subset of $\mathbb{R}^n$, though there is no reason to believe that the feasible set *is* closed or convex. In fact, the set must be *open* if $\ddot{Z}_{\vec{\theta}}$ is continuous, since it is the inverse image of an open set (the finite reals) under the $\ddot{Z}_{\vec{\theta}}$.

An obvious solution is to consider the problem in terms of probabilities instead of log-probabilities, which effectively eliminates the constraint in Equation 3.21, since $\ddot{p}_{\vec{\theta}}$ goes to zero as $\ddot{Z}_{\vec{\theta}}$ goes to $+\infty$. If the constraint can be eliminated in the probability domain, then it can be eliminated in the log-probability domain as well (indeed, if at a particular value $\vec{\theta}$ the objective value is $-\infty$, corresponding to probability zero, then $\vec{\theta}$ cannot be a local maximum). To our knowledge, no optimization algorithm can guarantee that every iterate $\vec{\theta}^{(i)}$ will have finite $\ddot{Z}_{\vec{\theta}^{(i)}}$, since one of the iterative updates could always lead to parameters that are infeasible from the perspective of Equation 3.21. "Traveling" outside the feasible region creates a real problem for gradient-based methods, since the gradient (differences of feature expectations) are not well-defined where $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ is not finite. If the feature expectations and therefore the gradient are taken to be zero, then the parameters will be taken to have converged, which is incorrect. If they are taken to diverge, then we have an infinite gradient in some or all directions.

At the very least, a method for *detecting* whether $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ converges is required. This would allow us to enforce, on each iteration, a sufficiently small step to keep $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ finite. Solving for $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$, when it exists, typically involves solving a set of non-linear equations (linear equations in the case of finite-state models), subject to non-negativity constraints. Efficiently detecting that no (feasible) solution exists, usually just part of the job of the solver, may be a non-trivial problem by itself.

Importantly, this problem with MLE on log-linear models over unboundedly many structures is not specific to the hidden variable case. Even if the examples are fully observed, $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ is required for maximum likelihood estimation. While the problem is probably surmountable, most NLP practitioners prefer to avoid it. This is evidenced by the use of *conditional* estimation, a topic to which we will return in Section 4.1. In Chapter 4 we will present contrastive estimation, a solution that manages to avoid the issue and that has other fortunate advantages.

### 3.5.2 Viterbi/MCLE Approximation

An interesting attempt to combine EM with log-linear models, not explored experimentally in this thesis, can be found in Klein and Manning (2001a). The idea is to approximate the E step using the Viterbi approximation, and also replace the M step with maximum *conditional* likelihood estimation (MCLE). We will return to MCLE in Section 4.1; here we simply give the form of the objective:

$$\vec{\theta}_{\mathrm{MCLE}} \stackrel{\mathrm{def}}{=} \operatorname*{argmax}_{\vec{\theta}} p_{\vec{\theta}}(\vec{\hat{\mathbf{y}}} \mid \vec{\mathbf{x}}^{\mathrm{t}}) \tag{3.22}$$

MCLE deals only in conditional probabilities, so the all-events partition function $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ never needs to be computed and does not even need to be finite. It is replaced with $\ddot{Z}_{\vec{\theta}}(\{\mathbf{x}\} \times \mathcal{Y}_{\mathbf{x}})$, which is finite if $|\mathcal{Y}_{\mathbf{x}}|$ is finite. This is the most common supervised training method for log-linear models, and we will draw inspiration from this idea in Chapter 4.

This approach to estimation, in which a model is encouraged to become more confident about its current hypotheses, is sometimes called **self-training** (Nigam and Ghani, 2000) and recently achieved very good results in semi-supervised parsing with the Penn Treebank and two million additional unannotated sentences (McClosky, Charniak, and Johnson, 2006a). The danger is that the model will learn to generalize its own errors, performing worse as training proceeds.[21]

Together, the Viterbi and MCLE approximations do not correspond to an obvious objective function. The approximate E step in combination with the standard MLE M step *does* locally optimize an objective function that is related in an interesting way to likelihood (see Chapter 5). The approximate M step (MCLE) in combination with the standard E step corresponds to a flat objective function.

These approximations did give Klein and Manning the best published bracketing-grammar induction results up to 2001 (on the ATIS corpus). In later work, however, Klein and Manning (2002a) switched to a stochastic version of the model and replaced the estimation with standard MLE/EM, with substantial performance gains. A similar approach has been used in automatic speech recognition, notably by Gunawardana and Byrne (2001) for speaker adaptation (see also Leggetter and Woodland (1995) and Woodland, Pye, and

[21] **Co-training** is similar in spirit but uses two learners in different feature spaces to "teach" each other (Blum and Mitchell, 1998). Co-training and self-training are usually described as semi-supervised methods that start with supervised classifiers and use unannotated data to improve them.

Gales (1996) for comparable work in the MLE paradigm). Typically in speech recognition, a further approximation is made, in which a lattice or $k$-best list is used in lieu of all possible structures compatible with $\mathbf{x}$. That is, $p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x})$ is replaced by $p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}, \mathbf{Y} \in \mathcal{Y}')$ where $\mathcal{Y}' \subseteq \mathcal{Y}_{\mathbf{x}}$, a set of top hypotheses for $\mathbf{y}$ (rather than all).

### 3.5.3 Locally-Normalized Models

A hybrid alternative to log-linear grammars and classical multinomial-based grammars, not explored in this thesis, is to use a stochastic grammar in which the probability distributions are defined as log-linear models (for examples, see Charniak (2000), McCallum, Freitag, and Pereira (2000), and Eisner (2001)). This allows the incorporation of arbitrary features over the rules themselves.

To take an example, suppose we have a context-free grammar with rule set $\mathcal{R}$, nonterminal set $\mathcal{N}$, and terminal set $\Sigma$. The rules in a given distribution are of the form $\mathbf{n} \to \alpha$, where $\mathbf{n} \in \mathcal{N}$ and $\alpha \in (\mathcal{N} \times \Sigma)^*$. The stochastic grammar must define a probability distribution

$$p(\alpha \mid \mathbf{n}), \forall \alpha : (\mathbf{n} \to \alpha) \in \mathcal{R}$$

The multinomial parameterization of the model simply gives each such $\alpha$ some probability mass. The log-linear parameterization defines

$$p_{\vec{\theta}}(\alpha \mid \mathbf{n}) = \frac{\exp\left[\vec{\theta} \cdot \vec{f}(\mathbf{n} \to \alpha)\right]}{\ddot{Z}_{\vec{\theta}}(\mathbf{n})} = \frac{\exp\left[\vec{\theta} \cdot \vec{f}(\mathbf{n} \to \alpha)\right]}{\sum_{\alpha' : (\mathbf{n} \to \alpha') \in \mathcal{R}} \exp\left[\vec{\theta} \cdot \vec{f}(\mathbf{n} \to \alpha')\right]} \tag{3.23}$$

Features might include an indicator for $(\mathbf{n} \to \alpha)$, or anything of smaller granularity, such as an indicator for $\alpha$ alone or subsequences of $\alpha$. Of course, it is possible to define a log-linear distribution over infinitely many $\alpha$s, for example by making the above model a log-linear Markov model. In that case, the convergence problem returns.

Interestingly, the formulation above is equivalent to the usual log-linear parameterization in which an indicator feature function is added for $\mathbf{n}$ (the parent nonterminal). That feature function's corresponding weight is a function of the remaining feature weights.

$$\begin{aligned} f_{\text{Parent}=\mathbf{n}}(\mathbf{n}' \to \alpha) &= \begin{cases} 1 & \text{if } \mathbf{n} = \mathbf{n}' \\ 0 & \text{otherwise} \end{cases} \\ \theta_{\text{Parent}=\mathbf{n}}(\mathbf{n} \to \alpha) &= -\log \sum_{\alpha' : (\mathbf{n} \to \alpha') \in \mathcal{R}} \exp\left[\vec{\theta} \cdot \vec{f}(\mathbf{n} \to \alpha')\right] \end{aligned}$$

In this formulation, $\theta_{\text{Parent}=\mathbf{n}}$ is not a free parameter; it is a function of the other parameters. With a finite set of rules in $\mathcal{R}$, this approach has the advantage of allowing arbitrary (local fine-grained) features *without* requiring an explicit summation over $\mathcal{W}$. This is because, like classical SCFGs, this model forces $\ddot{Z}_{\vec{\theta}}(\mathcal{W}) = 1$ by design. The same trick can be applied to locally normalize markovized distributions, of course. A GEM algorithm or gradient descent method could be applied to locally optimize likelihood for this model.

### 3.5.4 Equivalence of Log-Linear and Stochastic CFGs

For many models, including Model A, all of this worry about convergence of $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ turns out to be unnecessary. Recall from Sections 2.3.1 and 2.3.2 that Model A is a stochastic CFG, which is a special case of a *log-linear* (weighted) CFG. In a recent result that extends some theorems by Chi (1999) and Abney, McAllester, and Pereira (1999), Smith and Johnson (in review) proved that, despite their extra degrees of freedom and relaxed constraints, log-linear CFGs cannot represent conditional distributions that stochastic CFGs cannot represent.

A log-linear context-free grammar with rules $\mathcal{R}$ and weights $\vec{\theta}$ always defines $p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})$ if $|\mathcal{Y}_{\mathbf{x}}|$ is finite. Smith and Johnson showed how any such log-linear CFG can be transformed into a *stochastic* CFG that defines exactly the same distribution with the same set of rules—just different weights. Hence the two classes of distributions are equally powerful for expressing distributions over derivation trees given yield sequences.

Indeed, the stochastic and log-linear models even define likelihood (and related probabilistic quantities) using the same function of the weights $\vec{\theta}$. The difference is that the former only defines the function in the portion of $\mathbb{R}^n$ where simplex constraints are met (Equations 2.3). Maximizing likelihood with either model, then, means optimizing the same function. For SCFGs, we add constraints to the problem.

It follows that to get a local maximum likelihood estimate for the log-linear grammar, we can carry out local maximum likelihood estimation (by EM as in Section 3.3, for example) on the *stochastic* version of the grammar. Any specialized method for optimizing over the distributions given by one class of models (log-linear or stochastic) automatically optimizes over the other, as well.

A limitation of this result is that, if the log-linear grammar includes fine-grained features of the production rules (as suggested in Section 3.5.3), those features' weights will not appear as free parameters in the stochastic CFG. In Model A, each production rule has exactly one parameter, so this is not a problem.

### 3.5.5 Gaussian Priors

Recall from Section 3.4 that maximum *a posteriori* estimation can be a useful way to prevent overfitting to the training data, using a prior over parameters. The most commonly used prior distribution, which is used in the next chapter, for the parameters of log-linear models is the Gaussian. It is parameterized by a vector of means $\vec{\mu}$ and a (symmetric, positive-definite) covariance matrix $\mathbf{\Sigma}$:

$$\pi_{\text{Gaussian}}(\vec{\theta}; \vec{\mu}, \mathbf{\Sigma}) = \frac{1}{Z(\mathbf{\Sigma})} \exp \left(\vec{\theta} - \vec{\mu}\right)^{\top} \mathbf{\Sigma}^{-1} \left(\vec{\theta} - \vec{\mu}\right) \tag{3.24}$$

(The normalizing term $Z(\mathbf{\Sigma})$ is equal to $\sqrt{(2\pi)^{|\mathcal{E}|}|\mathbf{\Sigma}|}$.) The most common application of the Gaussian prior sets all $\mu_{\mathbf{e}} = 0$ and $\mathbf{\Sigma} = \sigma \mathbf{I}$ for a fixed value $\sigma > 0$. The larger $\sigma$ is, the lighter the smoothing. This approach to smoothing log-linear models is due to Chen and Rosenfeld (2000) and has since been widely applied.

The term "regularization" is often used to refer to techniques that modify learning to prevent overfitting (as discussed in Section 3.4). This term was originally used in the neural network community. If fitting the training data corresponds to maximizing a function $f(\vec{\theta})$, then the $L_2$-regularized objective function is:

$$\max_{\vec{\theta}} f(\vec{\theta}) - c \sum_i \theta_i^2 \tag{3.25}$$

where $c > 0$ is a scalar constant that corresponds to the strength of the regularizer. If $f$ corresponds to the log-likelihood of the data under a log-linear model, the above is equivalent to MAP estimation with a diagonal Gaussian prior ($\vec{\mu} = \langle 0, 0, ..., 0 \rangle$ and $\mathbf{\Sigma} = \sqrt{c}\mathbf{I}$).

We noted in the last section that MLE on stochastic grammars and log-linear grammars was equivalent if the features in the latter were counts of the grammar rules in the former. Adding a prior changes things. For example, Dirichlet-smoothed SCFGs and Gaussian-smoothed log-linear CFGs will behave differently.

Although they are not directly applied in this thesis, readers interested in alternative smoothing/regularization methods for log-linear models are referred to Kazama and Tsujii (2005) and Goodman (2004), which give a thorough presentation of some newer techniques that give empirically good performance.

## 3.6 Related Work

Partial-data MLE and EM have been widely used in NLP. We reviewed work on dependency grammar induction in Section 2.3.4. Here we note some other important papers that used EM for unsupervised language learning in various settings.

### 3.6.1 EM for Grammars

MLE for stochastic CFGs from sequences alone, with parse structure taken as the hidden variable $\mathbf{Y}$, was applied to constituent-based models by Lari and Young (1990), word dependency models by Carroll and Charniak (1992), and to stochastic CFGs with partially-bracketed text by Pereira and Schabes (1992). These attempts met with mixed success at best, certainly in part due to the usual problems with the EM algorithm (local optima and disconnect between likelihood and accuracy) that this thesis aims to address. Klein and Manning (2004) note that Charniak and Carroll's model performed poorly in part due to random initialization, and that better performance is possible with a better initializer. An excellent discussion of these difficulties can be found in de Marcken (1995a).

In later work, Hwa (1999) found that Pereira and Schabes' partial-bracketing approach could be successful when *adapting* a probabilistic grammar from one domain to another, and that if only some data were to be labeled in this framework, higher-level constituents were more helpful for learning (cf. Pereira and Schabes, 1992). This is consistent with our finding (in Chapter 6) that guiding a learner to focus on local structure first, before attempting to learn long-distance dependencies, can improve performance.

Klein and Manning (2001a) described a generative probabilistic (log-linear) model of sentences given binary bracketings in which the yield (a part-of-speech tag sequence) of each constituent and its context (the tags on either side) are independently chosen. The "yield" and context of each *non-constituent* subsequence is also predicted in a similar way. Klein and Manning (2002a) made the model generative (with greatly improved results, although the model was quite deficient). These *constituent-context models* (CCM) were then combined with the dependency model we call Model A, during training and decoding, to achieve the best reported (to date) unlabeled PARSEVAL scores for English, German, and Mandarin grammar induction on the ATIS corpus and the WSJ-10 corpus.[22]

[22]I.e., the sentences in the Penn Treebank of ten words or fewer, after the removal of punctuation punctuation; the set from which corpora used in experiments in this chapter were drawn.

Merialdo (1994) explored the possibility of using some part-of-speech tagged text to initialize a trigram tag model (second order HMM), followed by iterations of EM on unlabeled text. He found that the latter generally failed to improve the initial model's tagging accuracy. This is consistent with our findings in this chapter (and many others); similar results were achieved by Elworthy (1994).

Clark (2001) addressed morphology induction using pair hidden Markov models (PHMMs; Durbin, Eddy, Krogh, and Mitchison (1998)) to model transduction between root and morphologically-inflected forms of words. Rather than addressing unsupervised morphology learning directly, he argued that word categories (such as "past tense verb") could be learned without supervision (a problem he also addressed) and that morphology could be reduced to finding a matching between roots and inflected forms.[23] Clark used the EM algorithm to estimate parameters for his PHMMs and posteriors over root/inflected form matchings; the models were applied successfully to English past tenses but met with less success on Arabic.

### 3.6.2 Minimum Description Length

Estimation is only part of unsupervised learning; the structure of the model—in our case, the rules in the grammar—must either be specified from the start (as we have done) or learned. A typical strategy is to allow *all* possible rules (provided only a finite number can be specified, as in Model A but not in unbinarized CFGs). This leaves it to the estimation algorithm to "weed out" rules that aren't needed, by giving them zero or nearly-zero probability (or, in the log-linear case, weights approaching $-\infty$). Approaches to the direct learning of model structure have generally employed the minimum description length principle.

The minimum description length (MDL) principle (Rissanen, 1978) is a kind of learning in which a bias toward simpler models is integrated into the objective function. Rather than simply finding the model to maximize the probability of the data, MDL seeks the model to minimize the number of bits required to describe the model plus the number of

[23]Clark looked for a matching in the formal sense: he viewed the root forms and inflected forms as a bipartite graph, and he sought the best matching, except that the weights on the edges were not given explicitly. They were given by PHMMs, whose parameters must be learned. His approximate approach to dealing with the seemingly intractable problem (inside EM) of summing over all possible weighted matchings of a bipartite graph (equivalently, computing the permanent of a matrix) can be contrasted with the greedy approximation (competitive linking) of Melamed (2000) and the Viterbi approximation to EM—via solving the maximum weighted bipartite matching problem—of Smith (2002).

bits required to describe the data given the model. Since under optimal coding conditions, the number of bits required to describe an event is equal to the negative log of the event's probability (see, for example, Cover and Thomas, 1991), MDL equates to:

$$\max_{\text{model}} \log p_{\text{model}}(\text{data}) + \log p(\text{model}) \tag{3.26}$$

where the probability of the model is inversely proportional to its complexity. It should be noted that the MDL criterion does not inherently admit a good search procedure or a good prior "model over models," and approaches to MDL generally involve non-optimal search. MDL is essentially equivalent to MAP estimation (Section 3.4), with the selection of a prior that prefers simpler models (cf. the Dirichlet used in Section 3.4, which does not care about simplicity).

In the experiments of this thesis, we have always assumed that parameter search takes place for a fixed model *structure*. For weighted grammars, this means a fixed set of rules. For log-linear models, this means a fixed feature set, and in our case we used all possible rules (since there are only finitely many in Model A for a fixed $\Sigma$). MDL approaches tend to be associated with the search for the set of rules themselves.

For example, several researchers have applied MDL to problems in morphology and lexicon induction. Brent (1993) used MDL to discover suffixes in a corpus. de Marcken (1995b) went farther, seeking to segment Chinese and (whitespace-purged) English corpora by finding character subsequences to add to a lexicon (using an MDL criterion). Goldsmith (2001) used the MDL criterion and heuristic search to discover the analyses of English words that annotators might select. Kazakov (1997) used an MDL criterion with genetic algorithms for search to find "naïve" morphology, or word segmentations.

Chen (1995) applied MDL to the search for grammar structure, beginning with an extremely simple grammar capable of generating the entire (unlabeled) training corpus. First, he considered in turn each sentence and rules that might be added to the grammar to modify the parse of the sentence. His algorithm greedily considers adding each such rule to the grammar, accepting it if it improves the overall likelihood objective. After one pass over the corpus, he ran the EM algorithm to improve the parameter weights. Chen reported perplexity gains over an $n$-gram model and the inside-outside algorithm without structure learning. He did not evaluate the parses produced against a gold standard, since his goal was to improve language modeling.

Stolcke and Omohundro (1994) used a similar approach, but their search began with a model that described the full data set in detail (without generality). The (greedy) search proceeded by merging nonterminals (or, in the HMM case, states) with similar production rule properties. and creating new nonterminals to rewrite to short sequences of nonterminals with a high affinity for each other. Stolcke and Omohundro (1994) applied this algorithm with success to a variety of toy problems.

### 3.6.3 EM with Knowledge Sources

As noted in the last section, a common approach is to assume that the model structure is known, perhaps from an expert. Beil, Carroll, Prescher, and Rooth (1999), for example, started with a known grammar (a lexicalized CFG), and used EM to estimate parameters. Riezler, Prescher, Kuhn, and Johnson (2000) also did this for a lexical-functional grammar, Osborne and Briscoe (1997) for a stochastic categorial grammar, and Carroll and Rooth (1998) started with a hand-built lexicalized grammar and used EM to learn subcategorization frames.

Charniak (2001) applied the EM algorithm to learning the internal structure of names (i.e., segmentation into descriptors, honorifics, first, middle, and last names, etc.) using possible (hidden) coreference relations. In other words, candidate previous mentions of the same named entity are used to help constrain the possible labelings.

A related technique involves exploiting *unambiguous* instances in data for estimation. Hindle and Rooth (1993) and Ratnaparkhi (1998) used the unambiguous attachments in part-of-speech-tagged, NP-bracketed text to estimate a model for predicting attachment of prepositional phrases (PPs) to candidate nouns and verbs. The model was then applied to PPs whose attachment was ambiguous; no iterative training was done. Similar work for coordinated nouns inside a PP was done by Goldberg (1999). This approach relies on knowledge of which examples are ambiguous and which are not (in these cases annotation is not needed for that knowledge). Brill (1995) used this kind of approach, albeit within a transformation-based learning framework rather than in a model, for unsupervised part-of-speech tagging.

## 3.7 Future Work

In this chapter we have presented one approach to the problem of selecting *hyperparameters* (here, the initializer $\vec{\theta}^{(0)}$ and the smoothing $\lambda$): train with a variety of hyperparameter values and use development data to select a model (supervisedly or unsupervisedly). This is computationally expensive and not practical when there are many hyperparameter settings to choose from, as we will see in later chapters. How are we to know, for instance, that selecting a different value for $\lambda$—one we did not test yet—would not further improve our performance? In the future, great computational savings would be provided by a method that automatically searched for good hyperparameter settings (that is, optimized parameters like the smoothing $\lambda$) without having to train a huge number of models to completion. This kind of technique would benefit all of the methods presented in this thesis.

Under the assumption that some quantity of annotated data is available, it remains an open question how it should be used. Here we use it primarily to select among a set of unsupervisedly trained models (see Section 3.3.2), but we also explored its use in building initializers and priors for MAP/EM training (see Section 3.4.2). The initialization idea is originally due to Merialdo (1994). There is now a huge literature in the area called "semisupervised" learning (see, e.g., Yarowsky, 1995; Blum and Mitchell, 1998; Ando and Zhang, 2005).

Another line of future work is the exploration of online algorithms (such as online-EM) in combination with hyperparameter choices. Throughout the thesis we use a learning setting where a fixed set of tagged sentences are used as training data, but the reality is that new natural language data is generated every day. Learning algorithms should be able to continue learning as long as more data is available. This is certainly possible with EM (see Section 3.3.4).

A limitation of the experiments presented here (and throughout the thesis) is that we do not train or test on sentences of longer than ten words. Longer sentences are likely to contain syntactic constructions that short sentences do not. The main reason for this is the computational expense. The Inside-Outside algorithm has cubic runtime in the sentence length, so EM (and other iterative methods with similar subroutines, as in later chapters) can be run on short sentences and converge in a matter of hours (on a modern computer). This will not scale up well to long sentences; we suggest some solutions.

The first is to use a simpler model, such as vine grammar (Eisner and Smith, 2005), that has faster inference algorithms. Vine grammars can be designed to have similar features to Model A, but they permit an asymptotic speedup in the dynamic programming algorithms by constraining the trees (specifically, by disallowing long dependencies). A second solution is to use an incremental learning algorithm (see Section 3.3.4), which may give faster convergence. A third option is to use a small training set for estimation, but incorporate distributional features learned from a larger, unannotated corpus. In grammar induction, for example, we might gather cooccurrence statistics from a very large corpus and use them as features of potential attachments. Features of this kind were used by Taskar, Lacoste-Julien, and Klein (2005) for word alignment for machine translation. To do that, we would need a model that let us incorporate arbitrary features of the data (e.g., a log-linear model).

## 3.8 Summary

The most widely applied unsupervised learning method is the Expectation-Maximization algorithm.[24] Though in most textbooks it is mentioned in the same breath as Gaussian mixture models or other continuous data clustering approaches, EM is a very broadly applicable technique. We have described how it can be applied to stochastic grammars and shown its performance, as a baseline, at uncovering linguistic dependency structure, both in maximum likelihood estimation and maximum *a posteriori* estimation. EM suffers from several difficulties, illustrated in this chapter, which this thesis aims to address. The first is that the function maximized by EM—likelihood of the data or, with a prior, the posterior probability of the model—is not a good predictor of accuracy. The second problem is that, as a local hillclimber, EM is highly sensitive to the initialization method. Good initializers appear to be crucial to its success, but very difficult to find.

The dependency grammar induction task to which EM was applied in this chapter will continue to be the focus of application in the remainder of the thesis.[25] The next two chapters present novel estimation techniques that seek to address each of these problems in turn by changing the objective criterion (Chapter 4) and changing the search (Chapter 5).

[24]At the forty-third annual meeting of the Association for Computational Linguistics in Barcelona in 2004, around 15% of the papers presented included some application of an EM algorithm.

[25]The reader is reminded that this task, while difficult, is somewhat artificial in its definition and evaluation criteria (essentially, to match human annotators). See Section 1.4.

|  | test accuracy |  |
|---|---|---|
|  | directed | undirected |
| ATTACH-RIGHT | 39.5 | 62.1 |
| MLE/EM (s-sel.) | 41.7 | 62.1 |
| MAP/EM (s-sel.) | 41.6 | 62.2 |

Table 3.6: Summary of results in this chapter.

We will see that both approaches can lead to improved accuracy. The following chapter, Chapter 6, will borrow ideas from both techniques and achieve even better performance.

# Chapter 4

# Contrastive Estimation

*Use the word* cybernetics*, Norbert, because nobody knows what it means. This will always put you at an advantage in arguments.*

—Claude Shannon (1916–2001)

*Where there is much light, the shadow is deep.*

—Johann Wolfgang von Göthe (1749–1832; *Götz von Berlichingen*, 1773)

In this chapter we present **contrastive estimation** (CE), a novel technique for estimating the parameters of models of hidden structure. The technique addresses two of the main difficulties encountered in applying MLE to log-linear models. First, CE provides a way to estimate models over infinitely many structures (such as log-linear grammars) without summing over all those structures (as would be required for MLE). (Recall from Section 3.5.1 that the possible divergence of $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ poses a problem for estimating such models.) Second, CE provides a mechanism for injecting a particular kind of general domain knowledge into the learner.

This chapter applies CE to Model A, but we emphasize that the idea is potentially much more broadly applicable. Wherever log-linear models are used in structured modeling, CE may be helpful. To our knowledge, this is the first principled approach to unsupervised learning with log-linear models over infinite discrete spaces. The experimental results in this chapter, while substantially better than those achieved by EM, are not the best in the thesis. This should not be taken as a flaw in contrastive estimation, but rather in our creativity in developing appropriate instances of it (neighborhood functions, defined below) for this task.

CE was originally introduced and applied with great success to part-of-speech tagging in Smith and Eisner (2005a) and to weighted grammar induction in Smith and Eisner (2005b). These results are presented here, with a focus on grammar induction (the tagging experiments are in Section 4.7). This chapter goes beyond those papers in more carefully exploring the effect of initializers and regularization and additional neighborhood functions for grammar induction. The exposition is more leisurely, and we provide more substantial error analysis and apply CE to a new model.

## 4.1 Maximum Conditional Likelihood Estimation

We begin by reviewing an estimation criterion, maximum *conditional* likelihood, appropriate for *supervised* learning of models to be used in classification. Recall that $\vec{\mathbf{x}}^{\mathrm{t}}$ refers to our training data (hence the $\mathrm{t}$). Because MCLE is a supervised method, it also has access to the correct hidden structures $\vec{\mathbf{y}}^{\mathrm{t}}$ for the training examples. The objective function is:

$$\vec{\theta}_{\mathrm{MCLE}} \stackrel{\mathrm{def}}{=} \operatorname*{argmax}_{\vec{\theta}} p_{\vec{\theta}}(\vec{\mathbf{y}}^{\mathrm{t}} \mid \vec{\mathbf{x}}^{\mathrm{t}}) \tag{4.1}$$

This technique is "discriminative," meaning that it does not attempt to model the input/observed variable $\mathbf{X}$ at all.[1] For models of discrete structure, this approach has been notably applied by Lafferty *et al.* (2001), who trained a log-linear bigram HMM for part-of-speech tagging. In that case each $\mathbf{x}^{\mathrm{t}}$ is a sequence of words, and each $\mathbf{y}^{\mathrm{t}}$ is a sequence of part-of-speech tags. They call the model a **conditional random field** (CRF).

Since the original paper, CRFs have been very widely applied to sequence labeling tasks in information extraction (e.g., McCallum and Li, 2003) and shallow natural language processing (e.g., Sha and Pereira, 2003). The same idea was applied to parsing with a log-linear CFG by Miyao and Tsujii (2002) and more recently to reranking of parses by Charniak and Johnson (2005).

There are three advantages to MCLE over MLE:

1. When the model has a log-linear parameterization, MLE requires the iterative computation of $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$, the sum of scores of all structures allowed by the model.[2] This term might diverge; see Section 3.5.1. MCLE deals only in *conditional* probabilities

[1]Minka (2005) argues for thinking of this as a joint model where we simply do not estimate the marginal distribution $p(\mathbf{X})$.

[2]The double-dot notation merely reminds us that the model is log-linear.

and therefore requires only $\ddot{Z}_{\vec{\theta}}(\mathbf{x}^t)$ to normalize the distribution $\ddot{p}_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{x}^t)$ for each training example. Related quantities needed for estimation are feature expectations under the model. For this, MLE requires $\mathbf{E}_{\ddot{p}_{\vec{\theta}}(\mathbf{X},\mathbf{Y})}[f_i(\mathbf{X},\mathbf{Y})]$, while MCLE requires $\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot\ddot{p}_{\vec{\theta}}(\mathbf{Y}|\mathbf{X})}[f_i(\mathbf{X},\mathbf{Y})]$.

2. Because those quantities are straightforward to compute for many log-linear models using Inside algorithms, arbitrary features of the data, examples $\mathbf{x}$ and $\mathbf{y}$, can be incorporated.[3] This is not true of MLE with classical stochastic models.

3. MCLE is *discriminative* rather than *generative*. That means that the estimation method is designed to assign the correct $\mathbf{y}$ to each $\mathbf{x}$, rather than model $\mathbf{x}$ and $\mathbf{y}$ together. For interesting comparisons in the NLP domain, see Johnson (2001) and Klein and Manning (2002b).

Unfortunately, MCLE cannot be directly applied when learning from $\vec{\mathbf{x}}^t$ alone (without $\vec{\mathbf{y}}^t$). To see why, consider that the hidden structures $\mathbf{y}^t$ must be known if we want to compute the quantity to be maximized. If we marginalize over them in the numerator (which is what MLE does), we end up with a ratio where the numerator and denominator are equal—such an objective function is flat and not interesting to optimize. Contrastive estimation is inspired by the MCLE approach, but aims to achieve the benefits listed above in the partial data setting.

## 4.2 Implicit Negative Evidence

Why, intuitively, does discriminative modeling outperform generative modeling? If estimation of a model is in service of a particular task (here, structural classification) then there is no strong reason to model $\mathbf{x}$ (the input), since it will always be given to the classifier. Generative approaches model both $\mathbf{x}$ and $\mathbf{y}$, as we have seen throughout the thesis. Discriminative modeling only estimates what is needed to make the decision among different $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}$.[4] One of the most compelling arguments for (certain) discriminative methods is that they have provable error generalization bounds.

[3]As noted earlier, the more local the features are, the easier they are to incorporate. Features that consider large substructures $\langle \mathbf{x}, \mathbf{y} \rangle$ can result in more expensive inference and decoding algorithms.

[4]For an extensive discussion of generative and discriminative approaches to machine learning, see Jebara (2003), who aims to unify the best of both worlds.

A word of warning: the form of the model and the estimation criterion should be considered orthogonal, modulo computational efficiency. A stochastic grammar, for instance, can in principle be estimated using MCLE (discriminative) or MLE (generative), as can a log-linear grammar. It is for computational and historical reasons that stochastic models tend to be associated with MLE and log-linear models with MCLE. It is also worth mentioning that many other discriminative approaches exist and have been applied to different kinds of weighted grammars, including the perceptron (Collins, 2002), support-vector machines (Altun, Johnson, and Hofmann, 2003), and maximum margin methods (Taskar *et al.*, 2004; McDonald *et al.*, 2005a).

In our probabilistic setting, we can think of MCLE as manipulating the distributions $p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{x}_i^{\mathrm{t}})$. Since we know the correct $\mathbf{y}_i^{\mathrm{t}}$, the goal is to "move" probability mass from other $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}$ onto $\mathbf{y}_i^{\mathrm{t}}$.[5] This must be done by manipulating the model parameters, and it is carried out by solving the numerical optimization problem shown in Equation 4.1. The "other" $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}} \setminus \{\mathbf{y}_i^{\mathrm{t}}\}$ are treated as a class of **implicitly negative** instances that should be made improbable (given $\mathbf{x}_i^{\mathrm{t}}$) to maximize the objective function.

Because the log-linear model has a parameterized form—it does not simply enumerate structures in $\mathcal{W}$ and map them to probabilities—moving probability mass must be done by manipulating the parameters $\vec{\theta}$. Changing a particular $\theta_i$ will have an effect on some examples, and the goal of estimation is to change *all* $\theta_i$ together so that the net effect is the discrimination between each $\mathbf{y}_i^{\mathrm{t}}$ and its (incorrect) competitors.

Turning to the partial data setting, we want, analogously, to specify a class of negative instances. The central idea in contrastive estimation is to use knowledge of the domain and the intended task of the model to *design* such a class of "implicit negative evidence" for each example $\mathbf{x}^{\mathrm{t}}$. Let $\mathcal{N} : \Sigma^* \to 2^{\Sigma^*}$ be a function that maps every possible observed structure to a set of structures in $\Sigma^*$. Then contrastive estimation seeks:

$$\vec{\theta}_{\mathrm{CE}} \stackrel{\mathrm{def}}{=} \operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{|\vec{x}^{\mathrm{t}}|} p_{\vec{\theta}}(\mathbf{X} = \mathbf{x}_i^{\mathrm{t}} \mid \mathbf{X} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})) = \operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{|\vec{x}^{\mathrm{t}}|} \frac{p_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}})}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} p_{\vec{\theta}}(\mathbf{x})} \tag{4.2}$$

CE tries to move probability mass from each example's negative evidence class onto the

[5] More generally, we can say that in the training data, for a given $\mathbf{x}^{\mathrm{t}} \in \Sigma^*$, we may have observed more than one hidden structure—the training data provides a *distribution* over hidden structures for each $\mathbf{x}^{\mathrm{t}}$, which the learner is to try to match.

example itself. For log-linear models, this becomes:

$$\max_{\vec{\theta}} \prod_{i}^{|\vec{x}^{\mathrm{t}}|} \frac{\ddot{p}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}})}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \ddot{p}_{\vec{\theta}}(\mathbf{x})} \tag{4.3}$$

$$\equiv \max_{\vec{\theta}} \prod_{i}^{|\vec{x}^{\mathrm{t}}|} \frac{\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}} \ddot{p}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{p}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})} \tag{4.4}$$

$$\equiv \max_{\vec{\theta}} \prod_{i}^{|\vec{x}^{\mathrm{t}}|} \frac{\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}} \frac{\ddot{u}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})}{\ddot{Z}_{\vec{\theta}}(\mathcal{W})}}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \frac{\ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}{\ddot{Z}_{\vec{\theta}}(\mathcal{W})}} \tag{4.5}$$

$$\equiv \max_{\vec{\theta}} \prod_{i}^{|\vec{x}^{\mathrm{t}}|} \frac{\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})} \tag{4.6}$$

$$\equiv \max_{\vec{\theta}} \prod_{i}^{|\vec{x}^{\mathrm{t}}|} \frac{\ddot{Z}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}})}{\ddot{Z}_{\vec{\theta}}(\mathcal{N}(\mathbf{x}_i^{\mathrm{t}}))} \tag{4.7}$$

Like MCLE, CE deals only in conditional probabilistic quantities, and the $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ terms cancel out. Even if $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$ is infinite, the above quantity can be defined and manipulated. Crucially, if the summations in each numerator and denominator are all over *finitely* many structures—that is, $\forall \mathbf{x}$, $|\mathcal{Y}_{\mathbf{x}}| < \infty$ and $|\mathcal{N}(\mathbf{x})| < \infty$—then there are no worries about convergence and standard unconstrained optimization techniques can be applied on $\vec{\theta}$.

Apart from its computational properties, CE makes explicit an important distinction for structured learning. There may be more than one *kind* of hidden structure latent in the training data. Choosing an implicit negative evidence class corresponds to choosing what kind of structure the learner should find, by deciding what the learner is supposed to explain in the data.

## 4.3 Neighborhoods

Having seen the general form of contrastive estimation, we turn now to the function $\mathcal{N}$. We find it useful to think of $\mathcal{N}$ as a perturbation function that returns a set of "nearby" examples that are, in some sense, deprecated or damaged. This forces the model to discriminate between observed $\mathbf{x}$ and its perturbed neighbors that, given our understanding of the domain, should be unlikely. From here on, we call $\mathcal{N}$ a **neighborhood** function.

Suppose we wish to estimate a model of the syntax of the language (for example, Model A). If we want the syntactic model to "explain" the observed ordering of words in each sentence, the neighborhood of a sentence $\mathbf{x}$ might be taken to be the set of permutations of the elements in $\mathbf{x}$: $x_1, x_2, ..., x_{|\mathbf{x}|}$. This would encourage the model to use the hidden structure to explain the ordering of the sentence.[6]

## 4.3.1 Selective Perturbation

Another way to think about syntax learning with CE is to imagine a scenario in which the learner observes an utterance $\mathbf{x}$ and understands what it means, without having a fully-formulated syntax model. If the surface form $\mathbf{x}$ and the meaning are known, then the inferred syntax should be chosen so as to make $\mathbf{x}$ the best choice among competing surface forms for the known meaning. So the implicit negative examples should be *like* $\mathbf{x}$ in that, if observed, they would lead to the same meaning, but *different* from $\mathbf{x}$ in that they presumably violate syntactic well-formedness and should be made unlikely under the model. The goal is to perturb syntactic quality without affecting semantics.

If we know of an operation $\Sigma^* \rightarrow \Sigma^*$ or $\Sigma^* \rightarrow 2^{\Sigma^*}$ that *damages* syntax but *preserves* semantic interpretation, CE will allow us to use it to aid syntax learning. Perhaps surprisingly, no explicit representation of semantics is needed! In this thesis, our perturbation operations for building $\mathcal{N}$ are relatively crude but sometimes surprisingly successful.

The neighborhood function allows us to inject into the learner systematic domain knowledge of the following form: our observation of $\mathbf{x}$ would make us surprised to observe examples in the set of perturbations of $\mathbf{x}$, $\mathcal{N}(\mathbf{x}) \setminus \{\mathbf{x}\}$.[7] When selecting such a perturbation in the observable space ($\Sigma^*$ in this thesis), we guide the learner to structure in the *hidden* structural domain ($\mathcal{Y}$) by requiring that it explain, via hidden structure, why the perturbation damages the data.

[6]There are up to $|\mathbf{x}|!$ distinct reorderings of $\mathbf{x}$, so there may be computational problems with this approach; we will return in Section 4.5 to the question of efficiency.

[7]In this thesis, $\mathcal{N}(\mathbf{x})$ always includes $\mathbf{x}$ itself, making the quantity to be maximized a conditional probability.

### 4.3.2 Selective Destruction

Here we give another presentation of CE, suggested by Dale Schuurmans (p.c.). Suppose our observed random variable, $\mathbf{X} \in \mathcal{X} = \mathcal{X}_c \times \mathcal{X}_s$, breaks into two sub-events, $\mathbf{X}_c \in \mathcal{X}_c$ (implicit content) and $\mathbf{X}_s \in \mathcal{X}_s$ (implicit syntax). While $\mathbf{X}$ is observable, it is not easy to discern $\mathbf{X}_c$ and $\mathbf{X}_s$. MLE maximizes:[8]

$$p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s) = \frac{p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s)}{\sum_{\mathbf{x}'_c, \mathbf{x}'_s} p_{\vec{\theta}}(\mathbf{x}'_c, \mathbf{x}'_s)} \tag{4.8}$$

CE, on the other hand, uses a neighborhood function $\mathcal{N}$ that preserves $\mathbf{X}_c$ but obliterates $\mathbf{X}_s$. Ideally, this neighborhood only varies the implicit syntax (not the implicit content) of the message:

$$\mathcal{N}(\mathbf{x}) = \mathcal{N}(\mathbf{x}_c, \mathbf{x}_s) = \{(\mathbf{x}_c, \mathbf{x}'_s) : \mathbf{x}'_s \in \mathcal{X}_s\} \tag{4.9}$$

Then CE maximizes:

$$p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s \mid \mathcal{N}(\mathbf{x}_c, \mathbf{x}_s)) = \frac{p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s)}{\sum_{\mathbf{x}'_s} p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}'_s)} = p_{\vec{\theta}}(\mathbf{x}_s \mid \mathbf{x}_c) \tag{4.10}$$

So to the extent that $\mathcal{N}(\mathbf{x})$ expands alternative syntactic analyses without altering the implicit content, CE will (unlike MLE) avoid modeling content and more appropriately model just syntax. Here we are concerned with a *joint* model $p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})$. Where $\mathbf{Y}$ is a hidden variable that is supposed to capture syntax and not content. Dividing $\mathbf{X}$ into $\mathbf{X}_c$ and $\mathbf{X}_s$ as before, we see that MLE maximizes:

$$p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s) = \frac{\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s, \mathbf{y})}{\sum_{x' = (x'_c, \mathbf{x}'_s)} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}'}} p_{\vec{\theta}}(\mathbf{x}'_c, \mathbf{x}'_s, \mathbf{y}')} \tag{4.11}$$

But CE still maximizes:

$$p_{\vec{\theta}}(\mathbf{x}_c, \mathbf{x}_s \mid \mathcal{N}(\mathbf{x}_c, \mathbf{x}_s)) = p_{\vec{\theta}}(\mathbf{x}_s \mid \mathbf{x}_c) = \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} p_{\vec{\theta}}(\mathbf{x}_s, \mathbf{y} \mid \mathbf{x}_c) \tag{4.12}$$

so that $\mathbf{Y}$'s distribution will be manipulated to maximize syntax given content, matching $\mathbf{X}_s$ given $\mathbf{X}_c$.

[8] In this section we leave out the product over the training dataset, for clarity.

Generally speaking, if we wanted to design $\mathbf{Y}$ to maximize $p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})$ where the marginal $p(\mathbf{X})$ was fixed, then the solution is $\mathbf{Y} = \mathbf{X}$. In our case, the marginal is not fixed; it is what we want to maximize, and the generative model can be stated as $p_{\vec{\theta}}(\mathbf{Y})p_{\vec{\theta}}(\mathbf{X} \mid \mathbf{Y})$. Assuming, as we do here, that both factors can be manipulated, the best solution to maximizing $p(\mathbf{X})$ is still to set $\mathbf{Y} = \mathbf{X}$. Doing so will make $p_{\vec{\theta}}(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}) = 1$ when $\mathbf{x} = \mathbf{y}$, or 0 when $\mathbf{x} \neq \mathbf{y}$, and $p_{\vec{\theta}}(\mathbf{Y}) \equiv p_{\vec{\theta}}(\mathbf{X})$ will be learned to match the empirical distribution $\tilde{p}(\mathbf{X})$.[9]

Similarly, if the goal is to maximize $p(\mathbf{Y}, \mathbf{X}_s \mid \mathbf{X}_c)$, where the marginal $p(\mathbf{X}_s \mid \mathbf{X}_c)$ is fixed, then the best solution is $\mathbf{Y} = \mathbf{X}_s \mid \mathbf{X}_c$. Here the marginal is not fixed; it is what we want to maximize, and the generative model can be stated as $p_{\vec{\theta}}(\mathbf{Y})p_{\vec{\theta}}(\mathbf{X}_c \mid \mathbf{Y})p_{\vec{\theta}}(\mathbf{X}_s \mid \mathbf{X}_c, \mathbf{Y})$.

The solution $\mathbf{Y} = \mathbf{X}_s$ given a fixed value for $\mathbf{X}_c$ leads to the following equalities:

$$\begin{aligned}
p(\mathbf{y}) &= \sum_{\mathbf{x}_c} p(\mathbf{x}_c)p(\mathbf{y} \mid \mathbf{x}_c) \\
&= \sum_{\mathbf{x}_c} p(\mathbf{x}_c)p(\mathbf{x}_s \mid \mathbf{x}_c) && (4.13) \\
p(\mathbf{x}_c \mid \mathbf{y}) &= \frac{p(\mathbf{y} \mid \mathbf{x}_c)p(\mathbf{x}_c)}{p(\mathbf{x}_s)} \\
&= \frac{p(\mathbf{x}_s \mid \mathbf{x}_c)p(\mathbf{x}_c)}{\sum_{\mathbf{x}_c} p(\mathbf{x}_c)p(\mathbf{y} \mid \mathbf{x}_c)} && (4.14) \\
p(\mathbf{x}_s \mid \mathbf{x}_c, \mathbf{y}) &= p(\mathbf{y} \mid \mathbf{x}_c, \mathbf{y}) \\
&= 1 && (4.15)
\end{aligned}$$

Therefore the objective function

$$\begin{aligned}
&\frac{\sum_{\mathbf{y}} p_{\vec{\theta}}(\mathbf{y})p_{\vec{\theta}}(\mathbf{x}_c \mid \mathbf{y})p_{\vec{\theta}}(\mathbf{x}_s \mid \mathbf{x}_c, \mathbf{y})}{\sum_{\mathbf{y},\mathbf{x}'_s} p_{\vec{\theta}}(\mathbf{y})p_{\vec{\theta}}(\mathbf{x}_c \mid \mathbf{y})p_{\vec{\theta}}(\mathbf{x}'_s \mid \mathbf{x}_c, \mathbf{y})} \\
&= \frac{p_{\vec{\theta}}(\mathbf{x}_s \mid \mathbf{x}_c)p_{\vec{\theta}}(\mathbf{x}_c)}{\sum_{\mathbf{x}'_s} p(\mathbf{x}_s \mid \mathbf{x}_c)p_{\vec{\theta}}(\mathbf{x}_c)} && (4.16) \\
&= p_{\vec{\theta}}(\mathbf{x}_s \mid \mathbf{x}_c) && (4.17)
\end{aligned}$$

which is exactly as desired. We have demonstrated that, under CE, the best choice of the hidden variable $\mathbf{Y}$ is to match the distribution over the part of the observation that the neighborhood obliterates, given the part of the observation that is kept intact.

[9]Indeed, we do not have so much freedom to manipulate these two factors, since $\mathbf{X}$ is a sentence and $\mathbf{Y}$ is a tree. MLE learning will, nonetheless, try to use the model over $\mathbf{Y}$ to make the observed $\mathbf{X}$ probable.

## 4.4 CE as a Generalization of Other Methods

A number of earlier approaches, including MLE, can be seen as special cases of CE.

### 4.4.1 MLE as a Case of CE

MLE is, trivially, CE with the neighborhood function $\mathcal{N}(\mathbf{x}) = \Sigma^*$, for all $\mathbf{x}$. Thinking contrastively, what does this tell the learner? Hardly anything—it tells the learner to find a syntax model that makes $\mathbf{x}$ more likely than anything else in $\Sigma^*$ (balancing among all of the observed examples). There are many ways in which a particular $\mathbf{x}$ is different from all of the other $\mathbf{x}' \in \Sigma^*$.

For example, suppose $\mathbf{x}$ contains the word *coffee* and not *granola*, and we are trying to estimate a lexicalized version of Model A. One way for a learner to improve likelihood is to make *coffee* more likely, for example by increasing the weight of every feature function that notices the presence of the word *coffee*. Another way to improve likelihood is to *decrease* all *granola* feature functions. This has nothing to do with syntactic structure—it has to do with content. Why should the sentence *Coffee can be reheated*, when observed, result in decreased probability for a never-before seen (but grammatical) sentence like *Granola watched carefully does not become tasty* as well as the ungrammatical *Coffee jar be reheated*?

In this thesis, we do not want the model to do the work of predicting content. Indeed, the content of $\mathbf{x}$ is always known anyway; it is an input to the parser! MLE, we suggest, demands that the model do far more explanation than necessary. In this sense CE aims to be discriminative (Section 4.2).

### 4.4.2 Riezler's Approximation

Riezler (1999) and Riezler *et al.* (2000) employed a generalized EM algorithm to carry out unsupervised training with a log-linear model corresponding to a weighted lexical-functional grammar. In order to avoid the partition function $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$, Riezler made the assumption that $\Sigma^* \approx \tilde{\mathcal{X}} = \{\mathbf{x}_i^{\mathrm{t}}\}_i$. Therefore the partition function becomes $\ddot{Z}_{\vec{\theta}}(\tilde{\mathcal{X}})$, which with a finite training set is guaranteed to be finite.[10] In fact, the method achieved some good results for Riezler. This is an instance of contrastive estimation where the neighborhood of each example is the set of all observed examples in training: $\mathcal{N}(\mathbf{x}) = \tilde{\mathcal{X}}$. In the

[10]Note that estimating the model in this way does not necessarily result in a model that gives zero probability mass to $\mathbf{x}$ not seen in training, just as MLE doesn't.

contrastive interpretation, this seems difficult to justify and even circular: probability mass is to be moved onto each example in the training set, at the expense of the others.

### 4.4.3 CE and Conditional EM

*Conditional* EM is a method used to optimize MCLE when there are hidden variables involved that are not part of the input or output of the model. Suppose $\mathcal{W} = \Sigma^* \times \mathcal{Y} \times \mathcal{T}$, and examples $\vec{\mathbf{x}}^{\mathrm{t}}$ with annotations $\vec{\mathbf{y}}^{\mathrm{t}}$ are given, but $\mathbf{T}$ is always unobserved. Then MCLE requires summing over values of $\mathbf{T}$:

$$\max_{\vec{\theta}} \prod_{i=1}^{|\vec{\mathbf{x}}^{\mathrm{t}}|} p_{\vec{\theta}}(\mathbf{y}_i^{\mathrm{t}} \mid \mathbf{x}_i^{\mathrm{t}}) \equiv \max_{\vec{\theta}} \prod_{i=1}^{|\vec{\mathbf{x}}^{\mathrm{t}}|} \sum_{\mathbf{t} \in \mathcal{T}} p_{\vec{\theta}}(\mathbf{y}_i^{\mathrm{t}}, \mathbf{t} \mid \mathbf{x}_i^{\mathrm{t}}) \tag{4.18}$$

Conditional EM locally optimizes the above. If we think of this as an unsupervised learning problem, where $\mathbf{T}$ is the variable whose distribution we wish to infer and $\mathbf{X}$ and $\mathbf{Y}$ are the observed part of the data, then $\mathcal{N}(\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}, \mathbf{y}') : \mathbf{y}' \in \mathcal{Y}_{\mathbf{x}}\}$.

As it happens, CE with any neighborhood function that defines an equivalence relation

$$R(\mathbf{x}, \mathbf{x}') \Leftrightarrow \mathbf{x}' \in \mathcal{N}(\mathbf{x})$$

equates to this hidden-variable MCLE. An example is the permutation neighborhood described earlier (Section 4.3). In this case, the neighborhood of a sequence corresponds to all reorderings of the elements in the sequence. $\Sigma^*$ is therefore partitioned into all possible bags of words; there are infinitely many equivalence classes, each containing a finite set of sequences. To see how this is an example of MCLE with a hidden variable, we must redefine the variables. Let $\mathbf{x}$ correspond to a bag of words (histogram). Let $\mathbf{y}$ correspond to the sequence and let $\mathbf{t}$ correspond to the syntactic structure. Then MCLE (maximizing $p_{\vec{\theta}}(\vec{\mathbf{y}} \mid \vec{\mathbf{x}})$, summing over all $\mathbf{t}$) corresponds exactly to CE with the all-permutations neighborhood.

CE in general, however, is not just a shell game on the random variables. The neighborhood function need not define an equivalence relation, and in this thesis, it usually does not, and when CE performs best, it does not. In the next section, we will consider some novel neighborhood functions.

## 4.5 Computational Considerations

Contrastive estimation is a very natural fit for log-linear models. Let us consider what is required computationally for parameter estimation of a log-linear model as shown in Equation 4.6. To apply a gradient-based numerical optimization algorithm, we need the value of the function and the gradient. As with many parameter estimation methods, we will maximize a summation over log-ratios:

$$\max_{\vec{\theta}} \prod_{i=1}^{|\vec{x}^{\mathrm{t}}|} \frac{\displaystyle\sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})}{\displaystyle\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}$$

$$\equiv \quad \max_{\vec{\theta}} \sum_{i=1}^{|\vec{x}^{\mathrm{t}}|} \left( \underbrace{\log \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i^{\mathrm{t}}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})}_{\text{A}} - \underbrace{\log \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}}} \ddot{u}_{\vec{\theta}}(\mathbf{x}, \mathbf{y})}_{\text{B}} \right) \tag{4.19}$$

For a given example, we must compute two terms (labeled A and B above). The A terms correspond to $\log \ddot{Z}_{\vec{\theta}}(\mathbf{x}^{\mathrm{t}})$ terms that can be computed by an Inside algorithm. If the cardinality of $\mathcal{N}(\mathbf{x}_i^{\mathrm{t}})$ is finite, we can compute the B terms by taking a logarithm of the sum of each $\ddot{Z}_{\vec{\theta}}(\mathbf{x})$ term in the neighborhood and carrying out the Inside algorithm separately for each of the $\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})$. This will scale runtime (for us, already cubic) by the size of the neighborhood, unfortunately—but we will show a speedup applicable to many neighborhoods.

The first derivative of the contrastive objective in Equation 4.19 with respect to an arbitrary $\theta_j$ is:

$$\frac{\partial}{\partial \theta_j} = \sum_{i=1}^{|\vec{x}^{\mathrm{t}}|} \Big( \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{x}_i^{\mathrm{t}})} \left[ f_j(\mathbf{x}_i^{\mathrm{t}}, \mathbf{Y}) \right] - \mathbf{E}_{p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y} \mid \mathbf{X} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}}))} \left[ f_j(\mathbf{X}, \mathbf{Y}) \right] \Big) \tag{4.20}$$

The gradients of the A and B terms (the two expectations above) can be computed using the Inside and Outside algorithms. The reader may note the similarity between the above gradient and Equation 3.20, which gave the gradient for partial-data MLE with log-linear models.[11]

[11] It is left as an exercise in algebra to show that the two are equivalent up to a constant factor when $\mathcal{N}(\mathbf{x}) = \Sigma^*$.

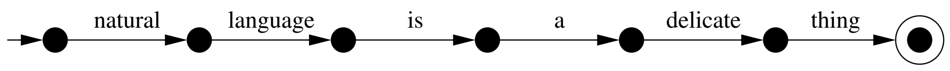


Figure 4.1: Lattice representation of the sentence *Natural language is a delicate thing*.

When our data are sequences (in $\Sigma^*$), there is a computational trick that can make term B and its gradient far more efficient to compute. Recognizing that many of the $\mathbf{x}'$ in $\mathcal{N}(\mathbf{x})$ may share structure with each other, we can represent the neighborhood as an acyclic finite-state machine, or **lattice**. Dynamic programming algorithms for grammatical processing of sequences can be very straightforwardly adapted to lattices (Eisner *et al.*, 2004). Note that lattices are more general than sequences; a sequence can be represented as a straight-line lattice with a single path (Figure 4.1). The lattice semiring dynamic programming algorithm for Model A is given in Section A.2.

Formally, running an Inside algorithm on a lattice corresponds to computing the sum of probabilities of all structures over all paths in the lattice. If there are multiple paths through the lattice that correspond to the same sequence, that sequence will be counted multiple times—so it is important to determinize the lattice to avoid this double counting. Unfortunately determinization can explode the size of the lattice, so the tradeoff between exactness and efficiency is up to the user.

Further, as an upper bound, if the sequence dynamic programming algorithm for a weighted grammar requires $O(|\mathbf{x}|^P)$ for some polynomial degree $P$, then the *lattice* variant run on a lattice with $a$ arcs will require $O(a^P)$. This does not always affect the highest-degree terms (an example is CKY); the bound may be very loose. For computational efficiency, it may be wise to apply finite-state machine minimization to the lattice, as well. (Minimization requires determinization, so this may not be a practical option.) For the lattices discussed here, determinization and minimization were practical and fast using the FSM toolkit (Mohri, Pereira, and Riley, 1998).

### 4.5.1 Lattice Neighborhoods

We motivated CE with an example neighborhood including all permutations of a sentence $\mathbf{x}$. Unfortunately, there is no efficient way to represent and sum over all reorderings of $\mathbf{x}$.[12] We now describe and illustrate some simple neighborhoods designed also for the purpose of uncovering natural language sentence structure. Unlike the permutation neighborhood, these neighborhoods are computationally efficient because they can be represented as lattices using $O(|\mathbf{x}|)$ space. They are meant to capture simple insights about natural language syntax, but they are admittedly rather crude. It is left to future work to more carefully craft neighborhood functions for this problem (and others).

**Single Transposition**

We have noted how an intuitive view of syntax is that it should explain word order. While a neighborhood consisting of *all* permutations of an observed sentence is unwieldly and cannot be represented in a small lattice, we can consider a smaller set of permutations.

The single transposition neighborhood, TRANS1, consists of $\mathbf{x}$ plus the $|\mathbf{x}| - 1$ sequences generated by transposing any two adjacent symbols in $\mathbf{x}$:

$$\mathcal{N}_{\text{TRANS1}}(\mathbf{x}) = \left\{x_1...x_{\ell-1}x_{\ell+1}x_\ell x_{\ell+2}...x_{|\mathbf{x}|} : 1 \leq \ell < |\mathbf{x}|\right\} \cup \{\mathbf{x}\} \quad (4.21)$$

This neighborhood guides the model to explain why words are locally ordered as observed. Consider, for example, the sentence *Natural language is a delicate thing*. The TRANS1 neighborhood includes:

*Language natural is a delicate thing.* (poetic)
*Natural is language a delicate thing.* (poor)
*Natural language a is delicate thing.* (poor)
*Natural language is delicate a thing.* (poor)
*Natural language is a thing delicate.* (poetic)

These are not examples of fluent, dissertation-genre English. Applying contrastive estimation with this neighborhood will lead the learner to discriminate the correct, observed sentence from its transposed perturbations, perhaps making use of good syntactic descriptions to do so.

Figure 4.2 shows the lattice representation of this neighborhood, for our example sentence. Notice that it has $O(|\mathbf{x}|)$ arcs (roughly $4|\mathbf{x}|$), so a cubic-runtime dynamic programming algorithm on the original sequence $\mathbf{x}$ stays cubic-runtime on the lattice. Figure 4.3 shows the finite-state transducer that can be used to generate the lattice by composition with $\mathbf{x}$'s lattice (Figure 4.1).

[12]Though not tractable for large $|\mathbf{x}|$, the all-permutations neighborhood *can* be represented using a lattice of $O(2^{|\mathbf{x}|})$ states and arcs.

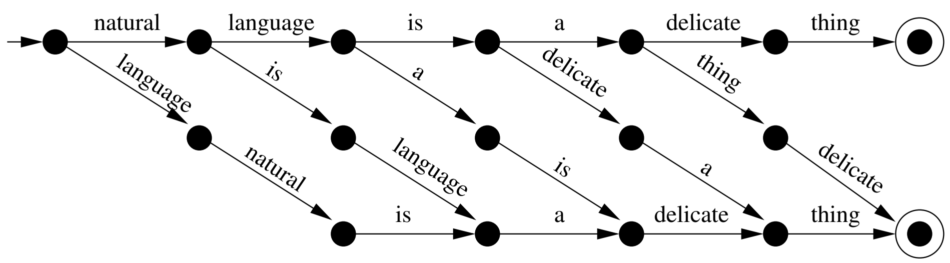


Figure 4.2: Lattice representation of the TRANS1 neighborhood for *Natural language is a delicate thing*. This lattice (acyclic finite-state machine) can be produced by composing the original sentence (Figure 4.1) with a simple transducer (Figure 4.3).

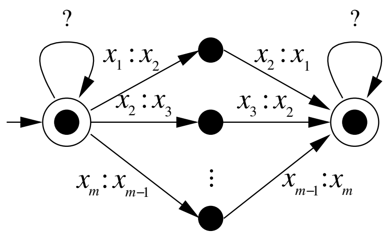


Figure 4.3: A transducer for constructing the TRANS1 neighborhood. Each bigram $\langle x_i, x_{i+1} \rangle \in \Sigma^2$ has an arc pair $\langle x_i : x_{i+1}, x_{i+1} : x_i \rangle$.

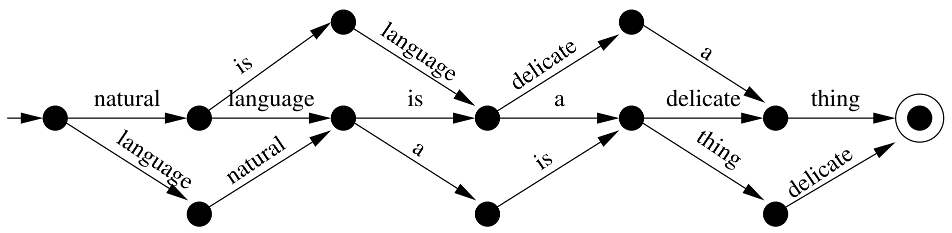


Figure 4.4: Lattice representation of the DYNASEARCH neighborhood for *Natural language is a delicate thing*. This lattice (acyclic finite-state machine) can be produced by composing the original sentence (Figure 4.1) with a simple transducer (Figure 4.5).

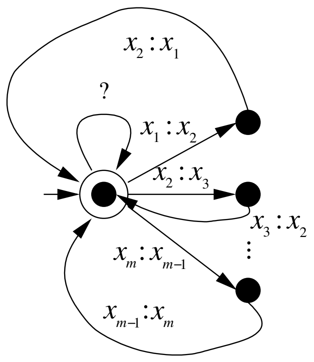


Figure 4.5: A transducer for constructing the DYNASEARCH neighborhood. Each bigram $\langle x_i, x_{i+1} \rangle \in \Sigma^2$ has an arc pair $\langle x_i : x_{i+1}, x_{i+1} : x_i \rangle$.

**Dynasearch Neighborhood**

The DYNASEARCH neighborhood is also based on transpositions. This neighborhood was originally proposed in the context of generalized search by Potts and van de Velde (1995) and later used by Congram, Potts, and van de Velde (2002). It allows any pairs of adjacent words to be transposed, but the transpositions cannot overlap (each word may only be moved once). This is best understood by looking at an example lattice like Figure 4.4. This neighborhood lattice can be obtained by composing the sentence (Figure 4.1) with the generating transducer in Figure 4.5. Interestingly, the number of sentences in $\mathcal{N}_{\text{DYNASEARCH}}(\mathbf{x})$ is equal to the $|\mathbf{x}|$th Fibonacci number $\langle 1, 1, 2, 3, 5, ... \rangle$, which grows as exponentially, but the lattice representation requires only $O(|\mathbf{x}|)$ arcs.

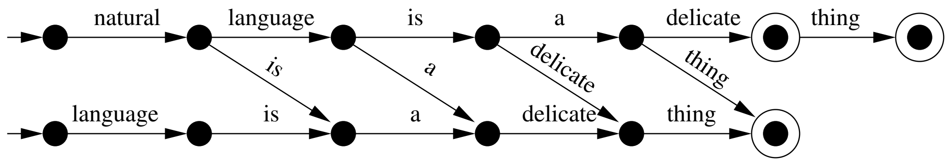


Figure 4.6: Lattice representation of the DEL1 neighborhood for *Natural language is a delicate thing*. This lattice (acyclic finite-state machine) can be produced by composing the original sentence (Figure 4.1) with a simple transducer (Figure 4.7).

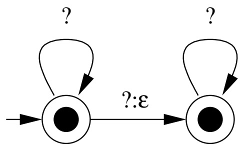


Figure 4.7: A transducer for constructing the DEL1 neighborhood. Each unigram $x$ has an arc $x : \epsilon$.

### Single Deletion

Another view of syntax is that it should explain the presence of each observed word. The single deletion neighborhood, DEL1, includes all sequences in which a single word has been deleted from $\mathbf{x}$:

$$\mathcal{N}_{\text{DEL1}}(\mathbf{x}) = \left\{x_1...x_{\ell-1}x_{\ell+1}...x_{|\mathbf{x}|} : 1 \leq \ell \leq |\mathbf{x}|\right\} \cup \{\mathbf{x}\} \tag{4.22}$$

The number of sequences in this neighborhood is $|\mathbf{x}| + 1$. An example of a DEL1 neighborhood lattice is shown in Figure 4.6; it has roughly $3|\mathbf{x}|$ arcs. The generating finite-state transducer is shown in Figure 4.7. Notice that some single-word deletions do not damage the well-formedness of the sentence (the original is *Natural language is a delicate thing*):

*Language is a delicate thing.*
*Natural is a delicate thing.* (poor)
*Natural language a delicate thing.* (poor)
*Natural language is delicate thing.* (poor)
*Natural language is a thing.*
*Natural language is a delicate.* (poor)

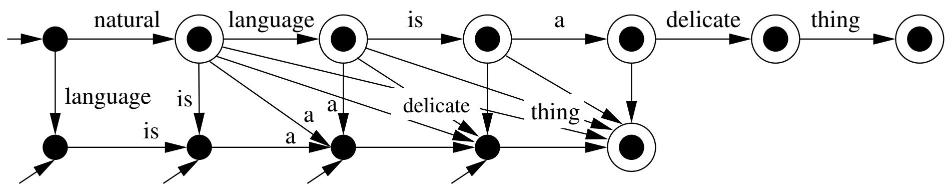


Figure 4.8: Lattice representation of the DEL1SUBSEQ neighborhood for *Natural language is a delicate thing*. This lattice (acyclic finite-state machine) can be produced by composing the original sentence (Figure 4.1) with a simple transducer (Figure 4.9).

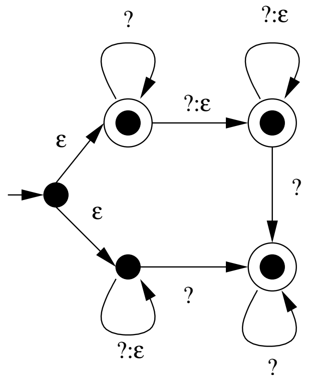


Figure 4.9: A transducer for constructing the DEL1SUBSEQ neighborhood. Each unigram $x$ has an arc $x : \epsilon$.

### Subsequence Deletion

A similar neighborhood to DEL1 is DEL1SUBSEQ, which allows the deletion of a single consecutive subsequence (but not the entire string):

$$\mathcal{N}_{\text{DEL1SUBSEQ}}(\mathbf{x}) = \left\{x_1...x_\ell x_k...x_{|\mathbf{x}|} : 0 \leq \ell < k \leq |\mathbf{x}| + 1\right\} \setminus \{\epsilon\} \tag{4.23}$$

This neighborhood contains $O(|\mathbf{x}|^2)$ sequences and requires $O(|\mathbf{x}|^2)$ arcs. Illustrations of the neighborhood lattice and its generating transducer are in Figures 4.8 and 4.9, respectively.

### Same Length

We refer to the neighborhood of all $|\mathbf{x}|$-length sequences of $\Sigma$, $\Sigma^{|\mathbf{x}|}$ as the LENGTH neighborhood. This can be represented by a lattice similar to Fig. 4.1, but where each arc accepts any $x \in \Sigma$. The number of arcs, then, is $O(|\Sigma||\mathbf{x}|)$, which is potentially quite large.

The good news is that all sentences of the same length have the same neighborhood. In a given training set $\vec{\mathbf{x}}^{\mathrm{t}}$, many sentences will be of the same length; the computation of the "B" term (Equation 4.19, here corresponding to $\log \ddot{Z}_{\vec{\theta}}(\Sigma^k)$) can be done once per training iteration and reused for all $k$-length sentences, for each $k$.

Further, because the lattice for $\Sigma^{k-1}$ is a prefix of the lattice for $\Sigma^k$, computation can be shared as follows. Begin by solving the dynamic program for $k = 1$ (the lattice contains $\Sigma$). Then increase the lattice by adding another state (and arcs into it), so that the lattice contains $\Sigma^2$. The stored terms for the first round ($k = 1$) are needed for the second, and do not need to be recomputed. This continues, with incremental addition of states and arcs, until $k = \max_i |\mathbf{x}_i^{\mathrm{t}}|$.

While it might be prohibitive for computation with some models, LENGTH was reasonably fast with Model A, partly because our datasets contained only sentences such that $|\mathbf{x}^{\mathrm{t}}| \leq 10$. LENGTH was originally proposed as an approximation to log-linear MLE (Smith and Eisner, 2005a), before Smith and Johnson (in review) demonstrated that (for CFG-based models) stochastic MLE is equivalent to log-linear MLE. Because LENGTH partitions $\Sigma^*$, it is an instance of MCLE (see Section 4.4.3: redefine $\mathbf{x}$ to be the length of the sequence, $\mathbf{y}$ to be the sequence, and $\mathbf{t}$ to be the dependency tree).

### Approximating $\Sigma^*$ with $\Sigma^{\leq k}$

Haghighi and Klein (2006) recently proposed a CE neighborhood that is a closer approximation to log-linear MLE. Let $k$ be an integer larger than the longest training sequence. Then let the set of all sequences up to $k$ words long be denoted by

$$\Sigma^{\leq k} \overset{\mathrm{def}}{=} \bigcup_{i=0}^{k} \Sigma^k \tag{4.24}$$

We do not make use of this neighborhood here, since (as noted) for our model, log-linear and stochastic MLE are equivalent.

### Union and Composition of Neighborhood Functions

An attractive property of the finite-state lattice approach to neighborhoods is that well-known finite-state operations can be applied (Hopcroft and Ullman, 1979). For instance, unions of neighborhoods can be taken; the union of DEL1 and TRANS1 (which we name DEL1ORTRANS1), we will find, is a very effective neighborhood that still has $O(|\mathbf{x}|)$ arcs in a lattice.

Going farther, we can apply the neighborhood transducers in sequence. Composing $\mathbf{x}$ with the TRANS1 transducer twice gives TRANS2 (up to two transpositions, which *may* overlap, cf. DYNASEARCH). Composing $\mathbf{x}$ with DEL1 then TRANS1 allows up to one deletion followed by up to one transposition. Obviously there are many options; we will also experiment with DEL1ORTRANS2, which is the union of DEL1 and TRANS2.

Readers are also referred to Eisner and Tromble (2006), who describe ways to efficiently search over an exponentially large subset of sequence permutations. Their tree-based neighborhoods are a natural extension of our lattice-based neighborhoods and could be efficiently used in the contrastive training of finite-state models (e.g., as in Smith and Eisner, 2005a).

## 4.6 Experiments

We now describe experiments comparing the accuracy of models learned by CE training with those learned by MLE training. Recall that CE is a natural fit for log-linear models, and that the log-linear variant of Model A is exactly as expressive as a class of probability distributions $p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})$ as the stochastic variant (Section 3.5.4. For these reasons, our contrastive estimates are always on log-linear Model A. The experimental setup is the same as in Section 3.3.1, with English part-of-speech sequences used for training. In Section 8.3 we will see similar experiments on data in five additional languages. We first consider unregularized CE with different neighborhoods using the three initializers.

To optimize the CE objective, we use a limited memory variable metric (LMVM) method (also known as L-BFGS) due to Benson and Moré (2001). Unlike EM, this method does not automatically compute the step-size on each iteration. Instead, a line search is used, so more than one evaluation of the objective may be required to take a single step. We report the number of evaluations required. Training is run until either 100 steps have been taken or relative convergence within $10^{-5}$ is reached (whichever comes first).

To optimize the MLE objective, we used the EM algorithm (see Section 3.3.1, where the relevant conditions were initially presented). This is because local optimization for the constrained, stochastic grammar equates to local optimization of the unconstrained, log-linear grammar (see Section 3.5.1). The search methods are different, then, which might

|  | Zero |  |  | K&M |  |  | Local |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  |  | test accuracy |  |  | test accuracy |  |  | test accuracy |  |
| $\mathcal{N}$ | evaluations | directed | undirected | evaluations | directed | undirected | evaluations | directed | undirected |
| $\Sigma^*$ (MLE/EM) | 49 | 22.7 | 58.8 | 62 | 41.7 | 62.2 | 49 | 22.8 | 58.9 |
| DEL1 | 17 | 18.3 | 33.3 | 16 | 39.7 | 53.5 | 16 | 38.3 | 56.3 |
| TRANS1 | 107 | 39.6 | 61.0 | 104 | 39.8 | 61.7 | 107 | 28.9 | 61.7 |
| DEL1ORTRANS1 | 103 | 35.8 | 62.2 | 103 | **48.6** | **64.9** | 104 | **57.6** | **69.0** |
| LENGTH | 104 | 34.9 | 61.9 | 104 | **45.5** | **64.9** | 104 | 37.6 | 61.2 |
| DYNASEARCH | 105 | 42.4 | **63.6** | 54 | **44.7** | **64.8** | 106 | 38.6 | 60.6 |
| TRANS2 | 108 | 33.5 | **64.1** | 102 | 40.2 | 61.8 | 105 | 24.8 | 62.7 |
| DEL1ORTRANS2 | 103 | 25.6 | 61.6 | 106 | 38.9 | 59.6 | 104 | 24.9 | 61.9 |
| DEL1SUBSEQ | 16 | 18.9 | 35.9 | 17 | 25.6 | 43.0 | 15 | 21.0 | 40.4 |

Table 4.1: Model A trained on English data using unregularized CE, with three different initializers and various neighborhoods $\mathcal{N}$. Boldface marks trials significantly better than $\Sigma^*$/K&M (sign test, $p < 0.05$).

underlie some of the differences, though both methods (EM and LMVM) are local hill-climbers.

Results are presented in Table 4.1 for a variety of neighborhoods. Just like likelihood, these objective functions are bumpy and sensitive to initialization (compare performance across any row). However, certain neighborhoods lead to considerably more accurate models. Looking at directed accuracy, we see that DEL1ORTRANS1, LENGTH, and DYNASEARCH are able to surpass the performance of MLE if the right initializer is chosen. Further, for any of the three initializers, each of these three objective functions outperforms MLE with the same initializer.

### 4.6.1 Regularization

We experimented with different regularization settings for contrastive estimation. In addition to the unregularized setting already mentioned, each objective was trained with a 0-mean diagonal Gaussian prior on parameters, with $\sigma^2 \in \{10^{-2/3}, 10^{-1/3}, 1, 10^{1/3}, 10^{2/3}\}$ (see Section 3.5.5). Supervised model selection (the model with the best directed accuracy on 531 sentences of annotated development data) and unsupervised model selection (the

|  | supervised selection |  |  |  |  | unsupervised selection |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  | test accuracy |  |  |  |  | test accuracy |  |
| $\mathcal{N}$ |  |  | evaluations | directed | undirected |  |  | evaluations | directed | undirected |
| $\Sigma^*$ (MAP/EM) | K&M | $\lambda = 10^{-2/3}$ | 49 | 41.6 | 62.2 | K&M | $\lambda = 10^{-2/3}$ | 49 | 41.6 | 62.2 |
|  |  | $\sigma^2 =$ |  |  |  |  | $\sigma^2 =$ |  |  |  |
| Del1 | Local | $\infty$ | 16 | 39.7 | 53.5 | Zero | $\infty$ | 17 | 18.3 | 33.3 |
| Trans1 | Zero | 1 | 75 | 41.2 | 62.5 | K&M | $10^{-1/3}$ | 103 | 29.4 | 57.8 |
| Del1OrTrans1 | Local | $\infty$ | 104 | **57.6** | **69.0** | K&M | 1 | 105 | **47.3** | **63.6** |
| Length | K&M | $\infty$ | 104 | **45.5** | **64.9** | K&M | $\infty$ | 104 | **45.5** | **64.9** |
| DynaSearch | Zero | $10^{1/3}$ | 105 | **47.6** | **65.3** | K&M | $10^{1/3}$ | 107 | 39.7 | 61.9 |
| Trans2 | K&M | $\infty$ | 102 | 40.2 | 61.8 | K&M | $\infty$ | 102 | 40.2 | 61.8 |
| Del1OrTrans2 | K&M | $10^{-1/3}$ | 110 | **51.2** | **65.0** | K&M | 1 | 104 | 37.9 | 59.3 |
| Del1Subseq | K&M | $10^{1/3}$ | 38 | 26.6 | 41.0 | Zero | $\infty$ | 16 | 18.9 | 35.9 |
| overall: Del1OrTrans1 | Local | $\infty$ | 104 | 57.6 | 69.0 |  |  |  |  |  |

Table 4.2: Model A trained on English data using CE with regularization. For each neighborhood $\mathcal{N}$, supervised and unsupervised model selection (over initializers and regularizers) were applied. Note that $\Sigma^*$ gets the same advantage of regularization, though with a Dirichlet prior instead of a Gaussian. Boldface marks trials significantly better than the $\Sigma^*$ (MAP/EM) trial under a sign test ($p < 0.05$). For a graphical comparison against other methods over six languages, see Figure 8.2 (page 213) and Figure 8.3 (page 214).

model achieving the highest training objective function value on 531 sentences of unannotated development data) were applied across values of $\sigma^2$ and $\vec{\theta}^{(0)}$; then across $\mathcal{N}$ as well (in the supervised case). Table 4.2 shows the results. Note that supervised model selection tends to perform much better than unsupervised model selection, but CE can outperform MAP under either setting. Note that with regularization and supervised selection, four of our lattice neighborhoods outperform EM: DEL1ORTRANS1, LENGTH, DYNASEARCH, and DEL1ORTRANS2.

Supervised model selection was also carried out across all of the models trained, and the model selected was the DEL1ORTRANS1 model without regularization. The improvement over MAP training by EM is 16 absolute points of directed accuracy. For this dataset, problem, and initializer, unregularized DEL1ORTRANS1 seems to be a very good choice. (We will explore other datasets in Chapter 8, and see Section 4.6.5 for more initializers.) Regularization did not degrade DEL1ORTRANS1's performance too strongly; as long as $\sigma^2 \geq 1$, accuracy dropped by no more than 3 points.

### 4.6.2 Error Analysis

We consider the supervisedly selected CE model ($\mathcal{N}$ = DEL1ORTRANS1, Local initializer, no regularization) and compare its errors to those of MAP/EM. The CE model achieved 57.6% directed accuracy (69.0% undirected) against EM's 41.6% (62.2%). On whole-sentence accuracy, CE achieved 13.6% against EM's 7.7%. The average number of (directed) attachments corrected per sentence by CE over EM was 0.95.

Table 4.3 shows errors by undirected link type. It is difficult to see a pattern of errors here. Whereas in the analogous EM table (Table 3.3 on page 61) it was easy to spot high-recall/low-precision (over-hypothesized) and high-precision/low-recall (under-hypothesized) link types, we see only only a few stark cases in this table. Most notably, preposition-verb (IN/VBZ, IN/VBD) links are guessed too often by the CE model, and adverb-verb (RB/VB) links too seldom. The latter problem was shared by EM.

The types of link errors are not as sharply distributed for CE as for EM. This is a good sign, suggesting that the model has learned approximately the right distribution of link types in data. Indeed, when we compare the distributions over link types in the hypothesized trees to the gold-standard trees, CE is closer, with 0.05 bits (measured in mean KL divergence to the mean) as compared to EM's 0.13 bits for undirected links, and

0.15 bits as compared to EM's 0.28 bits for directed links.

Recall that MAP/EM made more errors on tag pairs of greater distance (Table 3.4, page 62). Considering CE's errors by string distance (Table 4.4), there is considerably less of a distance effect—as distance increases, accuracy does not fall off as sharply as it did for EM.

Table 4.5 gives the path analysis of the test dataset trees hypothesized by MAP/EM and CE/DEL1ORTRANS1. The latter results in paths between children and their parents that are on average shorter, and in the directed case more frequently finite.

In summary, the errors made by the CE model are fewer in number than EM's errors, and are more diverse and more evenly distributed. This is a good sign; it means that improved models (with better features and perhaps more expressive power), or larger datasets, should lead to models that can better exploit more diverse types of clues necessary for better syntactic disambiguation.

### 4.6.3 Comparison by Objective Functions

Having trained and selected models in various ways, we compared the different resulting models under the different objective functions. That is, each learned model was used to compute the likelihood, and each of the contrastive conditional likelihoods of the development dataset. The results are shown in Table 4.6. Larger neighborhoods correspond to larger magnitudes in the log-likelihoods, because the denominator sums over more sentences.

In general, training on neighborhood $\mathcal{N}$ was the best way to optimize the contrastive likelihood on neighborhood $\mathcal{N}$. Some exceptions do stand out, though. EM training was better at optimizing contrast with the LENGTH neighborhood than LENGTH contrastive training. TRANS2 and DYNASEARCH were best at optimizing TRANS1. DEL1 and DEL1SUBSEQ were roughly as good as each other on both contrastive functions, though they learned different models (their rows in the table are not identical).

Importantly, EM training was always the best way to optimize likelihood. This means that the other neighborhoods tested are not improving performance by beating EM at its own game—rather, they are optimizing other functions that, in some cases, are better predictors of accuracy than likelihood is.

| tag types | | count in hyp. | precision | recall | count in gold |
|---|---|---|---|---|---|
| DT | NN | 156 | 83.3 | 57.0 | 228 |
| NNP | NNP | 202 | 59.9 | 67.6 | 179 |
| IN | NN | 132 | 81.8 | 84.4 | 128 |
| NN | VBD | 91 | 87.9 | 66.7 | 120 |
| JJ | NN | 112 | 78.6 | 74.6 | 118 |
| NN | VBZ | 95 | 85.3 | 74.3 | 109 |
| JJ | NNS | 63 | 96.8 | 75.3 | 81 |
| IN | NNS | 72 | 80.6 | 78.4 | 74 |
| NNS | VBD | 68 | 83.8 | 83.8 | 68 |
| NN | NN | 61 | 83.6 | 77.3 | 66 |
| NN | NNS | 49 | 83.7 | 67.2 | 61 |
| IN | VBD | 87 | 62.1 | 91.5 | 59 |
| RB | VBD | 32 | 75.0 | 42.9 | 56 |
| NN | NNP | 45 | 68.9 | 55.4 | 56 |
| NNS | VBP | 52 | 80.8 | 82.4 | 51 |
| NNP | VBD | 46 | 76.1 | 72.9 | 48 |
| CD | CD | 50 | 88.0 | 95.7 | 46 |
| RB | VBZ | 27 | 66.7 | 40.9 | 44 |
| NNP | VBZ | 36 | 94.4 | 77.3 | 44 |
| CD | NN | 50 | 74.0 | 86.0 | 43 |
| PRP | VBD | 39 | 89.7 | 85.4 | 41 |
| CD | IN | 37 | 94.6 | 87.5 | 40 |
| TO | VBD | 33 | 87.9 | 74.4 | 39 |
| PRP | VBZ | 37 | 94.6 | 89.7 | 39 |
| DT | NNS | 27 | 88.9 | 61.5 | 39 |
| MD | VB | 40 | 95.0 | 100.0 | 38 |
| TO | VB | 28 | 100.0 | 82.4 | 34 |
| CD | TO | 37 | 45.9 | 51.5 | 33 |
| CD | NNS | 32 | 87.5 | 84.8 | 33 |
| IN | NNP | 37 | 75.7 | 87.5 | 32 |
| IN | VBZ | 57 | 24.6 | 73.7 | 19 |
| DT | NNP | 36 | 22.2 | 53.3 | 15 |
| CD | VBD | 37 | 40.5 | 100.0 | 15 |
| DT | JJ | 61 | 3.3 | 100.0 | 2 |

Table 4.3: This table shows the undirected precision and recall of the Local-initialized, unregularized trial of CE with DEL1ORTRANS1 neighborhood, by the unordered tag-pair type. Note that attachments in either direction and with the tags in either order are counted together. Tag-pair types with a count $\geq 30$ in the hypothesized annotation or the gold-standard are listed. Compare with Table 3.3 on page 61.

| length: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| hyp. count: | 2242 | 429 | 233 | 108 | 81 | 39 | 20 | 5 | 1 |
| | 1759 483 | 118 311 | 56 177 | 33 75 | 27 54 | 13 26 | 6 14 | 1 4 | 0 1 |
| precision: | 73.6 | 70.2 | 55.4 | 43.5 | 33.3 | 41.0 | 35.0 | 20.0 | 0.0 |
| | 62.1 67.7 | 27.1 67.2 | 8.9 56.5 | 12.1 32.0 | 33.3 13.0 | 15.4 19.2 | 16.7 7.1 | 0.0 0.0 | – 0.0 |
| recall: | 84.2 | 43.2 | 45.4 | 40.5 | 49.1 | 53.3 | 58.3 | 25.0 | 0.0 |
| | 87.7 45.8 | 10.2 54.6 | 4.7 56.2 | 8.0 36.4 | 26.5 33.3 | 14.3 31.3 | 9.1 100.0 | 0.0 – | 0.0 – |
| gold count: | 1960 | 696 | 284 | 116 | 55 | 30 | 12 | 4 | 1 |
| | 1246 714 | 313 383 | 106 178 | 50 66 | 34 21 | 14 16 | 11 1 | 3 1 | 1 0 |

Table 4.4: Precision and recall by string distance between parent and child, for the CE/DEL1ORTRANS1 trial with $\sigma^2 = \infty$, initialized with Local. The large-typeface numbers show the undirected attachment precision and recall of dependencies at different lengths. Small-typeface numbers give the directed attachment precision and recall for left children and right children, by length. Compare with Table 3.4 on page 62.

| | ATTACH-RIGHT % of tag tokens | | MAP/EM % of tag tokens | | CE/DEL1ORTRANS1 % of tag tokens | |
|---|---|---|---|---|---|---|
| path length | undirected | directed | undirected | directed | undirected | directed |
| 1 | 55.3 | 36.0 | 64.9 | 47.3 | 66.9 | 57.1 |
| 2 | 20.0 | 9.6 | 25.2 | 21.1 | 21.3 | 15.3 |
| 3 | 9.8 | 5.0 | 7.1 | 5.8 | 7.7 | 5.3 |
| 4 | 5.3 | 3.6 | 2.0 | 1.3 | 2.7 | 1.8 |
| 5 | 3.1 | 2.5 | 0.7 | 0.5 | 1.2 | 0.8 |
| 6 | 2.8 | 2.4 | 0.0 | 0.0 | 0.2 | 0.1 |
| 7 | 1.8 | 1.8 | 0.0 | 0.0 | 0.1 | 0.1 |
| 8 | 1.1 | 1.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 9 | 0.6 | 0.6 | 0.0 | 0.0 | 0.0 | 0.0 |
| $\infty$ | – | 37.4 | – | 23.9 | – | 19.5 |

Table 4.5: Path analysis of the selected MAP/EM model and the selected CE/-DEL1ORTRANS1 model. The macro-averaged undirected path length in hypothesized trees between a word and its gold-standard parent is 1.31 for MAP/EM and 1.27 for DEL1ORTRANS1; under a sign test the CE/DEL1ORTRANS1 path averages were significantly shorter ($p < 0.000001$). Note that attachment accuracy is *not* identical to the first line because here we count paths to the wall (just another node in the tree), whereas our usual accuracy measures do not count attachments to the wall.

| selection | training $\mathcal{N}$ | log-likelihood | DEL1 | TRANS1 | DEL1ORTRANS1 | LENGTH | DYNASEARCH | TRANS2 | DEL1ORTRANS2 | DEL1SUBSEQ | directed accuracy | undirected accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| supervised | $\Sigma^*$ (EM) | **-23.76** | -9.20 | -2.00 | -8.75 | **-20.95** | -1.55 | -3.35 | -8.43 | -21.15 | 41.6 | 62.2 |
| | DEL1 | -28.83 | **-1.38** | -4.05 | -6.34 | -87.73 | -4.56 | -7.16 | -9.07 | **-2.52** | 39.7 | 53.3 |
| | TRANS1 | -24.75 | -2.61 | -1.93 | -4.10 | -29.10 | -1.51 | -3.33 | -5.17 | -6.18 | 41.2 | 62.5 |
| | DEL1ORTRANS1 | -25.38 | -1.39 | -2.24 | **-3.83** | -41.73 | -1.76 | -3.95 | -5.49 | -2.56 | 57.6 | 69.0 |
| | LENGTH | -24.06 | -6.52 | -2.05 | -6.34 | -20.98 | -1.62 | -3.42 | -6.44 | -11.79 | 45.5 | 64.9 |
| | DYNASEARCH | -24.53 | -2.36 | -1.91 | -3.90 | -30.90 | **-1.41** | -3.32 | -5.04 | -5.70 | 47.6 | 65.3 |
| | TRANS2 | -24.48 | -5.82 | **-1.88** | -5.66 | -39.43 | -1.42 | **-3.18** | -5.85 | -10.64 | 40.2 | 61.8 |
| | DEL1ORTRANS2 | -24.70 | -1.78 | -1.97 | -3.88 | -23.79 | -1.55 | -3.31 | **-4.92** | -4.62 | 51.2 | 65.0 |
| | DEL1SUBSEQ | -31.35 | -1.39 | -5.73 | -7.93 | -46.72 | -6.94 | -10.67 | -12.32 | -2.54 | 26.6 | 41.0 |
| unsupervised | $\Sigma^*$ (EM) | **-23.76** | -9.20 | -2.00 | -8.75 | **-20.95** | -1.55 | -3.35 | -8.43 | -21.15 | 41.6 | 62.2 |
| | DEL1 | -31.50 | **-1.38** | -4.29 | -6.69 | -40.48 | -5.44 | -7.74 | -9.46 | **-2.52** | 18.3 | 33.3 |
| | TRANS1 | -25.04 | -2.70 | -1.96 | -4.21 | -28.36 | -1.57 | -3.38 | -5.30 | -6.35 | 29.4 | 57.8 |
| | DEL1ORTRANS1 | -25.18 | -1.52 | -2.01 | **-3.66** | -25.04 | -1.55 | -3.58 | -5.18 | -3.65 | 47.3 | 63.6 |
| | LENGTH | -24.06 | -6.52 | -2.05 | -6.34 | -20.98 | -1.62 | -3.42 | -6.44 | -11.79 | 45.5 | 64.9 |
| | DYNASEARCH | -24.55 | -2.28 | **-1.88** | -3.85 | -32.29 | **-1.38** | -3.30 | -5.00 | -5.61 | 39.7 | 61.9 |
| | TRANS2 | -24.48 | -5.82 | **-1.88** | -5.66 | -39.43 | -1.42 | **-3.18** | -5.85 | -10.64 | 40.2 | 61.8 |
| | DEL1ORTRANS2 | -24.48 | -1.66 | -1.95 | -3.80 | -23.41 | -1.51 | -3.28 | **-4.85** | -4.22 | 37.9 | 59.3 |
| | DEL1SUBSEQ | -31.77 | **-1.38** | -6.46 | -8.60 | -60.04 | -7.92 | -12.13 | -13.73 | **-2.52** | 18.9 | 35.9 |

Table 4.6: Differently-trained models evaluated on different objective criteria. The objectives are measured on development data. "Supervised" refers to models selected across initialization/regularization conditions using annotated development data, "unsupervised" refers to models selected using unannotated development data (the best on the training criterion, applied to the development data). The likelihood criterion is computed after running one EM iteration; it is therefore approximate and an upper bound. (This is necessary because for log-linear models, we cannot compute $\ddot{Z}_{\vec{\theta}}(\mathcal{W})$.) Boldface corresponds to the best model in each column/model selection mode.

### 4.6.4 CE as a MLE Initializer

Each of the supervisedly and unsupervisedly selected models was used as an initializer to MLE training on the training set with EM. The results are shown in Table 4.7. EM training sometimes helped performance, most commonly in cases where CE did not perform well. In cases where CE was relatively successful, EM training had a small negative effect on accuracy (e.g., DEL1ORTRANS1). Note that log-likelihood did not consistently improve on the development dataset except where it was relatively poor after CE training. This supports the case that contrastive neighborhoods are not mere approximations to likelihood: improvements on likelihood from good CE models do not improve accuracy.

### 4.6.5 Further Analysis (Random Initializers)

Figure 4.10 shows the spread of accuracy performance of 100 random models (the same ones used in Chapter 3) before training, after EM training initialized from each, and after CE/DEL1ORTRANS1 training initialized from each. CE training, like EM training, gives highly variable results that depend heavily on the initializing model. The best of the CE models and the best of the EM models—from these random initializers—perform about the same on directed and undirected accuracy.

Figure 4.11 further illustrates that CE (with the DEL1ORTRANS1 neighborhood, at least) suffers from local maxima, just like MLE. 100 random initializers all find different local maxima with about the same function value and very different accuracies on test data.

Figure 4.12 shows that CE training with DEL1ORTRANS1—in contrast to MLE training (Figure 3.8)—does not always improve accuracy on random initializers. However, with any of our more carefully chosen initializers, it does improve accuracy (unlike EM). So CE is at least as sensitive to initialization as EM, but it can leverage a good initializer better than EM can. This is clearly illustrated in Figure 4.13, which plots the absolute improvement in accuracy (which is sometimes negative) by CE/DEL1ORTRANS1 training versus MLE/EM training, for our initializers and for random models. For more comparison among methods on random initializers, see Sections 5.3.2, 6.1.4, 6.2.2, and 7.4.

|  | supervised selection |  |  |  |  |  | unsupervised selection |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  |  | after EM |  |  |  |  |  | after EM |  |  |
|  |  | test accuracy |  |  | test accuracy |  |  | test accuracy |  |  | test accuracy |  |
| $\mathcal{N}$ | log-likelihood (dev.) | directed | undirected | log-likelihood (dev.) | directed | undirected | log-likelihood (dev.) | directed | undirected | log-likelihood (dev.) | directed | undirected |
| $\Sigma^*$ (EM) | -23.76 | 41.6 | 62.2 | – | – | – | -23.76 | 41.6 | 62.2 | – | – | – |
| DEL1 | -28.83 | 39.7 | 53.5 | -25.30 | 48.8 | 62.7 | -31.50 | 18.3 | 33.3 | -26.25 | 34.5 | 69.0 |
| TRANS1 | -24.75 | 41.2 | 62.5 | -24.89 | 40.6 | 61.7 | -25.04 | 29.4 | 57.8 | -24.94 | 31.3 | 56.4 |
| DEL1ORTRANS1 | -25.38 | 57.6 | 69.0 | -25.10 | 55.3 | 67.2 | -25.18 | 47.3 | 63.6 | -25.39 | 47.1 | 62.2 |
| LENGTH | -24.06 | 45.5 | 64.9 | -24.91 | 45.0 | 64.1 | -24.06 | 45.5 | 64.9 | -24.91 | 45.0 | 64.1 |
| DYNASEARCH | -24.53 | 47.6 | 65.3 | -24.93 | 46.3 | 64.1 | -24.55 | 39.7 | 61.9 | -24.97 | 37.9 | 61.2 |
| TRANS2 | -24.48 | 40.2 | 61.8 | -24.99 | 41.2 | 62.7 | -24.48 | 40.2 | 61.8 | -24.99 | 41.2 | 62.7 |
| DEL1ORTRANS2 | -24.70 | 51.2 | 65.0 | -24.91 | 57.3 | 67.9 | -24.48 | 37.9 | 59.3 | -24.99 | 42.8 | 62.9 |
| DEL1SUBSEQ | -31.35 | 26.6 | 41.0 | -25.28 | 47.9 | 62.1 | -31.77 | 18.9 | 35.9 | -26.29 | 35.1 | 60.6 |

Table 4.7: Performance of differently-trained models $\vec{\theta}_{\mathrm{CE}}$ and the resulting model when each is used to initialize MLE training (on the training data) with EM.

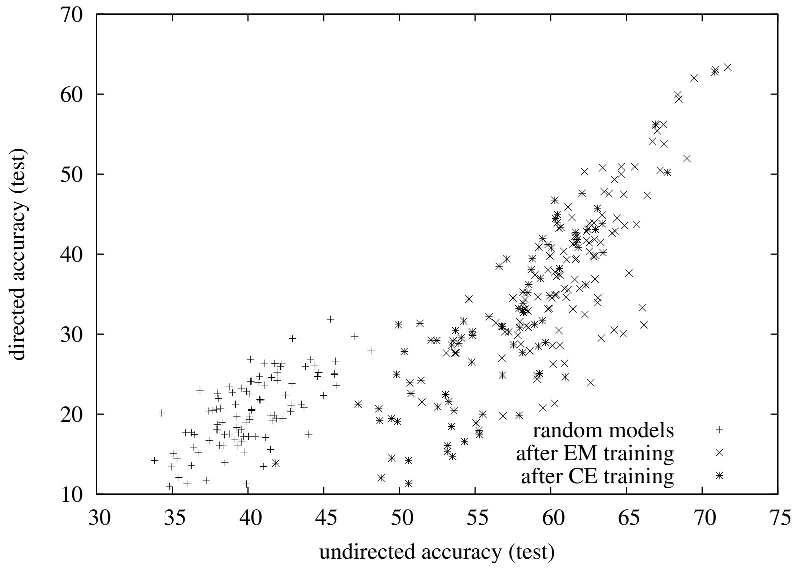


Figure 4.10: Accuracy on test data (directed *vs.* undirected): 100 random models before training, after EM training, and after CE/DEL1ORTRANS1 training.

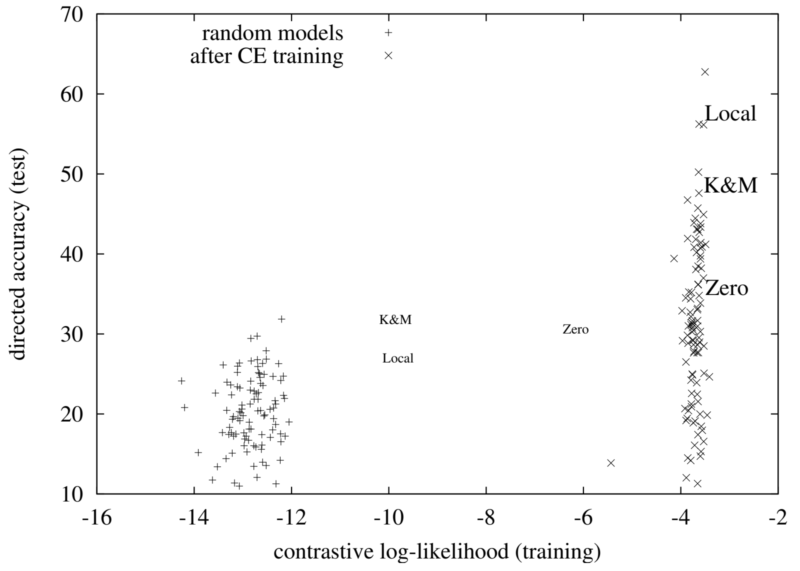


Figure 4.11: Accuracy on test data *vs.* contrastive/DEL1ORTRANS1 log-likelihood on training data: 100 random models (and our three initializer) before and after CE training. Small labels correspond to our initializers before training, large labels to after training. Compare with Figure 3.7 on page 51.

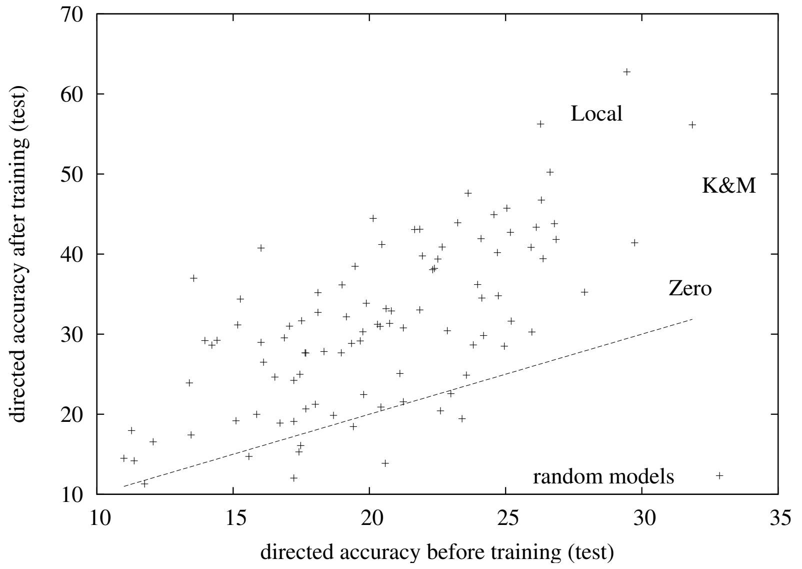


Figure 4.12: Directed accuracy on test data: 100 models after CE/DEL1ORTRANS1 training *vs.* before training. Compare with Figure 3.8 on page 52.

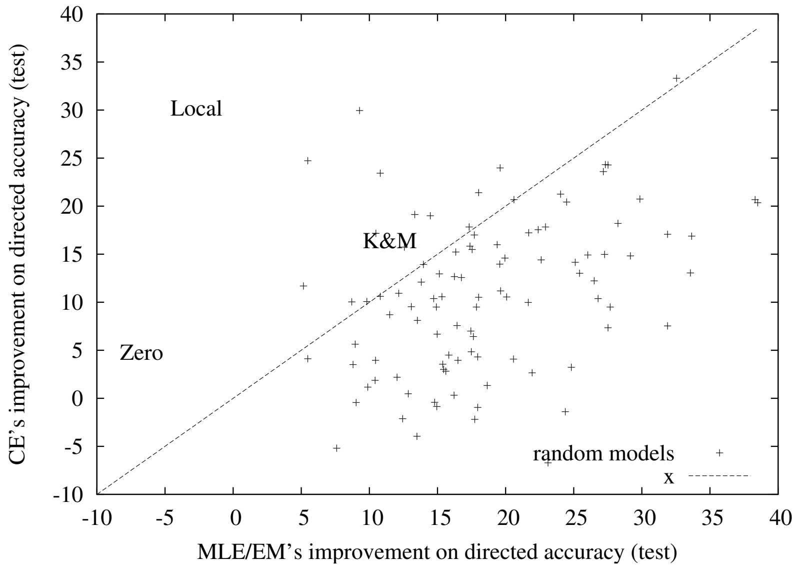


Figure 4.13: Absolute improvement (or reduction) of directed accuracy on test data via CE/DEL1ORTRANS1 *vs.* via MLE/EM. No regularization or smoothing was applied. CE makes better use of good initializers, but does not typically improve a random initializer as much as EM. The line is $y = x$.

### 4.6.6 Semisupervised Training

In Section 3.4.2 we considered the use of the annotated development dataset to construct an initializer (i.e., train supervised and initialize unsupervised training with the learned model) and to construct a prior for MAP estimation—both as alternatives to our unsupervised estimation/supervised selection strategy. We found that, with enough annotated data, either method was preferable to our strategy, though none of the methods tested performed as well as simple supervised training on the development dataset (except when only very small amounts of annotated data were used). (See Figure 3.10 on page 57, Section 3.4.2). We argued in Section 3.4.2 that our method is nonetheless a reasonable way to evaluate performance of unsupervised estimation methods.

Here we consider how CE fares when the annotated data are used to build an initializer.[13] Figure 4.14 illustrates the results. With more than ten annotated examples, supervised initialization followed by CE outperforms CE followed by supervised selection. However, performance of the latter holds steady for the entire range of development set sizes: the same performance is achieved with 5 annotated sentences as with 500. (At the low end of the curve, this method outperforms supervised training on the 5 examples.) EM training using a supervised initializer, however, outperforms CE for smaller annotated development datasets; they are about the same with more annotated data.

As in Section 3.4.2, these results cast some doubt on our unsupervised training/supervised selection methodology. We suggest that the advantages of CE over EM as an *unsupervised estimator* motivate further exploration on improved *semisupervised* estimation methods that make use of contrast in the objective function. Such an exploration is beyond the scope of this thesis.

## 4.7 Part-of-Speech Tagging Experiments

In this section we describe highly successful experimental results using CE for another task, namely training a part-of-speech tagger from unannotated word sequences. These experiments were presented in Smith and Eisner (2005a); they are inspired by those

[13] In principle, a prior could be constructed, as well, though this did not work as well as supervised initialization in Section 3.4.2. Further, the form of our prior for CE is different (a Gaussian) and it is not as obvious how to construct it. Therefore our semisupervised experiments for CE are limited to those involving supervised initialization.

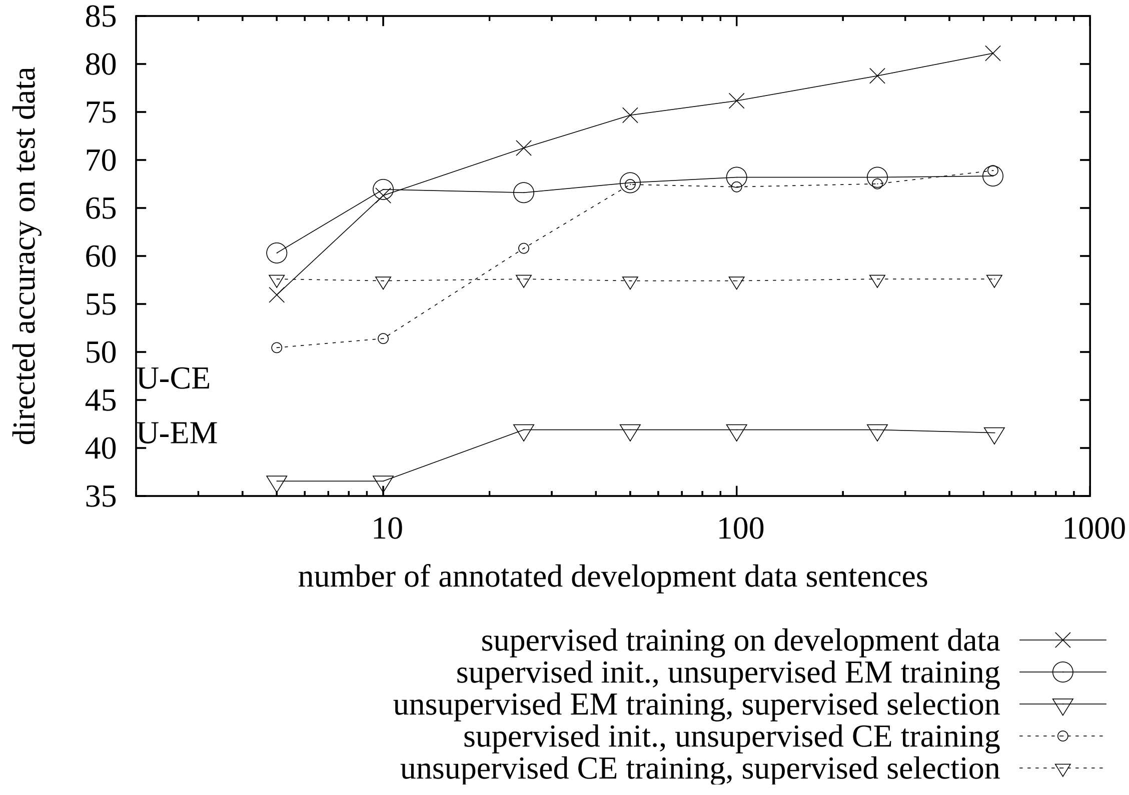


Figure 4.14: Different uses of the development dataset, as the size of the development dataset changes. The plot compares our standard setting of unsupervised estimation/supervised selection (both with MAP/EM and CE/DEL1ORTRANS1) against the use of the annotated development dataset to construct an *initializer* for MLE/EM or unregularized CE/DEL1ORTRANS1 training. (Smoothing/regularization had very little effect on accuracy in these conditions.) The "U" labels at the left correspond to unsupervised model selection.

in Merialdo (1994).[14] We train a trigram tagger using only unlabeled data, assuming complete knowledge of the tagging dictionary.[15] In these experiments, we varied the amount of data available (12,000–96,000 words of Wall Street Journal text), the heaviness of smoothing, and the estimation criterion. In all cases, training stopped when the relative change in the criterion fell below $10^{-4}$ between steps (typically $\leq 100$ steps). For this corpus and tag set, on average, a tagger must decide between 2.3 tags for a given token.

The stochastic model trained by EM was identical to Merialdo's: a second-order hidden Markov model. We smoothed using a symmetric Dirichlet prior with single parameter $\lambda$ for all distributions ($\lambda$-values from 0 to 10 were tested, as in the parsing experiments). The model was initialized by making each log-probability zero (the uniform posterior). The log-linear models trained by CE used the same feature set, though of course the feature weights are no longer log-probabilities and there are no sum-to-one constraints. In addition to an unsmoothed trial, we applied diagonal Gaussian priors (quadratic penalty) with $\sigma^2$ ranging from 0.1 to 10. The models were initialized with all $\theta = 0$. 5,000 words of text were used for unsupervised model selection.[16]

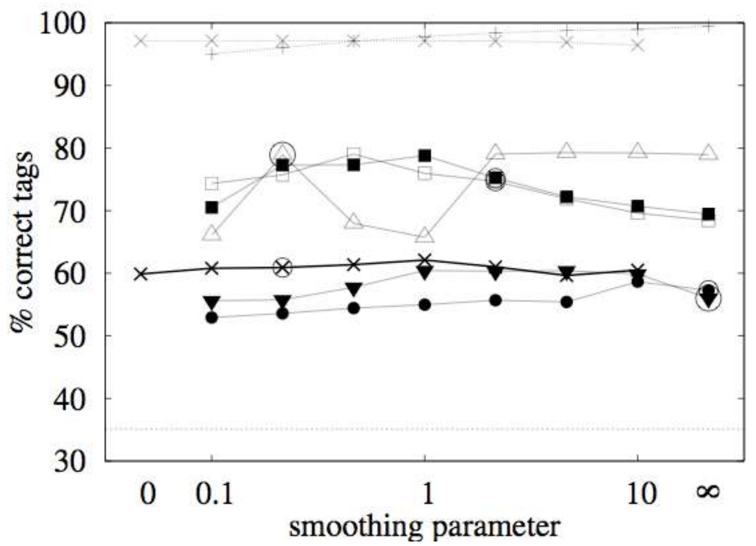
The plot in Figure 4.15 shows the Viterbi accuracy of each criterion trained on the 96,000-word dataset as smoothing was varied; the table shows, for each (criterion, dataset) pair the performance of the unsupervisedly selected $\lambda$ or $\sigma^2$ and the one chosen by an oracle. LENGTH (78.9% with 96K words of training data), TRANS1 (74.7%), and DEL1-ORTRANS1 (75.2%) are consistently the best, far out-stripping EM (60.9%). These gains dwarf the performance of EM on over 1.1 million words (66.6% as reported by Smith and Eisner (2004), ten times the 96K words tested here), even when the latter uses deterministic annealing for improved search (70.0%). (We did not evaluate DA on the smaller datasets explored here.) DEL1 and DEL1SUBSEQ, on the other hand, are poor, even worse than EM on larger datasets.

The LENGTH neighborhood is relatively close to log-linear MLE. The inconsistencies in the LENGTH curve (Figure 4.15) are notable and also appeared at the other training set sizes. Believing this might be indicative of brittleness in Viterbi label selection, we computed the *expected* accuracy of the LENGTH models; the same "dips" were present.

[14]Merialdo was interested in combining labeled and unlabeled data; we focus on the unlabeled-only case.

[15]Without a tagging dictionary, tag names are interchangeable and cannot be directly evaluated on gold-standard accuracy. We address the tagging dictionary assumption in Section 4.7.1.

[16]We did not carry out supervised model selection in these experiments; performance of supervised model selection is expected to lie somewhere between unsupervised selection and the oracle.



| | | 12K | | 24K | | 48K | | 96K | |
|---|---|---|---|---|---|---|---|---|---|
| | | u-sel. | *oracle* | u-sel. | *oracle* | u-sel. | *oracle* | u-sel. | *oracle* |
| + | CRF (supervised) | | *100.0* | | *99.8* | | *99.8* | | *99.5* |
| × | HMM (supervised) | | *99.3* | | *98.5* | | *97.9* | | *97.2* |
| △ | LENGTH | **74.9** | ***77.4*** | **78.7** | ***81.5*** | **78.3** | ***81.3*** | **78.9** | ***79.3*** |
| ■ | DEL1ORTRANS1 | 70.8 | *70.8* | 78.6 | *78.6* | **78.3** | *79.1* | 75.2 | *78.8* |
| □ | TRANS1 | 72.7 | *72.7* | 77.2 | *77.2* | 78.1 | *79.4* | 74.7 | *79.0* |
| × | EM | 49.5 | *52.9* | 55.5 | *58.0* | 59.4 | *60.9* | 60.9 | *62.1* |
| ▼ | DEL1 | 55.4 | *55.6* | 58.6 | *60.3* | 59.9 | *60.2* | 59.9 | *60.4* |
| ● | DEL1SUBSEQ | 53.0 | *53.3* | 55.0 | *56.7* | 55.3 | *55.4* | 57.3 | *58.7* |
| – | random expected | | *35.2* | | *35.1* | | *35.1* | | *35.1* |
| | ambiguous words | | 6,244 | | 12,923 | | 25,879 | | 51,521 |

Figure 4.15: Percent ambiguous words tagged correctly in the 96,000-word dataset, as the smoothing parameter ($\lambda$ in the case of EM, $\sigma^2$ in the CE cases) varies. The model selected from each criterion using unlabeled development data is circled in the plot. Dataset size is varied in the table beneath the plot, which shows models selected using unlabeled development data ("u-sel.") and using an oracle ("*oracle*," the highest point on a curve). Across conditions, some neighborhood roughly splits the difference between supervised models and EM.

This could indicate that the learner was trapped in a local maximum.

### 4.7.1 Removing knowledge, adding features

The assumption that the tagging dictionary is completely known is difficult to justify. While a POS lexicon might be available for a new language, certainly it will not give exhaustive information about all word types in a corpus. We experimented with removing knowledge from the tagging dictionary, thereby increasing the difficulty of the task, to see how well various objective functions could recover. One means to recovery is the addition of features to the model—this is easy with log-linear models but not with classical stochastic models.

We compared the performance of the best neighborhoods (LENGTH, DEL1ORTRANS1, and TRANS1) from the first experiment, plus EM, using three *diluted* dictionaries and the original one, on the 24,000-word dataset. A diluted dictionary adds (tag, word) entries so that rare words are allowed with *any* tag, simulating zero prior knowledge about the word. "Rare" might be defined in different ways; we used three definitions: words unseen in the first 500 sentences (about half of the 24,000-word training corpus); singletons (words with count $\leq 1$); and words with count $\leq 2$. To allow more trials, we projected the original 45 tags onto a coarser set of 17 (e.g., RB$*$ $\rightarrow$ADV; see Table B.2).

To take better advantage of the power of log-linear models—specifically, their ability to incorporate novel features—we also ran trials augmenting the model with *spelling* features, allowing exploitation of correlations between *parts* of the word and a possible tag. Our spelling features included all observed 1-, 2-, and 3-character suffixes, initial capitalization, containing a hyphen, and containing a digit.

Figure 4.16 plots tagging accuracy (on ambiguous words) for each dictionary on the 24,000-word dataset. The $x$-axis is the smoothing parameter ($\lambda$ for EM, $\sigma^2$ for CE). Note that the different plots are not comparable, because their $y$-axes are based on different sets of ambiguous words.

So that models under different dilution conditions could be compared, we computed accuracy on *all* words; these are shown in Table 4.8. The reader will notice that there is often a large gap between unsupervised and oracle model selection; this draws attention to a need for better unsupervised regularization and model selection techniques. The supervised model selection used elsewhere in the thesis is one solution to this problem.

| | | tagging dictionary | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | all train & dev. | | first 500 sents. | | count $\geq 2$ | | count $\geq 3$ | |
| estimation | model | u-sel. | *oracle* | u-sel. | *oracle* | u-sel. | *oracle* | u-sel. | *oracle* |
| MAP/EM | trigram | 78.0 | *84.4* | 77.2 | *80.5* | 70.1 | *70.9* | 66.5 | *66.5* |
| CE/DEL1ORTRANS1 | trigram | 78.3 | *90.1* | 72.3 | *84.8* | 69.5 | *81.3* | 65.0 | *77.2* |
| | + spelling | 80.9 | *91.1* | 80.2 | **90.8** | **79.5** | **90.3** | 78.3 | **89.8** |
| CE/TRANS1 | trigram | **90.4** | *90.4* | 80.8 | *82.9* | 77.0 | *78.6* | 71.7 | *73.4* |
| | + spelling | 88.7 | *90.9* | **88.1** | *90.1* | 78.7 | *90.1* | **78.4** | *89.5* |
| CE/LENGTH | trigram | 87.8 | *90.4* | 68.1 | *78.3* | 65.3 | *75.2* | 62.8 | *72.3* |
| | + spelling | 87.1 | **91.9** | 76.9 | *83.2* | 73.3 | *73.8* | 73.2 | *73.6* |
| random expected | | | 69.5 | | 60.5 | | 56.6 | | 51.0 |
| ambiguous words | | | 13,150 | | 13,841 | | 14,780 | | 15,996 |
| ave. tags/token | | | 2.3 | | 3.7 | | 4.4 | | 5.5 |

Table 4.8: POS tagging results using EM and CE, with diluted dictionaries. We report the percent of *all* words correctly tagged in the 24K dataset, as the tagging dictionary is diluted. Unsupervised model selection ("u-sel.") and oracle model selection ("*oracle*") across smoothing parameters are shown. Note that we evaluated on *all* words (unlike Figure 4.16) and used 17 coarse tags, giving higher scores than in Fig. 4.15. Boldface denotes the best score in a column.

Without spelling features, all models perform worse as knowledge is removed. But LENGTH suffers most substantially, relative to its initial performance. What is the reason for this? LENGTH (like EM) requires the model to explain why a given sentence was seen instead of some other sentence of the same length. One way to make this explanation is to manipulate emission weights (i.e., for (tag, word) features): the learner can construct a good class-based *unigram* model of the text (where classes are tags). This is good for the LENGTH objective, but not for learning good POS tag sequences.

In contrast, DEL1ORTRANS1 and TRANS1 do not allow the learner to manipulate emission weights for words not in the sentence. The sentence's goodness must be explained in a way other than by the words it contains: namely through the POS tags. To check this intuition, we built local normalized models $p(\text{word} \mid \text{tag})$ from the parameters learned by TRANS1 and LENGTH. For each tag, these were compared by KL divergence to the empirical lexical distributions (from labeled data). For the ten tags accounting for 95.6% of the data, LENGTH more closely matched the empirical lexical distributions. LENGTH is learning a correct distribution, but that distribution is not helpful for the task.

The improvement from adding spelling features is striking: DEL1ORTRANS1 and $transone$ recover nearly completely (modulo the model selection problem) from the diluted dictionaries. LENGTH sees far less recovery. Hence even our improved feature sets cannot compensate for the choice of neighborhood. This highlights our argument that a neighborhood is not an approximation to log-linear MLE; LENGTH tries very hard to approximate log-linear EM but requires a good dictionary to be on par with the other criteria. Good neighborhoods, rather, perform well in their own right.

In Section 7.2, we will compare these methods for learning POS tags by using the hypothesized tags as input to various learners for Model A.

## 4.8 Model U

One advantage of working with log-linear models is that we no longer need a stochastic process whose stochastic steps correspond to parameters in the model ($\vec{\theta}$). Instead, we just need feature functions on structures in $\mathcal{W}$ and inference and decoding algorithms (Section 2.4). We suggest here a new model (called Model U) that is log-linear in form and does not correspond to any stochastic process.

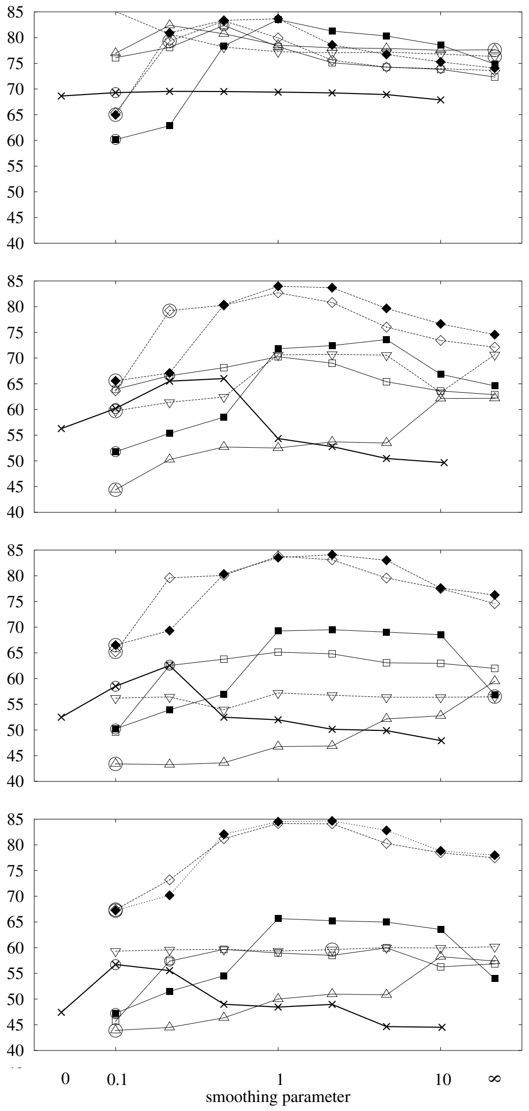
Let $\mathcal{W}$ be the set of all *undirected*, projective trees over sequences in $\Sigma^*$. Recall that

All train & development words are in the tagging dictionary:

Tagging dictionary taken from the first 500 sentences:

Tagging dictionary contains words with count $\geq 2$:

Tagging dictionary contains words with count $\geq 3$:



| | CE/Del1OrTrans1 | CE/Trans1 | CE/Length | MAP/EM |
|---|---|---|---|---|
| trigram HMM | ■ | □ | △ | × |
| + spelling | ◆ | ◇ | ▽ | |

Figure 4.16: Percent ambiguous words tagged correctly (with coarse tags) on the 24,000-word dataset, as the dictionary is diluted and with spelling features. Each plot corresponds to a different level of dilution. Models selected using unlabeled development data are circled. These plots (unlike Table 4.8) are *not* comparable to each other because each is measured on a different set of ambiguous words.

Model A defined distributions over *directed*, rooted projective trees. Model U puts those structures into equivalence classes so that trees that are equivalent up to the direction of the edges receive the same score. This is very similar to **link grammar**, a formalism that describes undirected, labeled dependency graphs (Sleator and Temperley, 1993); a stochastic variant has also been described (Lafferty *et al.*, 1992). The major differences are that link grammar is lexicalized, labels the edges in the graph, and allows cycles.

For a given sequence $\mathbf{x} \in \Sigma^*$, let $\mathcal{Y}_\mathbf{x}$ be defined by a set of edges $\langle i, j \rangle$ such that $0 < i < j \leq |\mathbf{x}|$. The edges are constrained to form a projective tree (see Section 2.1.2). The features in Model U correspond to all of the edges in a tree, so that

$$\ddot{p}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) = \frac{\exp \sum_{\langle i,j \rangle \in \mathbf{y}} \theta_{\mathbf{edge}(x_i, x_j)}}{\ddot{Z}_{\vec{\theta}}(\mathcal{W})} = \frac{\exp \sum_{\langle x, x' \rangle \in \Sigma^2} \theta_{\mathbf{edge}(x,x')} \cdot f_{\mathbf{edge}(x,x')}(\mathbf{x}, \mathbf{y})}{\ddot{Z}_{\vec{\theta}}(\mathcal{W})} \tag{4.25}$$

That is, a feature fires for every edge in the tree, and the total score of a tree is a product of weights, one per edge.

The motivation for Model U is that it does not fragment probability mass among differently directed trees. This may make sense for the purposes of learning, since the posterior will focus only on connections and not on which word is the parent and which is the child. Model A might get overcommitted to a particular wrong link direction (at the grammar level, not simply at the level of an individual parse tree) that is locally optimal, and as a result never learn a pattern that would become apparent if the link were reversed (see the discussion of determiners and nouns in Section 3.4.3). Also, Model U gets away with far fewer parameters than Model A (only $|\Sigma|^2$, fewer than half of Model A's $2|\Sigma|^2 + 9|\Sigma|$). It does not, however, model valency (stopping and continuing)—though the model and dynamic programming algorithms could be modified to handle valency.

### 4.8.1 Dynamic Programming

The size of $\mathcal{Y}_\mathbf{x}$ under Model U is exactly $1/|\mathbf{x}|$ times the size of $\mathcal{Y}_\mathbf{x}$ under Model A. To see why, note that if we select a root in the undirected tree, the direction of all edges is fully determined. There are $|\mathbf{x}|$ possible roots, so $|\mathbf{x}|$ directed trees are collapsed into one undirected tree. The dynamic programming equations for this model are very similar to those for Model A, except that the $\lambda$s are different, and for the purposes of the algorithm we simply fix the root of the tree to be $x_1$. The equations are given in Section A.3.

|  | undirected accuracy (test set) |  |
|---:|---:|---:|
| $\mathcal{N}$ | Model U | Model U + spelling |
| DEL1 | 52.7 | 50.6 |
| TRANS1 | 51.8 | 54.3 |
| DEL1ORTRANS1 | 49.6 | 51.4 |
| DYNASEARCH | 50.7 | 54.0 |
| TRANS2 | 51.4 | 55.5 |
| DEL1ORTRANS2 | 50.7 | 54.8 |
| overall: TRANS2, $\sigma^2 = 10^{-1/3}$ |  | 55.5 |

Table 4.9: Performance of Model U. Supervised model selection, here on the undirected attachment accuracy score, was applied across values of $\sigma^2$ (the regularization hyperparameter), and also the neighborhood and feature set (in the last line).

Note that if we wanted to add stopping and continuing features to Model U, we could do so, but we would need to further modify the dynamic programming algorithm.

### 4.8.2 Experiments

MLE with Model U—which is log-linear by design—suffers from all the computational difficulties we noted to motivate contrastive estimation; it is simply not a practical option. We present experimental results with Model U using several neighborhoods in contrastive estimation. All trials were initialized with a Zero model (better initializers might be possible).

Because Model U scores only undirected trees, we report only undirected accuracy. The first numerical column of Table 4.9 shows how Model U performs on undirected accuracy. We do not measure directed accuracy, because Model U does not predict directed trees. Model U performs rather badly, not even achieving the 62.1% achieved by the ATTACH-RIGHT baseline on this task. We do not explore here why it is bad (a likely issue is that it lacks valency features, cf. Model A; another is the initializer).

### 4.8.3 Adding Features

Adding features to a model is often an effective way to improve its performance. To illustrate how easily this can be done in the log-linear setting, we consider here the addition of *spelling* features to Model U. This moves in the direction of lexicalization (word features). Indeed, the original paper about CE (Smith and Eisner, 2005a) motivated the approach by citing a desire to incorporate spelling features into unsupervised tagging models as described in Section 4.7.

We added local features to Model U that relate the part-of-speech tag of one word to the prefix or suffix of a word it is linked to. These features take the form "tag $x$ precedes and is linked to a word with suffix $s$" where $s$ is 1–3 characters long. Similar "follows" features and "prefix" features were also included. For the purposes of these features, upper-case letters were collapsed to lower-case, and all digits were collapsed to a '0'. The model for this dataset contained 103,352 features.[17] All feature weights $\vec{\theta}$ were initialized to zero. Results are shown in Table 4.9.

Notice that in most cases, adding spelling features improves performance, if slightly. In the overall-selected model, trained with CE/TRANS2 with the spelling features, the following were the highest-weighted spelling features (the slash here implies a link and a linear order, e.g., -%/TO means that a word ending in % preceded *and* was linked to a tag token TO):

- -%/TO (1.48); i.e., a word with character suffix % is linked to and precedes a tag TO
- %-/TO (1.48)
- CD/-% (1.45)
- CD/%- (1.45)
- -*i*/VBP (1.38); -*i* words in English are usually proper nouns
- -*a*/JJ (1.36); -*a* words are usually the word *a*
- -*to*/VB (1.19)
- *be*-/VBN (1.14)
- -./CD (1.13); -. words are usually abbreviations, sometimes for months, like *Oct.*
- *to*-/VB (1.01)
- EX/-*s* (0.99)

[17]Only features that might have occurred in some structure in $\mathcal{Y}_{\mathbf{x}^t}$ for some training data sentence $\mathbf{x}^t$ were counted; others would never change from 0 under our optimization algorithm and therefore were not explicitly represented.

- VBP/*-t* (0.99)
- *ha-*/VBN (0.95)
- *he-*/VBZ (0.95)
- *fe-*/CD (0.94); *fell* is the most common *fe-* word
- CD/*mil-* (0.92)

Incorporating valence features into Model U, or spelling features into Model A, is left to future work. We emphasize again that doing either (indeed, even experimenting with Model U at all) requires unsupervised estimation methods for *log-linear* models, an obstacle removed by contrastive estimation.

## 4.9 Related Work

We have already noted that many approximations to maximum likelihood can be described as instances of contrastive estimation. One other very recent paper is worthy of mention here, because its goals are similar to those of CE: Xu, Wilkinson, Southey, and Schuurmans (2006).

Early in the chapter we noted the rise of discriminative methods in supervised machine learning. Maximum-margin Markov ($M^3$) networks (Taskar, Guestrin, and Koller, 2003) are an example of a supervised discriminative learning method that uses kernel methods to find a weighted model that maximizes the margin between correct hidden structures and incorrect alternatives. Maximum-margin learning has been applied to supervised estimation of weighted grammars as well (Taskar *et al.*, 2004).

Xu *et al.* (2006) applied $M^3$ networks to unsupervised learning of sequence models ($\mathbf{y}$ is a sequence of labels for the sequence of observed symbols $\mathbf{x}$). To do this, first consider the supervised problem (assuming we knew the correct label sequences) as an optimization problem where the objective's dependence on the correct label sequences $\vec{\mathbf{y}}^{\mathrm{t}}$ is made explicit:[18]

$$f\left(\vec{\mathbf{y}}^{\mathrm{t}}\right) = \min_{\vec{\theta}} \frac{\beta}{2}\|\vec{\theta}\|^2 + \sum_{i=1}^{|\vec{\mathbf{x}}^{\mathrm{t}}|} \max_{\mathbf{y}} \max\left(0, \Delta(\mathbf{y}, \mathbf{y}_i^{\mathrm{t}}) - \vec{\theta} \cdot \left(\vec{f}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y}_i^{\mathrm{t}}) - \vec{f}(\mathbf{x}_i^{\mathrm{t}}, \mathbf{y})\right)\right) \tag{4.26}$$

[18] We keep the notation as close to that in the rest of the thesis as possible.

where $\Delta(\mathbf{y}, \mathbf{y}^{\mathrm{t}_i})$ counts the labeling errors in $\mathbf{y}$ with respect to $\mathbf{y}_i^{\mathrm{t}}$. The dual of this objective function can be written as a quadratic program in variables that correspond to relabelings of elements of the sequences $\mathbf{x}_i^{\mathrm{t}}$ and adjacent pairs of those elements. The key step taken by Xu *et al.* is to treat the $\vec{y}^{\mathrm{t}}$ as another variable to optimize over. Through a change of variable, Xu *et al.* eliminate all references t $\vec{\mathbf{y}}^{\mathrm{t}}$, instead referencing pairwise comparisons between the labels of the all elements in the training data.

Two kinds of constraints are then added. The first is a balance constraint that prevents the discovery of a trivial solution where all $x_i$ are labeled the same way. The second proposed constraint enforces consistency between the singleton and pairwise variables in the reformulated problem. The latter would make the problem non-convex, so Xu *et al. relax* these constraints. They also relax the Boolean variables to the interval $[0, 1]$. The reader is encouraged to see the paper for the details of these relaxations. The result is a semidefinite program, which is convex. Solving the semidefinite program was found to be too computationally intensive to be practical, so two different approximations are presented. Xu *et al.* met with success on synthetic and small protein datasets. It is unclear whether the technique generalizes to weighted grammars like Model A, but if so a comparison with CE is certainly warranted.

Stepping back into the broader language learning setting (including humans, not just machines), negative evidence has figured heavily in the cognitive and formal literature on language learning (Marcus, 1993, see discussion in Section 1.3.2). Negative evidence there usually refers to explicit feedback; the child makes a mistake and the parent responds with a correction. Our approach to negative evidence, which we believe is entirely novel, defines a function from inputs to sets of inputs. These are treated in a "soft" manner that never makes a strong negative judgment about any particular example. This is very different from the usual formal language learning setting (e.g., Gold, 1967), in which examples are positive if they are in the language and negative if they are not.

## 4.10 Future Work

We suggest a few avenues for future directions of research on contrastive estimation. One idea is to *weight* the implicit negative examples in the neighborhood $\mathcal{N}$. This allows us to tell the learner that some examples in the neighborhood are expected to be worse than others. For example, the weighted neighborhood might express that swapping two words is far more damaging than scrambling three or more words. This approach seems intuitive and even has a probabilistic interpretation. If each $\mathbf{x}' \in \mathcal{N}(\mathbf{x})$ has a positive score $\zeta(\mathbf{x}'; \mathbf{x})$, then we can think of $\zeta$ as another feature weight in the log-linear model. The weighted-neighborhood contrastive objective would be (compare with Equation 4.2):

$$\operatorname*{argmax}_{\vec{\theta}} \prod_{i=1}^{|\vec{x}^{\mathrm{t}}|} \frac{p_{\vec{\theta}}(\mathbf{x}_i^{\mathrm{t}})}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i^{\mathrm{t}})} \zeta(\mathbf{x}; \mathbf{x}_i^{\mathrm{t}}) p_{\vec{\theta}}(\mathbf{x})} \tag{4.27}$$

This special feature is not trained; rather, it is fixed by the definition of the weighted neighborhood. We will consider the use of an untrained feature in Chapter 6, when we talk about using predefined *bias* among hypotheses to improve estimation.

A possible confound in our comparison of MAP estimation with regularized CE is the use of a Dirichlet prior for the former (a stochastic model with simplex-constrained weights) and a Gaussian for the latter (whose parameters are unconstrained).[19] We also might define alternative priors in contrastive estimation. We experimented only with a Gaussian here, but others are certainly possible (e.g., Goodman, 2004; Kazama and Tsujii, 2005). As in MLE learning, the appropriate use of annotated data should be more carefully considered; here we used it only for model selection. See the discussion in Section 3.7. Three additional ideas deserve more exposition: task-based neighborhoods, future directions on incorporating more linguistic knowledge, and other areas of applicability.

### 4.10.1 Task-Based Neighborhoods

Smith and Eisner (2005b) described some cases where contrastive estimation could be applied to text processing problems like spelling and punctuation correction. We call these **task-based** neighborhoods.

Manufacturing supervised data for correction tasks like these is easy: take clean text, and mangle it. This is a classic strategy for training accent and capitalization restoration (Yarowsky, 1994): just delete all of the accents or upper-case letters from clean text. For tasks like spelling and punctuation correction, we don't know the mangling process. The errors are not simply an omission of some part of the data; they are whatever mistakes humans (or speech recognition systems, or translation systems, etc.) make. Without a corpus of errors, this cannot be modeled empirically.

[19]Though note that *unregularized* CE performed extremely well with the right neighborhood and initializer.

We suggest it may be possible to get away with not knowing which mistakes a human would make; instead, try to distinguish each observed good sentence $\mathbf{x}$ from *many* differently punctuated (presumably *mis*punctuated) versions. This is not as inefficient as it might sound, because lattices allow efficient training. In CE terms, the set of all variants of the sentence with errors introduced is the neighborhood $\mathcal{N}(\mathbf{x})$.

For spelling correction, this neighborhood might be (for some $k$):

$$\{\mathbf{x}' \in \Sigma : \forall i \in \{1, 2, ..., |\mathbf{x}|\}, \mathbf{Levenshtein}(x_i, x'_i) \leq k\} \tag{4.28}$$

where **Levenshtein** is the edit distance between two words (Levenshtein, 1965). This neighborhood, like the others, can be represented as a lattice—in this case it will have a "sausage" shape. We might consider augmenting $\Sigma$ to include non-words that are within $k$ edits of real words. A neighborhood for punctuation correction might include all alternately-punctuated sentences, producible using a finite-state transducer.

Notice that the neighborhoods are orthogonal to the model. Model A, Model U, or any other model of interest (for which the computation is not too expensive) could be trained with these task-based neighborhoods.

The application of these models is not as simple as decoding was in parsing, when $\mathbf{Y}$ was the hidden structure that we cared about. Decoding now involves taking a sentence that might contain errors and selecting the sentence from its neighborhood (or possibly from $\Sigma^*$, not addressed here) that is most probable, according to our model. Note that now the neighborhoods are centered on the observed, possibly incorrect sentences, rather than on correct training examples. This approach is similar to certain noisy-channel spelling correction approaches (Kernighan, Church, and Gale, 1990) in which, as for us, only correctly-spelled text is observed in training. Like them, we have no "channel" model of which errors are likely to occur—only a set of possible errors that implies a set of candidate corrections.[20] The model we propose is a language model—one that incorporates induced grammatical information—that might then be combined with an existing channel model of errors. The other difference is that this approach would attempt to correct the entire sequence at once, making globally optimal decisions, rather than trying to correct each word individually. An obvious application is optical character recognition of full texts.

A subtlety is that the quantity we wish to maximize is a sum. Letting $\mathbf{x}$ be the noisy

[20]The neighborhood could perhaps be weighted to incorporate a channel model, so that we consider not only the probability of each candidate correction but also its similarity to the typed string.

observed input:

$$\hat{\mathbf{x}} = \operatorname*{argmax}_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} p_{\vec{\theta}}(\mathbf{x}') = \operatorname*{argmax}_{\mathbf{x}' \in \mathcal{N}(\mathbf{x})} \sum_{\mathbf{y} \in \mathcal{Y}_{\mathbf{x}'}} p_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) \tag{4.29}$$

This is intractable in general; it is an instance of the *consensus problem*, which is NP-hard (Casacuberta and de la Higuera, 2000; Lyngsø and Pedersen, 2002; Sima'an, 2002). Most often, researchers avoid the issue by maximizing over $\mathbf{y}$ as well as $\mathbf{x}'$.[21]

Another kind of neighborhood can be defined for a specific *system*. Let the neighborhood consist of mistakes made by the system, and retrain it (or a new component) to contrast the correct output with the actual errors. In parsing, this is treated as a supervised *reranking* problem (Collins, 2000; Charniak and Johnson, 2005); recently hidden variables have been incorporated by Koo and Collins (2005). Examples of this have also been applied in acoustic modeling for speech recognition, where the neighborhood is a lattice containing acoustically-confusable words (Valtchev, Odell, Woodland, and Young, 1997); the hidden variables are the alignments between speech segments and phones. Another example from speech recognition involves training a language model on lattices provided by an acoustic model (Vergyri, 2000; Roark, Saraclar, Collins, and Johnson, 2004); here the neighborhood is defined by the acoustic model's hypotheses and may be weighted. A more ambitious line of work would be to induce grammatical models for error correction in models that, for reasons of speed or estimation, do not use syntax themselves (e.g., in machine translation).

Neighborhood functions might also be iteratively *modified* to improve a system in a manner similar to bootstrapping (Yarowsky, 1995) and transformation-based learning (Brill, 1995).

### 4.10.2 More Linguistic Domain Knowledge

Arguably the most important argument for contrastive estimation is that it permits more expressive models with new features that do not fit into stochastic grammars. We moved in this direction by adding spelling features to Model U in Section 4.8.3. Lexical, spelling, and (if available) morphological features may be useful in grammar induction, depending on the language and the amount of training data. Novel uses of cross-lingual information are an exciting area where log-linear models might be helpful (Yarowsky and

[21] Three recent papers addressing instances of this problem in very different ways are Matsuzaki *et al.* (2005), May and Knight (2006), and Finkel, Manning, and Ng (2006).

Ngai, 2001; Smith and Smith, 2004; Kuhn, 2004; Smith and Eisner, 2006b), availing the learner of new information sources.

In the contrastive setting, linguistic phenomena like morphology can influence the model's features, the neighborhood, or both. A morphology-based neighborhood might guide the learner toward tree structures that enforce long-distance inflectional agreement. Other interesting neighborhoods might treat function and content words differently.

One may wonder about the relevance of word order-based neighborhoods like DEL1-ORTRANS1 to languages that do not have strict word order. This is an open and important question, and it will be partially addressed when CE is examined for languages like Bulgarian that have free(r) word order than English. Just as good syntax models for languages with this property may require re-thinking the models themselves (Chapter 6 of this thesis; Hoffman, 1995), good neighborhoods for learning may also have to be developed.

### 4.10.3 Other Applications

This thesis is focused on natural language parsing, but contrastive estimation is a very general model estimation technique. Models of discrete sequences—including weighted grammars and hidden Markov models—and other discrete structures have many applications, including computational biology and image processing.

To give one simple example, profile HMMs (Durbin *et al.*, 1998) are sequence models that are intended to model biological protein families (e.g., proteins with a similar structure or function). They can be estimated using EM from a set of positive examples. However, a molecular biologist might know (for example) that the presence of a particular amino acid is absolutely essential for a protein to have a particular structure.[22] A contrastive neighborhood might be designed that deletes or replaces that amino acid, wherever it is found, possibly in combination with other perturbations of the sequence. This would lead to a model that better predicts the function of interest. Note that this is a way of incorporating a global constraint into a locally factored model (an HMM).

Contrastive estimation is applicable to any problem where domain knowledge allows us to implicate a set of negative examples ("on average") for each positive observed example. Examples of domain knowledge suggested for future exploration include common spelling or grammar mistakes. CE is a very good fit for log-linear models for compu-

[22]Example due to K. R. Thickman, personal communication.

tational reasons, but there is no reason it cannot be applied in principle to any learning problem with the implicit negative evidence property.

## 4.11 Summary

This chapter presented a novel generalization of many unsupervised parameter estimation techniques: contrastive estimation. CE can be understood intuitively as a way of capitalizing on implicit negative evidence, and the choice of implicit negative evidence can allow the incorporation of domain knowledge. CE generalizes many earlier approaches, including MLE. We have shown some computationally efficient examples of CE for learning natural language structure, and demonstrated that they significantly outperform MLE when used to estimate Model A, with both supervised model selection and unsupervised model selection to choose regularization parameters and the initializer. See Table 4.10. We also showed that CE is substantially more successful than MLE/EM at learning to disambiguate part-of-speech tags in unannotated text.

We note that the parsing results in this chapter are not entirely conclusive: it is not clear that CE with any given neighborhood is a better objective function than likelihood. As illustrated in Figure 4.13, training using CE does not improve random models as much as training by EM. It *does* outperform EM with more carefully chosen initializers, including the "Zero" initializer. (Our experiments are fair in the sense that model selection for MAP/EM and CE/DEL1ORTRANS1 was across approximately the same number of trials: 3 initializers $\times$ six (for CE) or seven (for EM) regularization settings.) Of course, all of these results are presented with the caveat that they have focused on a single dependency grammar induction task that is somewhat artificial in its definition (see Section 1.4), rather than a serious application of language technology.

We showed how CE could be used to estimate a model of structures not representable by a stochastic grammar (Model U), and improved performance by adding spelling features. We have suggested future work in this area, most notably by designing neighborhoods for specific tasks, or even specific systems.

|  |  | test accuracy |  |
|---|---|---|---|
|  |  | directed | undirected |
|  | ATTACH-RIGHT | 39.5 | 62.1 |
| Model A | MLE/EM (s-sel.) | 41.7 | 62.1 |
|  | MAP/EM (s-sel.) | 41.6 | 62.2 |
|  | CE/DEL1ORTRANS1 (s-sel.) | 57.6 | 69.0 |
| Model U | CE/DEL1ORTRANS2 (s-sel.) | – | 50.7 |
| + spelling | CE/TRANS2 (s-sel.) | – | 55.5 |

Table 4.10: Summary of baselines and key parsing results in this chapter. Note that Model U does not predict directed attachment.

# Chapter 5

# Annealing Techniques for Improved Search

*Life improves slowly and goes wrong fast, and only catastrophe is clearly visible.*
—Edward Teller (1908–2003)

In the last two chapters, we presented the standard approach to unsupervised structure learning—MLE using the EM algorithm—and a new approach, contrastive estimation. We explained and illustrated empirically that both methods correspond to the optimization of a function that is "bumpy" (i.e., not globally concave). EM and the gradient-based optimization method used for CE are **hillclimbers** that, given a starting point, search locally for a parameter estimate that scores better on the given objective function.

This chapter describes methods in which the objective function *changes* over time, to improve the local search. We use the term **annealing** to refer to this practice (here and in the next chapter), in deference to **simulated annealing** (Kirkpatrick, Gelatt, and Vecchi, 1983) and **deterministic annealing** (Rose, Gurewitz, and Fox, 1990), the two best-known instances of these methods. We will show how deterministic annealing (DA) can be applied to MLE with the aim of avoiding shallow local maxima (Ueda and Nakano, 1998).[1] The underlying idea is to solve a sequence of related, progressively more difficult optimization problems, using each solution to initialize the next problem. DA, unlike EM, is not sensitive to the initializing parameter estimate $\vec{\theta}^{(0)}$. We demonstrate experimentally that DA is indeed more effective at optimizing likelihood, but models trained this way do

[1] In principle it could also be applied to CE, though we have not done so here.

not improve *accuracy*.

We therefore generalize DA, presenting **skewed deterministic annealing** (SDA) as an alternative that overcomes this problem. The experimental aim of the chapter is to better understand the extent to which partial-data MLE (on Model A, at least) works badly because of bad local search (as opposed to working badly because of the objective function, addressed in the last chapter).

Techniques described in this chapter were originally presented in Smith and Eisner (2004), though on different models.

## 5.1 Deterministic Annealing

A major difficulty of all of the unsupervised parameter estimation methods described so far—partial-data MLE, its variants, and contrastive estimation—is that the objective function is not globally concave. Global optimization of a "bumpy" function on $\mathbb{R}^n$ with local optima is a notoriously difficult problem (Törn and Zilinskas, 1989; Horst and Pardalos, 1995, *inter alia*). The algorithms for carrying out these approaches to parameter estimation can be expected only to find *local* optima. Of the experimental results, positive and negative, published on unsupervised learning, how much is due to fortuitous initialization and the chance of climbing a good hill or a bad one? This chapter will shed some light on the matter, showing that for maximum likelihood-based grammar induction, better search can improve likelihood *or* accuracy—though neither implies the other, as we have seen.

Deterministic annealing (DA; Rose *et al.*, 1990) is a technique for transforming inconvenient functions into ones that are more straightforward to optimize. For example, it has been used to approximate piecewise-constant functions with a sequence of continuous, differentiable ones (Rao and Rose, 2001), and it has been used to transform bumpy functions into concave ones (as we will see here). DA has another property, which we will discuss in Section 5.4.1 but will not exploit experimentally, that makes it useful for selecting the cardinality of discrete hidden variables. Readers are referred to Rose (1998) for a detailed exposition of DA as a general technique; here we consider DA as a means to improve EM's search (Ueda and Nakano, 1998).

A helpful way to understand DA is to imagine that EM optimizes a wrinkled, bumpy function in many parameters ($\vec{\theta}$ and the posterior over hidden structures $q$). Any hill-climbing method, such as EM, runs the risk of getting stuck on a shallow local maximum, depending on where it starts out (initialization). DA breaks this dependency by initially transforming the bumpy function into a smooth, concave one—one that is easy to optimize globally. Specifically, the global optimum is a model that predicts a uniform posterior distribution over hidden structures. After solving the easy problem, DA transforms the function into a slightly harder one and uses the solution from the last round to initialize training. Since the two functions are very similar, a local maximum of the new one should be nearby (and therefore quick to find), and further we hope that this local maximum is the global one. This continues, iteratively, until the function matches the original one we intended to maximize. There is no formal guarantee of finding the global maximum or even a better local maximum than EM, but in practice DA often performs better at optimizing likelihood.

### 5.1.1 Objective Function View of DA

A helpful way to think about DA is in terms of the sequence of intermediate objective functions. To do this, we follow Csiszár and Tusnády (1984) and Neal and Hinton (1998) in a particular view of the EM algorithm, namely as coordinate ascent in $\vec{\theta}$ (the model parameters) and $q$ (the posterior over hidden structures). Rather than thinking of EM as a method for finding $\vec{\theta}$ alone, then, assume we are also looking for the posterior $q$. (It may be helpful to refer back to Figure 3.3.)

Recall that the E step defines the distribution $q$ to be

$$q(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \leftarrow p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \tag{5.1}$$

Equivalently, we could say that the E step solves an optimization problem, in which $\vec{\theta}$ is fixed (we only optimize over $q$). The problem is to find the closest distribution to $p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})$ by minimizing a Kullback-Leibler divergence (Kullback and Leibler, 1951, denoted here $\mathbf{D}$). Remember that $\tilde{p}$ is the empirical distribution over examples $\mathbf{x}$; we write this as a maximization problem:

$$\max_q -\mathbf{D}\left(\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y} \mid \mathbf{X}) \,\middle\|\, \tilde{p}(\mathbf{X}) \cdot p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right) \tag{5.2}$$

$$\begin{aligned}\equiv \quad & \max_q -\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}\mid\mathbf{X})}\left[\log \tilde{p}(\mathbf{X}) + \log q(\mathbf{Y} \mid \mathbf{X})\right] \\ & \quad +\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}\mid\mathbf{X})}\left[\log \tilde{p}(\mathbf{X}) + \log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right]\end{aligned} \tag{5.3}$$

The expectation of $\log \tilde{p}(\mathbf{X})$ cancels out, giving:

$$\begin{aligned} \ldots \quad &\equiv \quad \max_q -\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log q(\mathbf{Y} \mid \mathbf{X})\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] && (5.4) \\ &\equiv \quad \max_q \mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\mathbf{H}\left(q(\mathbf{Y} \mid \mathbf{X})\right)\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] && (5.5) \end{aligned}$$

$\mathbf{H}$ is the Shannon entropy. $\mathbf{D}$ is trivially minimized when the two distributions are equivalent, so Equation 5.1 is a closed form solution to this problem and no complex optimization routine is needed.

The M step selects $\vec{\theta}$ by solving the complete data maximum likelihood estimation problem with the "empirical" distribution taken to be $\tilde{p}(\mathbf{X}) \cdot q(\mathbf{Y} \mid \mathbf{X})$, which is fixed:

$$\begin{aligned} &\max_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X}, \mathbf{Y})\right] && (5.6) \\ &\quad \equiv \quad \max_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X}) + \log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] && (5.7) \\ &\quad \equiv \quad \max_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X})\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] && (5.8) \\ &\quad \equiv \quad \max_{\vec{\theta}} \mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X})\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] && (5.9) \end{aligned}$$

This follows from the definition of conditional probabilities (5.7), linearity of expectation (5.8), and aggregating over values of $\mathbf{Y}$ (5.9). How easily the M step optimization problem is solved depends on the form of the model. For stochastic grammars based on multinomial distributions, as we have seen, there is a closed form solution in terms of the sufficient statistics (Section 3.3; here, frequency counts). For log-linear models, an auxiliary optimization technique is required (Section 3.5). Note that Equations 5.5 and 5.9 share a term. Now we combine them into a single objective function, $f_{\text{EM}}$:

$$f_{\text{EM}}(\vec{\theta}, q) \overset{\text{def}}{=} \tag{5.10}$$

$$\overbrace{\mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\mathbf{H}\left(q(\mathbf{Y} \mid \mathbf{X})\right)\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right]}^{\text{improved on E step}} \qquad \hspace{-20em} \underbrace{\hphantom{\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right]} + \mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X})\right]}_{\text{improved on M step}}$$

Both the E and M steps improve $f_{\text{EM}}$. The E step fixes $\vec{\theta}$ and optimizes with respect to $q$ (so the third term is a constant). The M step fixes $q$ and optimizes with respect to $\vec{\theta}$ (so the first term is a constant). Alternating E steps and M steps gives a *coordinate ascent* method in $q$ and $\vec{\theta}$.

Beware confusing $f_{\text{EM}}$ with likelihood. Likelihood—the objective we really hope to locally optimize by carrying out EM iterations—is captured in the third addend of Equation 5.10, which does not reference $q$ or even the hidden variable $\mathbf{Y}$. $f_{\text{EM}}$ is an implementational construct that helps us see how EM carries out that local optimization.

Now note that, in $f_{\text{EM}}$, the first term, expected entropy of $q$, is concave in $q$.[2] DA capitalizes on this property for search purposes. The idea is to introduce an additional parameter, $\beta$, which is manipulated during learning:

$$f_{\text{DA}}(\vec{\theta}, q, \beta) = \frac{1}{\beta}\mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\mathbf{H}\left(q(\mathbf{Y} \mid \mathbf{X})\right)\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}\mid\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X})\right] \tag{5.11}$$

First note that $\beta$ does not affect the M step, only the E step. When $\beta = 1$, the objective is exactly $f_{\text{EM}}$. During DA, $\beta$ is gradually increased from a small, near-zero positive value, to 1. Initially, when $\beta$ is close to 0, the first term dominates and the posterior $q$ is forced to stay near the maximum entropy distribution—uniformity.[3] Gradually, over time, $\beta$ is raised to 1, when the objective becomes equivalent to the EM objective. As $\beta$ increases, $q$ is "allowed" to be less and less agnostic about the values of the hidden variables. When $\beta = 0$, the function is convex, and gradually increasing $\beta$ morphs the function into a non-convex objective, arriving at $f_{\text{EM}}$ when $\beta = 1$.

### 5.1.2 DA with Stochastic Grammars

As it happens the implementation of DA with the stochastic grammars discussed in this thesis is very straightforward, with only two modifications to EM. The first is to nest EM within an outer loop that gradually increases $\beta$ from 0 to 1. The second is a change to the E step, which is essentially:

> Let $q(\mathbf{y} \mid \mathbf{x}^{\text{t}}) \propto \left(p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\text{t}})\right)^{\beta}$ for each observed $\mathbf{x}^{\text{t}}$ in $\vec{\mathbf{x}}^{\text{t}}$ and each possible value $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\text{t}}}$.

In both stochastic grammar EM and log-linear grammar EM, this can be done by replacing $\vec{\theta}$ with $\beta \cdot \vec{\theta}$ before running the Inside and Outside algorithms. The effect is to raise

[2]This fact, that the entropy of a multinomial (such as $q(\mathbf{Y} \mid \mathbf{x})$) is concave in the parameters of the multinomial, is one of the first points typically made about Shannon entropy; (Cover and Thomas, 1991, chapter 2).

[3]Not to be confused with the maximum entropy approach to modeling (footnote 20 in Chapter 3 on page 65), which involves a *constrained* maximum entropy problem. Without constraints, as here, maximum entropy for discrete-variable distributions is achieved by a uniform distribution.

```
DETERMINISTIC ANNEALING (stochastic grammars):

1. Initialize $\vec{\theta}^{(0)}$ and let $\beta \leftarrow \beta_0$.

2. Do:

   (a) (E step) For each rule $\mathbf{r} \in \mathcal{R}$ (the rules of the grammar) let $c_{\mathbf{r}} \leftarrow 0$. For each
       example $\mathbf{x}^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$:

       • Run the relevant Inside and Outside algorithms to compute $p_{\beta\vec{\theta}^{(i)}}(\mathbf{x})$. This
         is done by scaling $\vec{\theta}^{(i)}$ by $\beta$ before solving the dynamic programming equa-
         tions. For all $\mathbf{r} \in \mathcal{R}$, let $c_{\mathbf{r}} \leftarrow c_{\mathbf{r}} + \mathbf{E}_{p_{\beta\vec{\theta}^{(i)}}(\mathbf{Y} \mid \mathbf{x}^{\mathrm{t}})}\left[f_{\mathbf{r}}(\mathbf{x}^{\mathrm{t}}, \mathbf{Y})\right]$.

       $\vec{c}$ now contains sufficient statistics for the parameters of the model.

   (b) (M step; same as EM) Compute $\vec{\theta}^{(i+1)}$ by normalizing $\vec{c}$. That is, if $\mathbf{r}$ is in the
       partition $\mathcal{R}_i$ of the grammar rules, let $\theta_{\mathbf{r}} \leftarrow \frac{c_{\mathbf{r}}}{\sum_{\mathbf{r}' \in \mathcal{R}_i} c_{\mathbf{r}'}}$. This is the maximum like-
       lihood estimate given the sufficient statistics.

   (c) If $\vec{\theta}^{(i+1)} \approx \vec{\theta}^{(i)}$ then proceed to step 3; otherwise let $i \leftarrow i + 1$ and go to step 2a.

3. If $\beta = 1$ then stop; otherwise $\beta \leftarrow \min(1, \gamma\beta)$ and go to step 2.
```

Figure 5.1: DA for stochastic grammars.

the relative probabilities of each analysis $\mathbf{y}$ to the power $\beta$, with the Outside algorithm taking care of renormalization to ensure valid sufficient statistics. The M step does not change. See Figure 5.1.

It is helpful to think of the model $p_{\beta\vec{\theta}}$ as a log-linear grammar. Unless $\beta = 1$, the weights $\beta\vec{\theta}$ do not obey the constraints in Equations 2.3 and do not define multinomial distributions. If $\beta > 0$, which it is, by design), however, the parameters $\beta\vec{\theta}$ *do* define a log-linear model. The model may diverge (Section 3.5.1), but that does not matter. A valid *posterior* $q(\mathbf{Y} \mid \mathbf{X})$ is well-defined (Section 2.4), and the Inside-Outside algorithm will produce the required quantities (sufficient statistics under $q$).

Smith and Eisner (2004) applied DA with some success to a part-of-speech tagging task (from unannotated examples, with a known tagging dictionary, following experiments described by Merialdo, 1994) and to the unsupervised estimation of Klein and Manning's (2001a; 2002a) constituent-context model for binary unlabeled constituent structure. Smith and Eisner improved the part-of-speech tagging accuracy on ambiguous words in test data from EM's 66.6% to DA's 70.2% and, on constituent parsing, improved the unlabeled PARSEVAL $F_1$ score (Black *et al.*, 1991) by one point over EM (67% to 68%). These improvements, while small, were statistically significant.

If $\beta$ is increased past 1, toward $+\infty$, the objective function approaches Viterbi EM. Recall that Viterbi EM places all of $q$'s mass on the most probable analysis; see Section 3.3.4. It is left as an algebraic exercise to see that this is the case (see Equation 3.16).

### 5.1.3 Annealing Schedule

Following the algorithm in Equation 5.1, we do not gradually change $\beta$ during learning; it is a fixed value on any given EM iteration within DA. We must specify a *schedule* for changing $\beta$ over time. Here we start with a value $\beta_0$ and fix a growth constant $\gamma > 1$. Keeping $\beta = \beta_0$, we train the model using EM iterations with the modified E step. Upon convergence, we set $\beta \leftarrow \min(1, \gamma\beta)$. This continues until EM iterations converge at $\beta = 1$. The smaller we make $\beta_0$ and $\gamma$, the slower annealing will be: training will take longer, but we can have more confidence (though never a guarantee) in finding a taller mountaintop on the likelihood function.

### 5.1.4 A Note on DA

Readers familiar with DA as an alternative to stochastic gradient descent—that is, as a method for optimizing discontinuous functions—may be puzzled that there is only one additional parameter involved, $\beta$. For example, Rao and Rose (2001) and D. Smith and Eisner (2006a) describe their methods in terms of two parameters, temperature and another scale parameter. Their application of DA gradually *lowers* a "temperature" parameter, $\tau$ (we use our notation, not theirs, for within-thesis clarity):

$$\min_{\vec{\theta},\beta} \mathbf{E}_{p_{\beta\vec{\theta}}(\mathbf{W})}\left[\bullet\right] - \tau \cdot \mathbf{H}\left(p_{\beta\vec{\theta}}(\mathbf{W})\right) \tag{5.12}$$

where $\bullet$ suppresses a more complex function of modeled structures $\mathbf{w} \in \mathcal{W}$. In their setting, $\beta$ is a free parameter to be optimized over.

The difference between the present work and those papers is that the EM objective, when written in Neal and Hinton's form, *already* includes the (expected) entropy over $q$ (the second term). The functions optimized in the other papers are not based on likelihood and do not include entropy. Informally put, they add an entropy term to their objective functions, scaled by $\tau$. Eventually the entropy term vanishes (as $\tau \rightarrow 0$). That scalar corresponds to (the inverse of) our $\beta$.

For us, $\tau \overset{\text{def}}{=} 1/\beta$, and the scaling (by $\beta$, equivalently $1/\tau$) of the parameters $\vec{\theta}$ simply falls out of the derivation. Also, we drive $\beta$ to 1, since the entropy term is part of the original function and we do not want it to vanish.

## 5.2 Experiments

We compare deterministic annealing EM with regular EM (as in Chapter 3). The same three initializers were used, and we tested various annealing schedules. Results are presented in Table 5.1.

The first thing to notice is that the initializer has virtually no effect on the model learned. This is exactly as we expect—early rounds of DA correspond to a concave function with one global maximum that is quickly discovered (assuming that $\beta_0$ is close enough to zero), so it shouldn't matter where $\vec{\theta}^{(0)}$ is. Another way to think about it: when $\beta \approx 0$ (the beginning of learning), the posterior distribution $q$ is forced to have high entropy (like the Zero initializer), and the original parameters are quickly forgotten. Next, notice that

| | | | Zero | | | | K&M | | | | Local | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | test accuracy | | | | test accuracy | | | | test accuracy | | |
| | | | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected |
| EM | | | 49 | 26.07 | 22.7 | 58.8 | 62 | 25.16 | 41.7 | 62.1 | 49 | 26.07 | 22.8 | 58.9 |
| DA | $\beta_0 =$ | $\gamma =$ | | | | | | | | | | | | |
| | 0.1 | 2.5 | 57 | 14.25 | 27.5 | 60.1 | 57 | 14.25 | 27.8 | 60.2 | 56 | 14.25 | 27.2 | 59.8 |
| | 0.1 | 2.25 | 23 | 9.51 | 25.9 | 55.1 | 23 | 9.51 | 25.9 | 55.1 | 22 | 9.51 | 25.6 | 54.7 |
| | 0.1 | 2.0 | 68 | 20.21 | 29.5 | **64.0** | 68 | 20.21 | 29.5 | **63.8** | 67 | 20.21 | 29.6 | **64.0** |
| | 0.1 | 1.75 | 93 | 24.30 | 21.2 | 60.1 | 93 | 24.30 | 21.3 | 60.1 | 92 | 24.30 | 21.3 | 60.1 |
| | 0.1 | 1.5 | 95 | 18.94 | 27.5 | 62.6 | 95 | 18.94 | 27.5 | 62.6 | 94 | 18.94 | 27.5 | 62.6 |
| | 0.1 | 1.25 | 165 | 24.00 | 33.8 | **65.4** | 165 | 24.00 | 34.0 | **65.6** | 164 | 24.00 | 34.3 | **66.0** |
| | 0.01 | 2.5 | 73 | 25.42 | 22.5 | 58.8 | 73 | 25.42 | 22.5 | 58.7 | 73 | 25.42 | 22.6 | 58.9 |
| | 0.01 | 2.25 | 61 | 12.37 | 26.7 | 56.8 | 61 | 12.37 | 27.2 | 57.3 | 61 | 12.37 | 27.1 | 57.3 |
| | 0.01 | 2.0 | 81 | 14.81 | 26.3 | 60.2 | 81 | 14.81 | 26.5 | 60.2 | 81 | 14.81 | 26.2 | 60.1 |
| | 0.01 | 1.75 | 118 | 22.63 | 21.7 | 59.8 | 118 | 22.63 | 21.7 | 59.9 | 118 | 22.63 | 21.7 | 59.8 |
| | 0.01 | 1.5 | 134 | 22.12 | 34.8 | **66.6** | 134 | 22.12 | 34.1 | **65.9** | 134 | 22.12 | 34.4 | **66.5** |
| | 0.01 | 1.25 | 199 | 22.19 | 33.4 | **65.2** | 199 | 22.19 | 34.4 | **66.3** | 199 | 22.19 | 34.2 | **66.0** |
| | 0.001 | 2.5 | 69 | 13.69 | 27.4 | 58.5 | 69 | 13.69 | 28.0 | 58.9 | 69 | 13.69 | 27.9 | 58.6 |
| | 0.001 | 2.25 | 76 | 15.44 | 24.8 | 59.3 | 76 | 15.44 | 24.9 | 59.5 | 76 | 15.44 | 24.6 | 59.2 |
| | 0.001 | 2.0 | 45 | 9.74 | 25.7 | 55.0 | 45 | 9.74 | 25.8 | 55.1 | 45 | 9.74 | 25.7 | 55.0 |
| | 0.001 | 1.75 | 104 | 20.97 | 30.2 | **64.1** | 104 | 20.97 | 30.3 | **64.2** | 104 | 20.97 | 30.3 | **64.3** |
| | 0.001 | 1.5 | 167 | 25.51 | 28.2 | **64.2** | 167 | 25.51 | 28.2 | **64.2** | 167 | 25.51 | 28.2 | **64.2** |
| | 0.001 | 1.25 | 222 | 20.43 | 32.2 | **65.1** | 222 | 20.43 | 32.1 | **64.9** | 222 | 20.43 | 32.1 | **65.0** |
| unsupervised model selection: $\vec{\theta}^{(0)} =$ Zero, $\beta_0 = 0.1$, $\gamma = 2.25$ | | | | | | | | | | | 23 | 9.51 | 25.9 | 55.1 |
| supervised model selection: $\vec{\theta}^{(0)} =$ Zero, $\beta_0 = 0.01$, $\gamma = 1.5$ | | | | | | | | | | | 134 | 22.12 | 34.8 | 66.6 |

Table 5.1: MLE training using EM and DA with different annealing schedules and initializers. Model selection was applied across the initializer $\vec{\theta}^{(0)}$ and the annealing schedule, $\langle \beta_0, \gamma \rangle$. Boldface marks trials significantly more accurate (sign test, $p < 0.05$) than the EM/K&M trial.

| | insensitive to $\vec{\theta}^{(0)}$? | tries to avoid local optima? | unconstrained (cf. Eq. 2.3)? | objective function |
|---|---|---|---|---|
| EM (Section 3.3) | no | no | no | likelihood (also with Dirichlet prior) |
| LMVM (Section 4.6) | no | no | yes | any differentiable, unconstrained function (used for contrastive estimation) |
| DA (Section 5.1) | yes | yes | no | likelihood (also with Dirichlet prior) |
| SDA (Section 5.3) | no | yes | no | likelihood (also with Dirichlet prior) |

Table 5.2: Summary of optimization methods used in the thesis for unsupervised learning of weighted grammars. Note that *estimation* criteria (objective functions) and optimization algorithms are loosely coupled, but the choice of one does not fully imply the choice of the other.

DA usually does find a better local minimum of $\mathbf{H}_C$ (equivalent to a local maximum of likelihood) than EM, often much better. Unfortunately, this does not necessarily correspond to better directed or undirected accuracy. So even though DA is a better optimizer of likelihood than EM, the likelihood-accuracy disconnect remains. There is a visible trend toward higher undirected accuracy when the objective function is changed more gradually (slower annealing; $\gamma$ closer to 1).

Further, it continues to be evident that there are many local optima, since each annealing schedule found a different one. Which one is found depends very much on the annealing schedule, and it is not necessarily the case that annealing more slowly (smaller $\beta_0$ and $\gamma$) finds a better local optimum (lower value of $\mathbf{H}_C$).

Finally, note that while the number of EM iterations required does increase with the number of epochs,[4] each epoch is on average much shorter than a full run of EM (which takes 50–60 iterations). This is because each EM loop is initialized by the previous one, which has already found a local optimum of a very similar function.

[4]The number of epochs is equal to $\lceil -\log_\gamma \beta_0 \rceil$. This ranges from 3 to 31 in the experiments in Table 5.1.

## 5.3 Skewed Deterministic Annealing

We have seen how deterministic annealing can outperform EM at optimizing likelihood—through the avoidance of shallow local optima—and break dependence of performance on the initializer. The former is desirable insofar as the likelihood function approximates directed accuracy. However, we have seen that the models learned with DA are not more accurate than those learned with EM. A good initializer, as we have seen, can be very important for successful learning. **Skewed deterministic annealing** (SDA) is a generalization of DA that tries to avoid local optima without sacrificing the benefits of a good initializer (Smith and Eisner, 2004). SDA is a "cautious" alternative to EM and a biased alternative to DA.

To introduce SDA, we consider the following restatement of DA's E step (given in Figure 5.1):

> Let $q(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \propto p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}})^{\beta} \cdot \left(\frac{1}{|\mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}|}\right)^{1-\beta}$ for each observed $\mathbf{x}^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$ and each possible value $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}$.

The term $\left(\frac{1}{|\mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}|}\right)^{1-\beta}$ is of course a constant with respect to the parameters. We write the E step this way to show that the posterior under DA can be described as an interpolation (in the log domain and with coefficient $\beta$) between the model's posterior $p_{\vec{\theta}^{(i)}}$ and the uniform posterior (the distribution with the highest entropy).

SDA replaces the uniform posterior with an arbitrary one, which we will refer to as $\acute{p}(\mathbf{y} \mid \mathbf{x})$:

> Let $q(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}}) \propto p_{\vec{\theta}^{(i)}}(\mathbf{y} \mid \mathbf{x}^{\mathrm{t}})^{\beta} \cdot \acute{p}(\mathbf{y} \mid \mathbf{x}_t)^{1-\beta}$ for each observed $\mathbf{x}^{\mathrm{t}}$ in $\vec{\mathbf{x}}^{\mathrm{t}}$ and each possible value $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}^{\mathrm{t}}}$.

The SDA objective function is, then:

$$\begin{aligned} f_{\mathrm{SDA}}(\vec{\theta}, q, \beta, \acute{p}) &= \frac{1}{\beta}\mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\mathbf{H}\left(q(\mathbf{Y} \mid \mathbf{X})\right)\right] + \mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})\right] \\ &\quad + \mathbf{E}_{\tilde{p}(\mathbf{X})}\left[\log p_{\vec{\theta}}(\mathbf{X})\right] + \frac{1-\beta}{\beta}\mathbf{E}_{\tilde{p}(\mathbf{X})\cdot q(\mathbf{Y}|\mathbf{X})}\left[\log \acute{p}(\mathbf{Y} \mid \mathbf{X})\right] \end{aligned} \quad (5.13)$$

When $\beta$ is close to zero, SDA behaves cautiously, not wanting the model's posterior over hidden structures to diverge too far from $\acute{p}$. As $\beta$ increases toward one, the model is allowed to move up a nearby likelihood hill, but only if the hill is steep enough to overcome the SDA-imposed attraction toward the $\acute{p}$-friendly model.

In this work we do not represent $\acute{p}$ directly, instead using $\vec{\theta}^{(0)}$. To carry out the SDA E step, we replace $\vec{\theta}$ with $\beta\vec{\theta} + (1 - \beta)\vec{\theta}^{(0)}$, which has exactly the same effect as interpolating between the posteriors. This is very similar to DA (Figure 5.1), which simply *scaled* $\vec{\theta}$ by $\beta$.

To our knowledge, SDA is a novel generalization of DA. It is applicable to any problem where DA is applicable, and is likely to be helpful in cases wherever a good initializer is known.

### 5.3.1 Experiments

We compare skewed deterministic annealing EM with regular EM. The same three initializers were used, and we tested various annealing schedules (the same ones tested with DA). Results are presented in Table 5.3.

Unlike with DA, we see a strong dependence on the initializer, as expected. Note that the Zero initializer is essentially the same under DA (Table 5.1) and SDA; this is because the Zero initializer gives a uniform posterior over hidden structures. It is the model toward which DA tends by default (the maximum entropy model).

With the K&M and Local initializers, SDA does not typically find a better local optimum than EM using the same initializer, but it does often find a more *accurate* model. This adds to the body of damning evidence against likelihood (see Chapter 3): an alternative search strategy (SDA) performed worse at optimizing likelihood, but serendipitously found a more accurate model. Notably, when $\gamma$ is not too large, undirected accuracy typically improves over the best EM/MLE condition. Under these initializers, directed accuracy is rarely much worse under SDA than EM with the same initializer (often it is better).

Finally, SDA does not generally require more iterations than EM training to improve performance over EM. As with DA, each epoch requires only a few iterations to converge, since the previous epoch involved optimizing a similar function, and the local optimum of the new function is unlikely to be far from the local optimum of the previous one.[5]

We conclude that SDA is a useful alternative to EM that carries out a more cautious search of the likelihood surface. It does not, in general, find a better local optimum than EM, but given a good initializer it can improve accuracy. We offer the following explana-

[5]Indeed, SDA nearly always found a local optimum closer to the initial parameters $\vec{\theta}^{(0)}$ than EM did.

| | | | Zero | | | | K&M | | | | Local | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | test accuracy | | | | test accuracy | | | | test accuracy | |
| | | | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected |
| EM | | | 49 | 26.07 | 22.7 | 58.8 | 62 | 25.16 | 41.7 | 62.1 | 49 | 26.07 | 22.8 | 58.9 |
| SDA | $\beta_0 =$ | $\gamma =$ | | | | | | | | | | | | |
| | 0.1 | 2.5 | 57 | 14.25 | 27.5 | 60.2 | 38 | 29.62 | **44.6** | 61.2 | 36 | 29.54 | 25.8 | 59.1 |
| | 0.1 | 2.25 | 23 | 9.51 | 25.9 | 55.2 | 27 | 30.23 | 40.7 | 57.4 | 27 | 29.96 | 27.1 | 56.6 |
| | 0.1 | 2.0 | 68 | 20.21 | 29.5 | **64.0** | 52 | 27.92 | **46.7** | **64.3** | 77 | 28.25 | 30.5 | **63.9** |
| | 0.1 | 1.75 | 93 | 24.30 | 21.3 | 60.2 | 84 | 26.09 | **42.2** | **62.6** | 95 | 26.65 | 30.8 | **66.0** |
| | 0.1 | 1.5 | 95 | 18.94 | 27.5 | 62.5 | 66 | 28.38 | **45.4** | 62.7 | 66 | 28.62 | 29.1 | **63.0** |
| | 0.1 | 1.25 | 165 | 24.00 | 34.2 | **65.7** | 136 | 26.18 | **42.4** | **62.7** | 139 | 26.77 | 34.2 | **65.8** |
| | 0.01 | 2.5 | 73 | 25.42 | 22.5 | 58.9 | 73 | 25.51 | 41.7 | 62.0 | 71 | 26.34 | 22.1 | 59.4 |
| | 0.01 | 2.25 | 61 | 12.37 | 27.1 | 57.3 | 47 | 29.94 | 43.1 | 60.0 | 42 | 29.75 | 27.2 | 58.4 |
| | 0.01 | 2.0 | 81 | 14.81 | 26.1 | 60.0 | 50 | 29.51 | **44.1** | 61.5 | 50 | 29.46 | 25.9 | 59.8 |
| | 0.01 | 1.75 | 118 | 22.63 | 21.7 | 59.9 | 84 | 26.90 | 41.9 | 62.3 | 98 | 27.41 | 33.3 | **65.8** |
| | 0.01 | 1.5 | 134 | 22.12 | 34.1 | **66.1** | 105 | 27.09 | 42.1 | 62.4 | 106 | 27.57 | 33.3 | **65.0** |
| | 0.01 | 1.25 | 199 | 22.19 | 34.4 | **66.2** | 152 | 27.06 | 42.0 | 62.3 | 153 | 27.54 | 34.1 | **65.5** |
| | 0.001 | 2.5 | 69 | 13.69 | 27.5 | 58.6 | 54 | 29.73 | **44.3** | 61.3 | 50 | 29.61 | 26.4 | 58.7 |
| | 0.001 | 2.25 | 76 | 15.44 | 24.8 | 59.3 | 55 | 29.37 | **44.2** | 61.6 | 58 | 29.36 | 26.5 | 60.6 |
| | 0.001 | 2.0 | 45 | 9.74 | 26.0 | 55.4 | 49 | 30.21 | 41.3 | 57.9 | 49 | 29.94 | 26.9 | 56.7 |
| | 0.001 | 1.75 | 104 | 20.97 | 30.3 | **64.4** | 98 | 27.61 | 41.5 | 62.0 | 105 | 28.00 | 32.5 | **64.4** |
| | 0.001 | 1.5 | 167 | 25.51 | 28.2 | **64.2** | 141 | 25.38 | **42.2** | 62.4 | 155 | 26.08 | 32.9 | **65.2** |
| | 0.001 | 1.25 | 222 | 20.43 | 32.1 | **64.9** | 188 | 27.82 | 41.7 | 62.3 | 189 | 28.17 | 32.2 | **64.4** |
| unsupervised model selection: $\vec{\theta}^{(0)}$ = Zero, $\beta_0 = 0.1, \gamma = 2.25$ | | | | | | | | | | | 23 | 9.51 | 25.9 | 55.2 |
| supervised model selection: $\vec{\theta}^{(0)}$ = K&M, $\beta_0 = 0.1, \gamma = 2$ | | | | | | | | | | | 52 | 27.92 | 46.7 | 64.3 |

Table 5.3: MLE training using EM and SDA with different annealing schedules and initializers. Model selection was applied across the initializer $\vec{\theta}^{(0)}$ and the annealing schedule $\langle \beta_0, \gamma \rangle$. Boldface marks trials that achieved significantly better accuracy (sign test, $p < 0.05$) than the EM/K&M trial.

tion: likelihood is not a perfect approximation to accuracy, but in good regions of parameter space, improving likelihood often improves accuracy. Some initializers (like K&M, for this dataset) get the learner into a "good region." Different search algorithms that stay within that region (like EM and SDA, but not DA) will lead to different local optima given the same initializer. Some of those local optima are more accurate than others.

### 5.3.2 Further Analysis (Random Initializers)

Figure 5.2 shows the spread of accuracy performance of 100 random models (the same ones used in previous chapters) before training and after training initialized from each one, with EM and with SDA ($\beta = 0.01, \gamma = 1.5$). We consider here only one annealing schedule.

Figure 5.3 further illustrates that SDA, like EM, may select any of similar likelihood--valued local optima: 100 random initializers all find different local maxima with about the same function value and very different accuracies on test data. In fact, when we compare to Figure 3.7 on page 51, we see that SDA consistently finds local maxima that are *worse* in likelihood than EM finds (the exception is the Zero initializer). SDA is certainly not accomplishing anything like global optimization. This is consistent with the results in the last section. Like EM (Figure 3.8 on page 52), SDA does consistently improve the accuracy of the initializer (see Figure 5.4), though the improvement is more often than not less than improvement by EM for random models (see Figure 5.5). The result could turn out differently for a different annealing schedule.

For more comparison among methods on random initializers, see Sections 4.6.5, 6.1.4, 6.2.2, and 7.4.

### 5.3.3 Semisupervised Use of SDA

In Sections 3.4.2 and 4.6.6, we considered different uses of the annotated development data, most notably as an *initializer* to unsupervised training. Supervised initialization, we have seen, leads to higher accuracy than simply using the annotated data to choose among unsupervised models, but it is not as accurate as those supervised initializers themselves (except when very small amounts of annotated data are available).

SDA is a method designed to make the most of a good initializer. When we use the supervised model (trained on annotated development data) to initialize SDA training, we obtain considerable improvements; see Figure 5.6. Here we have used a fast growth rate

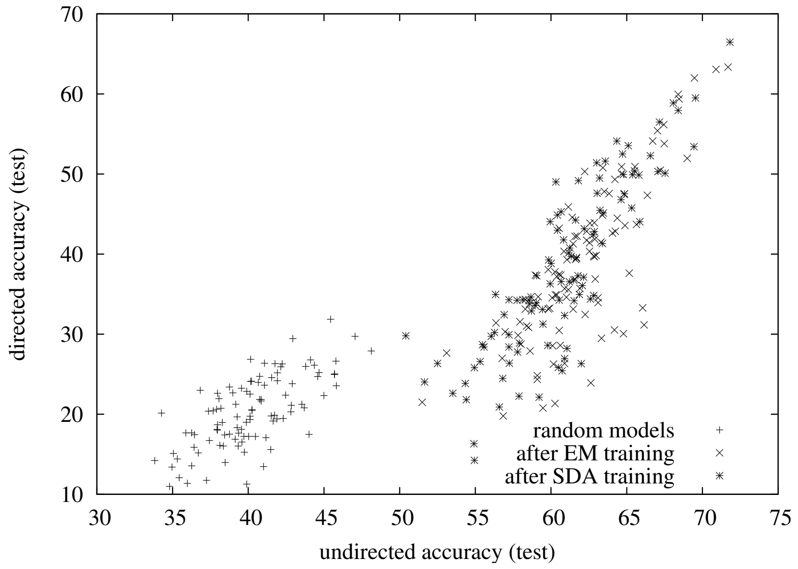


Figure 5.2: Accuracy on test data (directed *vs.* undirected): 100 random models before training, after EM training, and after SDA training with $\beta = 0.01, \gamma = 1.5$.

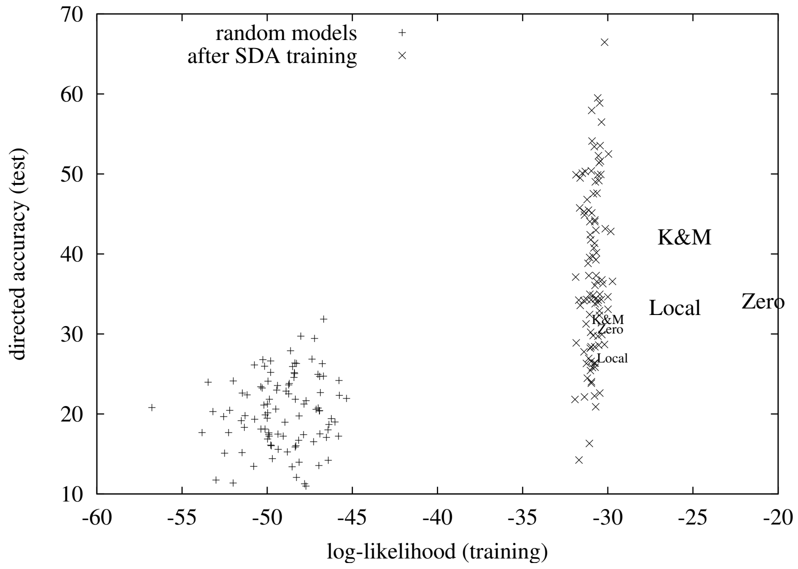


Figure 5.3: Accuracy on test data *vs.* log-likelihood on training data: 100 random models (and our three initializers) before and after SDA training ($\beta = 0.01, \gamma = 1.5$). Small labels correspond to our initializers before training, large labels to after training. Compare with Figure 3.7 on page 51.

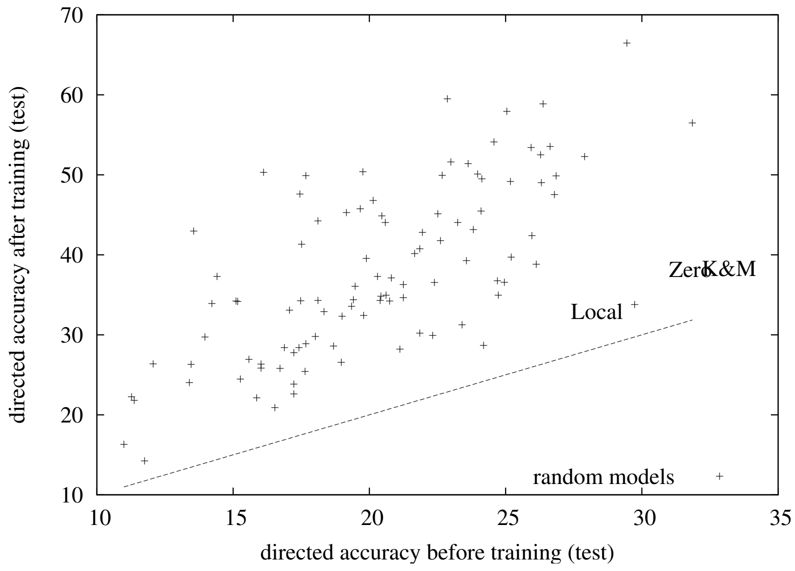


Figure 5.4: Directed accuracy on test data: 100 models after SDA training *vs.* before training. The text labels correspond to the more carefully designed initializers. The line is $y = x$. Compare with Figure 3.8 on page 52.

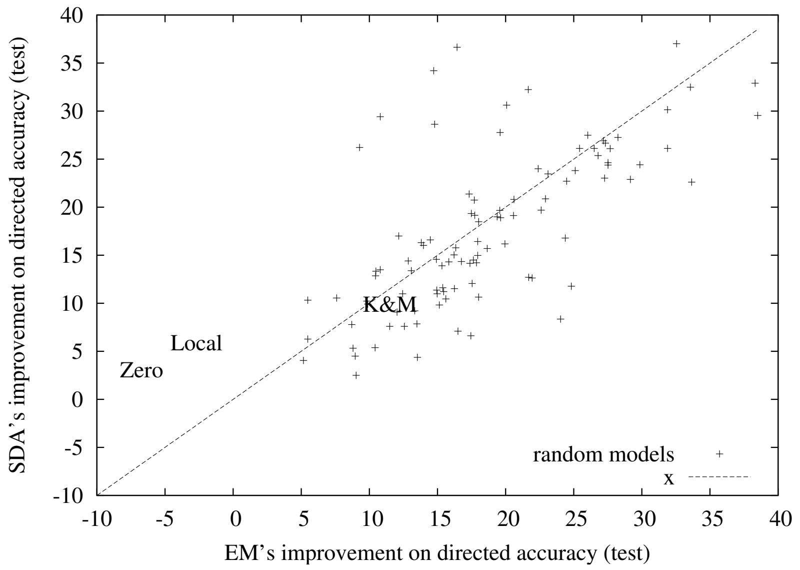


Figure 5.5: Absolute improvement (or reduction) of directed accuracy on test data via SDA verses EM. No smoothing was applied and the annealing schedule was fixed at $\beta = 0.01, \gamma = 1.5$. SDA makes better use of good initializers Local and Zero than EM, but EM tends to give more improvement over random initializers. The line is $y = x$.

($\gamma = 2.5$) and tried different initial $\beta_0$ values. For up to 50 annotated examples, SDA with supervised initialization is the best method, actually improving over supervised training alone. (Not shown in the figure, is supervised initialization *and* semisupervised SDA where the supervised data affect the counts after each E step. This performed virtually the same as mere supervised initialization and so is suppressed for clarity in the plot.) This does not extend to larger annotated datasets, though; the matter warrants further investigation.[6] Other values of $\gamma$ and $\beta_0$ (not shown) performed within about five points of these settings, occasionally slightly better. Note that unsupervised training by MLE/SDA with supervised selection of the annealing schedule and initializer is fairly stable as the development dataset diminishes, and performance is consistently higher than MAP/EM.

We have seen repeatedly that supervised initialization outperforms supervised model selection (see Section 3.4.2 for a defense of supervised model selection). The experimental results in this section are promising, suggesting that annealing "away" the initializer is a reasonable approach to combining labeled and unlabeled data. In future work we might consider different annealing schedules that do not stop at $\beta = 1$, but rather sooner, so that the annotated data influence the objective more strongly. Indeed, the development data could also be used to decide when to stop increasing $\beta$.

## 5.4 Related Work

We discuss an important feature of DA not capitalized on in this thesis: phase transitions in hidden variable cardinality. We then describe three general latent-variable modeling techniques that are similar in spirit to DA: example reweighting, latent maximum entropy, and the information bottleneck.

### 5.4.1 DA with Phase Transitions

An important application of deterministic annealing in natural language processing was by Pereira, Tishby, and Lee (1993). Their aim was to use distributional clustering to find words that were semantically related to each other. The central intuition was that two nouns that tend to appear as objects of the same verbs (more precisely, in the same

[6] Prior work, including Merialdo (1994) and Elworthy (1994) has found that combining nontrivial amounts of labeled data with unlabeled data in a maximum likelihood setting tends to learn worse models than MLE training on the labeled data alone. Our results are consistent with those findings, but SDA appears to push the "trivial" threshold higher than EM.

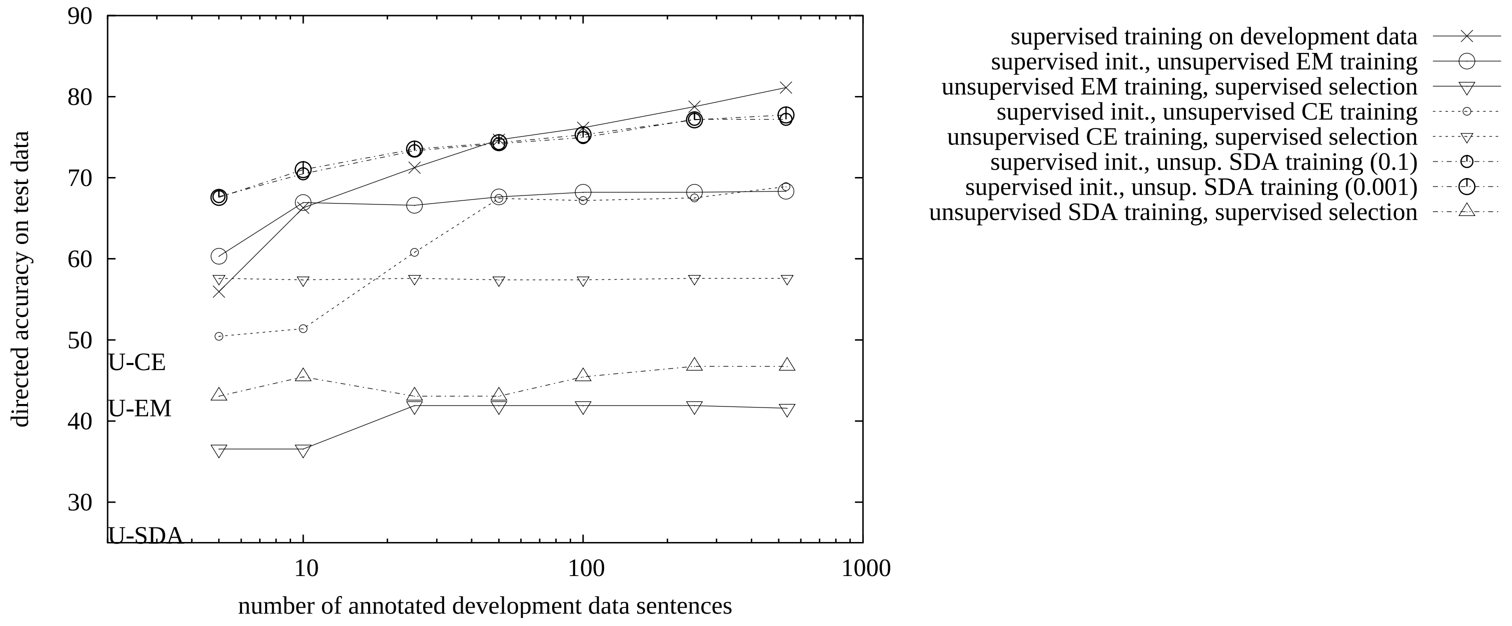


Figure 5.6: Different uses of the development dataset, as the size of the development dataset changes. The plot compares our standard setting of unsupervised estimation/supervised selection (with MAP/EM and MLE/SDA) against the use of the annotated development dataset to construct an *initializer* for MLE/EM and for MLE/SDA (with $\gamma = 2.5$; the parenthesized values refer to $\beta_0$). Supervised initialization with *semi*supervised SDA is not shown; it was indistinguishable from supervised initialization and unsupervised SDA training. CE is also shown; those trials are explained in more detail in Section 4.6.6. The "U" labels at left correspond to unsupervised model selection. When the initializer is not named, it has been selected over.

proportions for those verbs) are likely to have related meanings. The distance between these distributions over contexts was measured by KL divergence (Kullback and Leibler, 1951), and deterministic annealing was used in the construction of the clusters. This is a classic application of DA to clustering, and it highlights nicely one of the properties of DA that we have not touched upon here: phase transitions, or natural increases in hidden-variable cardinality that arise during DA learning. These phase transitions were central to the original exposition of DA as a vector quantization method (Rose *et al.*, 1990).

Note that in this probabilistic framework, each noun belongs to each cluster with some probability. Early in learning, when the entropy over cluster memberships for the nouns is high, all of the nouns belong more or less equally to all of the clusters; the clusters are effectively the same. Put another way, there is only one "effective cluster." Because of this, Pereira *et al.* need only define two clusters to start out.

Over time, as $\beta$ is increased, the entropy constraint is, as an effect, relaxed. This allows the two clusters to move apart from each other. When they move sufficiently far "apart" (measured by the divergence between their distributions over nouns), they are said to "split." When this happens, each cluster $\mathbf{c}$ gets a new twin which is a perturbed version of $\mathbf{c}$. The process repeats, with the cluster splits happening naturally as $\beta$ increases. The result is a hierarchical, soft clustering, if we keep track of the clusters before they split and their phylogenetic relationships. These phase transitions also serve the purpose of guiding the annealing schedule; the goal of the annealing process is to manipulate the temperature so as to find the phase transitions, and it can be sped up or slowed down—even reversed—to do so.

(Clustering in language learning has been very widely applied to words (Brill, 1991; Finch and Chater, 1992a,b; Brown, Della Pietra, de Souza, Lai, and Mercer, 1992; Schütze, 1993). More recently, Buchholz (1998) clustered verbs, and Rooth, Riezler, Prescher, Carroll, and Beil (1999) clustered words to discover subcategorization frames. Klein and Manning (2001b) and Clark (2001) have argued for the use of contextual distributions of possible constituents in inducing constituent grammars, treating grammar induction like a sequence-clustering problem. Both Klein and Manning and Clark applied distributional clustering, in some form, to discriminate sequences of part-of-speech tags that are constituents from those that are not.)

### 5.4.2 Example Reweighting

Elidan, Ninio, Friedman, and Schuurmans (2002) described a set of methods for avoiding local optima of bumpy objective functions by randomly perturbing the weights of observed examples. Reweighting the training data changes the empirical distribution $\tilde{p}$ and so changes the objective function, but only temporarily. Elidan *et al.* describe a random reweighting strategy and also an adversarial one that "challenges" the current hypothesis by updating the weights in the negative gradient direction. This technique is similar in spirit to simulated annealing (Kirkpatrick *et al.*, 1983). The perturbation of the training example weights is, in fact, annealed over time so that ultimately the examples are equally weighted.

### 5.4.3 Latent Maximum Entropy

Latent maximum entropy (LME; Wang, Rosenfeld, Zhao, and Schuurmans, 2002) is a generalization of the maximum entropy principle to models with latent variables. The maximum entropy principle (Jaynes, 1957) is, informally, the idea that, among models that fit the data, the one chosen should be the one that maximizes the entropy of the model distribution. A fascinating result is that, if "fitting the data" is taken to mean that, for a set of functions $f_i : \mathcal{W} \rightarrow \mathbb{R}_{\geq 0}$, the model expectations of those functions are equal to the empirical expectations, then the solution to the maximum entropy problem is exactly the maximum likelihood log-linear model (footnote 20 in Chapter 3, page 65). See Ratnaparkhi (1997) for a presentation of this property. It is for this reason that log-linear models are often called "maximum entropy" models.

The above result applies when dealing with fully observed data. Wang *et al.* (2002) pointed out that the relationship between MLE and maximum entropy does not hold for partially-observed data. They show that, for log-linear models, local maxima of the likelihood function are *feasible* solutions to the latent maximum entropy problem, and suggest finding many such estimates (using, for example, EM with multiple starting points) and selecting the model with the highest entropy over $\mathcal{W}$.[7]

[7]The entropy of the distribution of derivations defined by an SCFG can be computed by solving a system of linear equations (Grenander, 1967; Soule, 1974). The number of equations and the number of variables are equal to the number of nonterminals.

### 5.4.4 Information Bottleneck

Information bottleneck (IB; Tishby, Pereira, and Bialek, 1999) is a technique that relates statistical modeling to information theory. The relevant application of this idea to EM, in particular, is the IB-EM algorithm (Elidan and Friedman, 2003, 2005). The idea is very similar in spirit to DA. Recall that DA notes a tradeoff within the EM objective function (Equation 5.10) between maximizing the entropy of the posterior distribution $q$ over hidden structures and fitting the model to the data. IB-EM manipulates instead a tradeoff between *minimizing* the information in the hidden variables about the identity of the training instances and *maximizing* the information in the hidden variables about the observed data. Elidan and Friedman applied the method to Bayesian networks. Like DA in this kind of setting, it has the ability to automatically select the cardinality of discrete hidden variables through phase transitions (see Section 5.4.1).

Two additional important contributions to this area of parameter estimation were made by Elidan and Friedman, and these are worthy of note for future work with the methods developed in this thesis. The first is the use of **continuation** methods (Allgower and Georg, 1990) to automatically select the change in the annealing parameter (for instance, $\beta$ in our DA and SDA notation). This is a generalization of the idea of interpolation between "easy" and "difficult" problems. The second is the application of **early stopping** for better generalization. That is, instead of driving $\beta$ to the point where the training objective exactly matches likelihood (for us, $\beta = 1$), stop early and return the local optimum of a different function.

## 5.5 Summary

We started out by describing deterministic annealing (DA) and how it is applied to improve search during MLE training using EM. While DA often does better at maximizing likelihood, it replaces dependence on the initial parameters (as in EM) with dependence on the choice of annealing schedule (though annealing very slowly is an obvious choice). For Model A, DA does not generally improve labeled accuracy, though it sometimes improves unlabeled accuracy. We presented *skewed* DA (SDA), which brings back dependence on the initializer but usually finds a slightly more accurate model than EM. It appears to be more sensitive to the annealing hyperparameters than DA, and converges in 2–3 times as

|  | test accuracy |  |
|---|---|---|
|  | directed | undirected |
| ATTACH-RIGHT | 39.5 | 62.1 |
| MLE/EM (s-sel.) | 41.7 | 62.1 |
| MAP/EM (s-sel.) | 41.6 | 62.2 |
| MLE/DA (s-sel.) | 34.8 | 66.6 |
| MLE/SDA (s-sel.) | 46.7 | 64.3 |

Table 5.4: Summary of baselines and results in this chapter.

many iterations as EM, at worst. While SDA found more accurate solutions than EM, it did *not* do so by improving likelihood. Having given likelihood another chance under some alternative optimization schemes (cf. EM), we conclude that optimizing likelihood is not a particularly effective way to learn linguistically motivated grammars. This conclusion is consistent with the last two chapters. We also found that SDA is a more effective semisupervised method than EM or CE, though (like EM and CE) it is not as effective as supervised training (ignoring the unannotated data) when enough annotated examples are available.

The idea of annealing, however, is very general. In the next chapter we will apply it in a different way that guides search by changing a structural bias in the model. Finally, we point out that DA and SDA could be applied to alternative objective functions (like contrastive estimation, Chapter 4), which we have not done here.

# Chapter 6

# Structural Annealing

*Progress imposes not only new possibilities for the future but new restrictions.*
—Norbert Weiner (1894–1964)

This chapter presents a novel approach to unsupervised parameter estimation called **structural annealing**. The idea is a blend between changing the objective function for improved accuracy (like contrastive estimation, Chapter 4) and annealing (changing the function over time, as in Chapter 5).

First we discuss an example of structural **bias** that is helpful for language learning: a preference for string-local attachments. We show how this can be injected into Model A (giving a new model, which we call Model L) through a single structural feature: the total length of dependencies in a tree. While this **locality** measure is a global feature of the tree, it factors into local terms (the total length of dependency edges equals the sum of edge lengths). Model L is shown to give substantial improvements over Model A on English data without drastically changing the estimation algorithm, when the single added parameter is chosen properly and fixed (left untrained).

**Annealing** this parameter results in a very strong locality preference early in learning that is gradually relaxed. This approach is called **structural annealing** and gives the best results on the task to date.

We then present another alternative model, Model S, which differs from Models A and L in its hypothesis space, allowing partial parses. Structural annealing on the weight of a single **connectedness** feature (counting the number of breaks in the tree) gives a similar trend in performance: Model S beats Model A, more so with structural annealing.

The methods and preliminary experimental results presented in this chapter were originally published in Smith and Eisner (2006).

## 6.1 Structural Bias: Locality

A widely noted fact about syntactic dependencies in natural language is that they tend to be very *string-local* (see, among many others, Lin, 1996). For example, in our English training corpus, 95% of dependency links cover four or fewer words. Figure 6.1 shows the distribution over string distance between parents and children in the training corpora used in the thesis (experiments on the other languages will be presented in Chapter 8). Across languages, we see that at least half of the dependencies in these corpora are between adjacent words. Put simply, most syntax is local, and we want to take advantage of this tendency during parameter estimation.

Model A does not have any kind of string-locality feature. In *supervised* parsing, many models have incorporated different kinds of features of the material between parent/child pairs, which can help capture this trend where it is empirically observed. Collins (1997a) considered three binary features of the intervening material between each parent and child node: did it contain (a) any word tokens at all, (b) any verbs, (c) any commas or colons? Note that (b) is effective because it measures the length of a dependency in terms of the number of alternative attachment sites that the dependent skipped over, a notion that could be generalized. Similarly, McDonald *et al.* (2005a) had a feature for the presence of each of the intervening POS tags. Klein and Manning (2003a) conditioned child generation probabilities on coarse distance classes, and Eisner and Smith (2005) generated string distances directly in a deficient model estimated by maximum likelihood, achieving speed and accuracy improvements.

In unsupervised learning, our goal is not to *model* the distance configurations as in the work cited above, but rather to *exploit* the linguistic preference for local attachment when estimating the model's parameters. Incorporating a locality preference into the model (through additional features) is one way to do this. A single feature that captures the intuition is the total string distance between all words and their parents (known as "structural complexity;" see Lin, 1996)

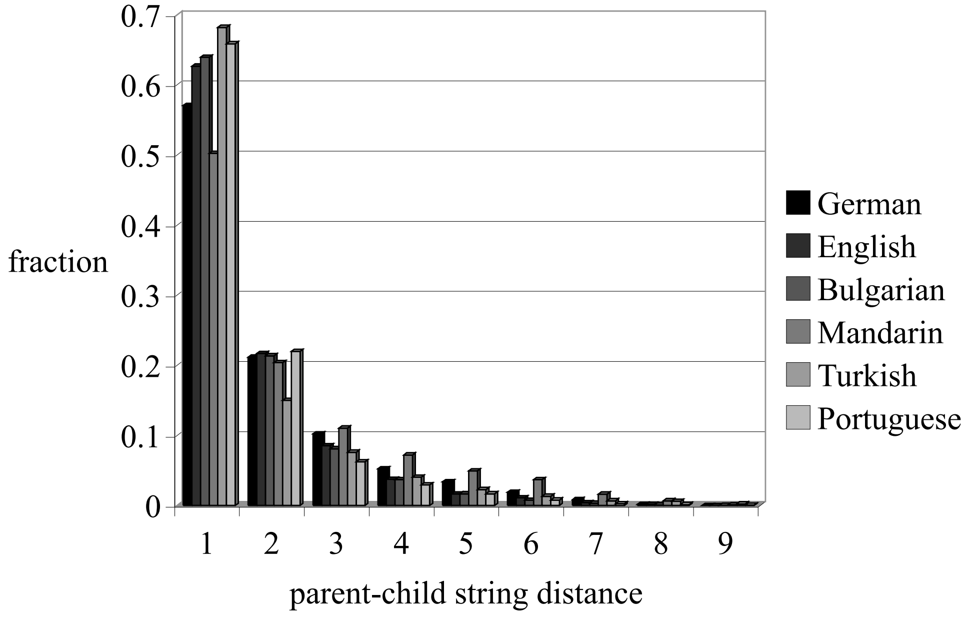


Figure 6.1: Distributions over parent-child string distances in training corpora in different languages. These training corpora contain only sentences of ten or fewer words, without punctuation.

$$f_{\mathbf{distance}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \sum_{i=1}^{|\mathbf{x}|} \sum_{j:j \in \mathbf{y}(i)} |i - j| \tag{6.1}$$

Our aim, however, is to posit the preference, not learn it (we already know about the preference—why try to learn what we already know?). Therefore, we will add the above feature to Model A and give it a weight, $\delta$, that is chosen before learning and *not trained*. The feature is used only during training; the end result is the same Model A, decoded as in previous chapters, but trained to optimize a different objective function: likelihood of a slightly different model.

### 6.1.1 Model L

We call that different model Model L. Model L is exactly like Model A, except for the incorporation of $f_{\mathbf{distance}}$ with untrained weight $\delta$:[1]

$$\ddot{p}^{\mathrm{L}}_{\vec{\theta},\delta}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \frac{\ddot{p}^{\mathrm{A}}_{\vec{\theta}}(\mathbf{x}, \mathbf{y}) \cdot e^{\delta \cdot f_{\mathbf{distance}(\mathbf{x},\mathbf{y})}}}{\ddot{Z}_{\vec{\theta},\delta}(\mathcal{W})} \tag{6.2}$$

If $\delta < 0$, Model L prefers local attachments; there is a **locality bias**. In the limit, as $\delta \to -\infty$, the model will tend toward an extremely strong preference for left- or right-branching trees. If $\delta = 0$, there is no bias and we have Model A. If $\delta > 0$, we have a bias *against* locality, preferring longer attachments. With finite $\delta$, the biased model does not eliminate any hypotheses from $\mathcal{Y}_{\mathbf{x}}$, it only changes their relative probabilities. Hence the language of the grammar underlying Model L (for a given $\Sigma$) is strongly equivalent to that underlying Model A, though Model L may represent distributions over trees that Model A cannot.

### 6.1.2 Training with Model L

Model L is a log-linear grammar, not a stochastic grammar. Therefore we would not expect to be able to train Model L with EM (see Section 3.5). This turns out not to be a difficulty because we do not directly train $\delta$.

Recall that during EM training (Figure 3.3), each E step sets $q$ to the conditional family $p_{\vec{\theta}}(\mathbf{Y} \mid \mathbf{X})$. For Model A this is done using an Inside-Outside algorithm (Section 2.4);

[1]This weight is in the log-domain, like $\vec{\theta}$.

an almost-identical Inside-Outside algorithm can be used for Model L (Section A.4). The M step carries out fully-observed MLE, using $q$ to fill in the distribution over the hidden variable. With $\delta$ fixed, and for a given $q$, this corresponds to the usual maximization of likelihood with respect to $\vec{\theta}$, carried out by normalizing sufficient statistics.

EM with Model L is essentially the same in practice as EM with Model A, except that the posterior $q$, computed on E steps, is now given by:

$$q_{\vec{\theta},\delta}(\mathbf{y} \mid \mathbf{x}) \quad \propto \quad p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) \cdot \exp\left(\delta \cdot f_{\mathbf{distance}}(\mathbf{x}, \mathbf{y})\right) \tag{6.3}$$

where $\propto$ means "proportional to" (equal up to a positive constant factor). Notice that, just as in DA (Chapter 5) and CE (Chapter 4), this isn't necessarily a proper distribution that sums up to one for each $\mathbf{x}$ (hence we use the $\propto$ symbol). As we have repeatedly mentioned, this does not pose a problem, since the E step for Model A (and other stochastic grammars) does not represent $q$ directly, only sufficient statistics under $q$ for use in the next M step. Correct computation of the sufficient statistics using the Inside-Outside algorithm does not require the computation of a normalizing term.

The dynamic programming equations for Model L are given in Section A.4. The essential change is to multiply in $e^{\delta|i-j|}$ every time an attachment is made between $x_i$ and $x_j$.

### 6.1.3 Experiments

Table 6.1 shows performance of EM with a locality bias $\delta$ set at different values. For this dataset, we see significant improvement over the baseline ($\delta = 0$), on both directed and undirected accuracy, when $\delta$ is set to a small negative number. Notably, the Zero initializer can match the performance of the K&M initializer if a good value of $\delta$ is used (in this case, $-0.4$).

Of course, adding a prior (smoothing) is expected to improve performance. We tested the same smoothing conditions as in Section 3.4.1, and applied model selection. Supervised selection across all three initializers, 11 values of $\delta$, and 7 values of $\lambda$ (241 conditions) chose Zero, $\delta = -0.6$, $\lambda = 10$, and achieved 61.8% directed accuracy and 69.4% undirected accuracy. Note that this is a more heavily smoothed model than EM's selected model—the bias and the prior cannot be chosen orthogonally. This is the best performance on the task achieved so far. *Unsupervised* model selection (choosing by likelihood, under Model A, on

| | Zero | | | | K&M | | | | Local | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | test accuracy | | | | test accuracy | | | | test accuracy | |
| $\delta$ | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected | iterations | $\mathbf{H}_C$ | directed | undirected |
| -1.0 | 37 | 25.74 | 37.6 | 58.8 | 34 | 25.74 | 37.6 | 58.8 | 32 | 25.74 | 31.3 | 56.4 |
| -0.8 | 31 | 25.65 | 38.2 | 59.1 | 28 | 25.65 | 38.2 | 59.1 | 52 | 25.65 | 30.3 | 56.1 |
| -0.6 | 59 | 25.58 | 37.6 | 59.1 | 54 | 25.58 | 37.4 | 59.0 | 41 | 25.58 | 30.2 | 56.1 |
| -0.4 | 41 | 25.38 | **45.5** | **63.2** | 43 | 25.52 | 38.6 | 59.0 | 54 | 26.09 | 29.3 | **65.0** |
| -0.2 | 57 | 25.93 | 35.3 | **62.8** | 41 | 25.29 | **45.3** | **62.9** | 48 | 25.96 | 29.4 | **66.0** |
| (≡Model A) 0.0 | 49 | 26.07 | 22.7 | 58.7 | 62 | 25.16 | 41.7 | 62.1 | 49 | 26.07 | 22.7 | 58.6 |
| 0.2 | 74 | 26.34 | 23.1 | 57.8 | 100 | 25.51 | 42.4 | 62.3 | 58 | 26.36 | 23.9 | 57.3 |
| 0.4 | 44 | 30.40 | 14.1 | 33.7 | 63 | 30.44 | 15.2 | 33.9 | 100 | 30.23 | 15.9 | 35.2 |
| 0.6 | 46 | 31.47 | 11.3 | 28.7 | 42 | 31.47 | 12.3 | 28.5 | 51 | 31.44 | 11.4 | 28.8 |
| 0.8 | 55 | 31.83 | 9.2 | 26.3 | 65 | 31.83 | 9.3 | 26.3 | 46 | 31.79 | 9.7 | 27.0 |
| 1.0 | 40 | 31.99 | 8.2 | 25.1 | 47 | 31.98 | 9.1 | 26.0 | 48 | 31.99 | 8.8 | 25.8 |
| unsupervised model selection: $\vec{\theta}^{(0)} = \text{K\&M}, \delta = 0$ | | | | | | | | | 62 | 25.16 | 41.7 | 62.1 |
| supervised model selection: $\vec{\theta}^{(0)} = \text{Zero}, \delta = -0.4$ | | | | | | | | | 41 | 25.38 | 45.5 | 63.2 |
| maximum *a posteriori* (Dirichlet prior): | | | | | | | | | | | | |
| unsupervised model selection: $\vec{\theta}^{(0)} = \text{K\&M}, \delta = 0, \lambda = 10^{-2/3}$ | | | | | | | | | 49 | 25.24 | 41.6 | 62.2 |
| supervised model selection: $\vec{\theta}^{(0)} = \text{Zero}, \delta = -0.6, \lambda = 10$ | | | | | | | | | 44 | 27.28 | 61.8 | 69.4 |

Table 6.1: Model L trained using EM different initializers and biases $\delta$. Note that $\delta = 0$ is essentially equivalent to MLE using EM on Model A (the scores differ slightly due to slight differences in implementation). No smoothing was done here, except for the last two lines (see text). $\mathbf{H}_C$ is cross-entropy on the training data *under Model A*—it is not the quantity being optimized. Boldface marks trials that achieved significantly better accuracy (sign test, $p < 0.05$) than the EM/K&M trial. The most accurate trial for any initializer always sets $\delta < 0$. For a graphical comparison against other methods over six languages, see Figure 8.2 (page 213) and Figure 8.3 (page 214).

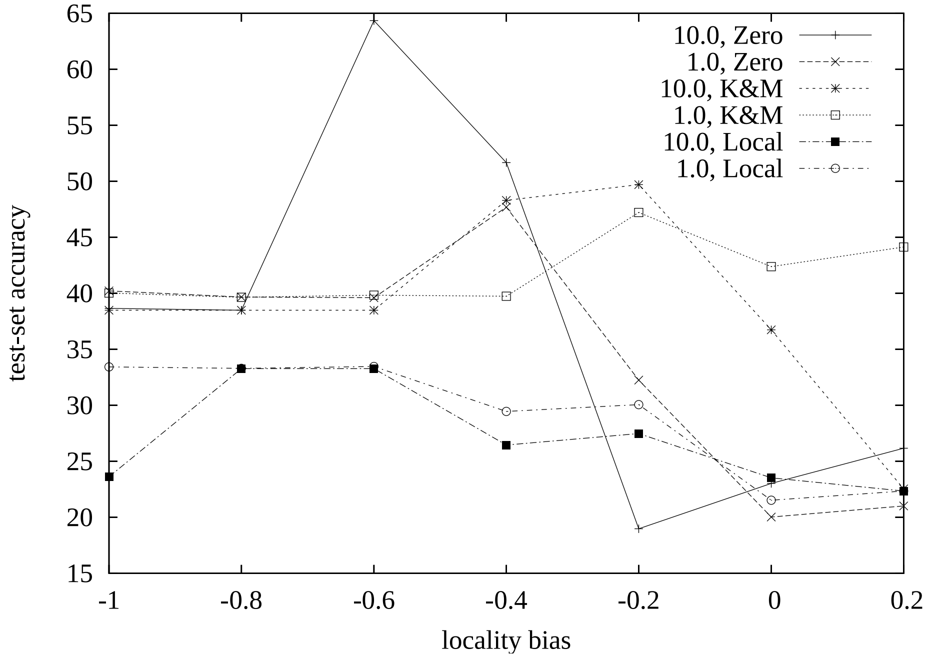


Figure 6.2: Test-set directed attachment accuracy of models at different values of $\delta$. Each curve corresponds to a different value of $\lambda$ and $\vec{\theta}^{(0)}$. Notice that the highest point on most curves is not at $\delta = 0$.

development data) across the same conditions chose the same model as in MAP/EM trials: K&M initializer, $\delta = 0$, $\lambda = 10^{-2/3}$. It achieves 41.6% (62.2% undirected). Figure 6.2 plots directed accuracy against $\delta$, for a few different initialization and smoothing conditions.

### 6.1.4 Further Analysis (Random Initializers)

Figure 6.3 shows the spread of accuracy performance of 100 random models (the same ones used in previous chapters) before training and after EM training initialized from each one with Model A and with Model L ($\delta = -0.4$). The outcomes are virtually indistinguishable.

Figure 6.4 further illustrates that the likelihood function for Model L suffers from local maxima, just like Model A. 100 random initializers all find different local maxima with about the same function value and very different accuracies on test data. However, as illustrated in Figure 6.5, EM/Model L training always results in a more accurate model than the initializing $\vec{\theta}$. This was not true in the case of EM/Model A (for the Local and Zero

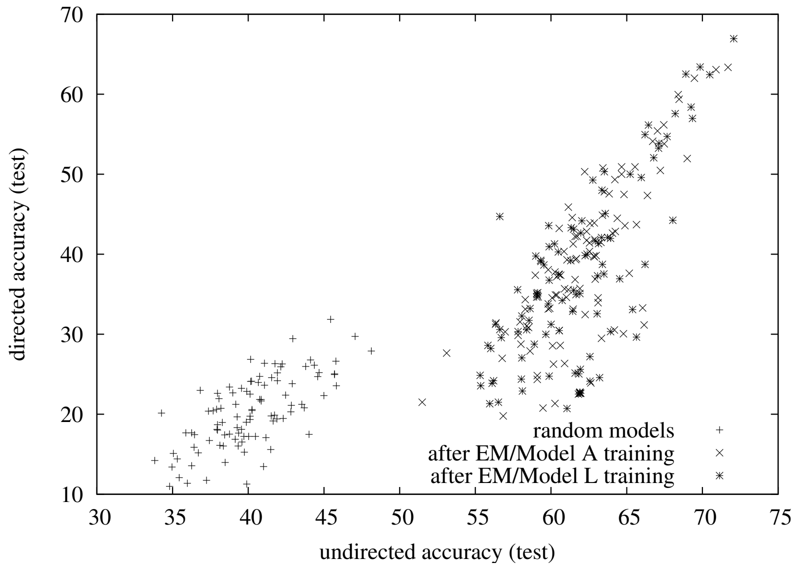


Figure 6.3: Accuracy on test data (directed *vs.* undirected): 100 random models before training, after EM training of Model A, and after EM training of Model L with $\delta = -0.4$.

initializers; see Figure 3.8), and it was not true for CE/Model A (for some random initializers; see Figure 4.12). Figure 6.6 compares the amount of test-set accuracy improvement under Model L with the same measure under Model A (both with EM training). Neither consistently improves random models more than the other.

For more comparison among methods on random initializers, see Sections 4.6.5, 5.3.2, 6.2.2, and 7.4.

## 6.2 Annealing $\delta$

We now turn to the central idea of this chapter: **structural annealing**. Instead of *fixing* $\delta$, structural annealing starts with $\delta \ll 0$ and gradually increases it. The goal is to force the learner to predict the most local patterns *first*, then gradually allow it to focus on more distant relationships. The algorithm is given in Figure 6.7 in a somewhat general form (without specifying exactly what the structural features are). This highlights that the idea can be applied in many ways: if there are features that quantify *simple* structures versus more complex ones, we can use them in structural annealing.

As in DA, there is an annealing schedule, here given by starting and ending values

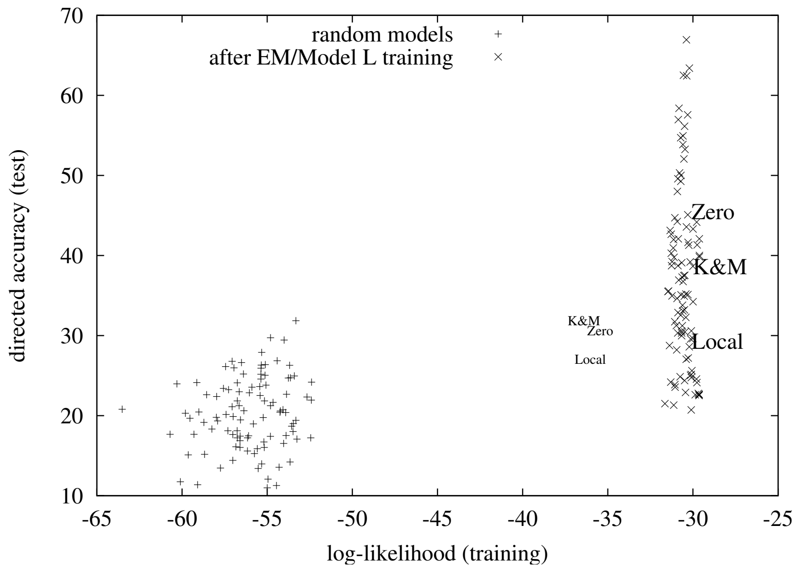


Figure 6.4: Accuracy on test data *vs.* log-likelihood on training data: 100 random models (and our three initializers) before and after EM training with Model L ($\delta = -0.4$). Small labels correspond to our initializers before training, large labels to after training. Compare with Figure 3.7 on page 51. The log-likelihood is off by a constant additive term because we have not renormalized; this means the absolute $x$-axis values are not directly comparable to Figure 3.7.

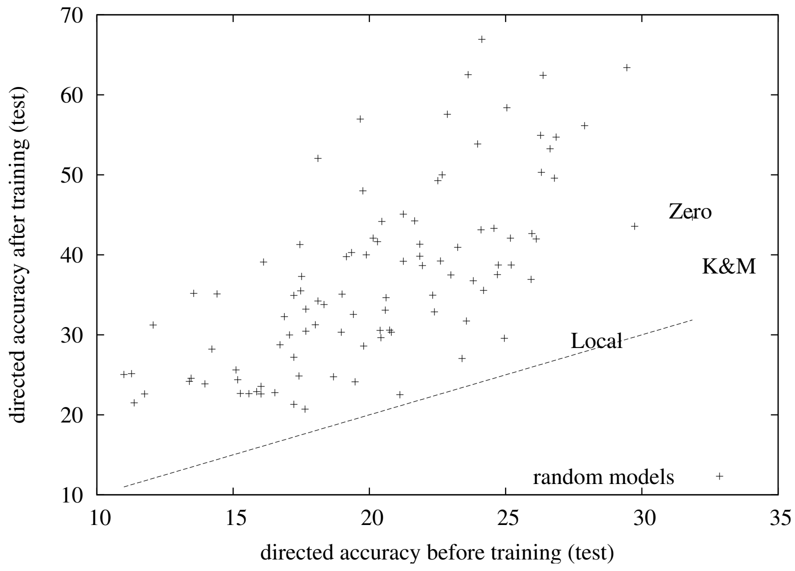


Figure 6.5: Directed accuracy on test data: 100 models after EM/Model L training *vs.* before training. The text labels correspond to the more carefully designed initializers. The line is $y = x$. Compare with Figure 3.8 on page 52.

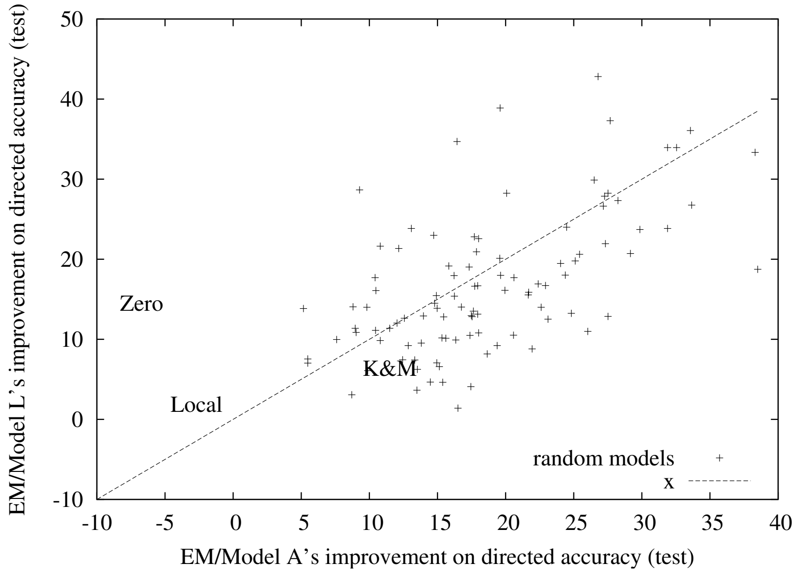


Figure 6.6: Absolute improvement (or reduction) of directed accuracy on test data via Model L *vs.* Model A. No regularization or smoothing was applied. Using EM, Model L makes better use of good initializers Zero and Local than Model A, but neither model consistently gives more improvement. The line is $y = x$.

```
STRUCTURAL ANNEALING (stochastic grammars):

1. Initialize $\vec{\theta}^{(0)}$ and let $\vec{\sigma} \leftarrow \vec{\sigma}_0$.

2. Do:

   (a) (E step) For each rule $\mathbf{r} \in \mathcal{R}$ (the rules of the grammar) let $c_\mathbf{r} \leftarrow 0$. For each
       example $\mathbf{x}^t$ in $\vec{\mathbf{x}}^t$:

       • Run the relevant Inside and Outside algorithms to compute $p_{\vec{\theta}^{(i)},\vec{\sigma}}(\mathbf{x})$. For
         all $\mathbf{r} \in \mathcal{R}$, let $c_\mathbf{r} \leftarrow c_\mathbf{r} + \mathbf{E}_{p_{\vec{\theta}^{(i)},\vec{\sigma}}(\mathbf{Y}|\mathbf{x}^t)}\left[f_\mathbf{r}(\mathbf{x}^t, \mathbf{Y})\right]$.

       $\vec{c}$ now contains sufficient statistics for the parameters of the model.

   (b) (M step; same as EM) Compute $\vec{\theta}^{(i+1)}$ by normalizing $\vec{c}$. That is, if $\mathbf{r}$ is in the
       partition $\mathcal{R}_i$ of the grammar rules, let $\theta_\mathbf{r} \leftarrow \frac{c_\mathbf{r}}{\sum_{\mathbf{r}' \in \mathcal{R}_i} c_{\mathbf{r}'}}$. This is the maximum like-
       lihood estimate for $\vec{\theta}$ on the complete data distribution $\tilde{p} \cdot q$, given the sufficient
       statistics.

   (c) If $\vec{\theta}^{(i+1)} \approx \vec{\theta}^{(i)}$ then proceed to step 3; otherwise let $i \leftarrow i + 1$ and go to step 2a.

3. Stop if, for all annealed parameters $\sigma_i$,

       $\sigma_i \geq \sigma_{f\,i}$    where $\Delta\sigma_i > 0$
       $\sigma_i \leq \sigma_{f\,i}$    where $\Delta\sigma_i < 0$

   Otherwise $\vec{\sigma} \leftarrow \vec{\sigma} + \Delta\vec{\sigma}$ and go to step 2.
```

Figure 6.7: Structural annealing for stochastic grammars. Here $\vec{\sigma}$ is a set of structural parameters (e.g., $\delta$ in Section 6.2 or $\beta$ in Section 6.3) that are not trained, but rather manipulated over time.

for $\delta$ ($\delta_0$ and $\delta_f$, respectively), and a value by which it is changed (arithmetically) after each epoch ($\Delta\delta$). This is yet another hyperparameter problem; here we explore different annealing schedules for $\delta$, illustrate the range of performance, and apply model selection across these settings.

### 6.2.1 Experiments

We tested a range of initial values for $\delta$, $\delta_0$ from -1 to -0.2, and two different growth factors $\Delta\delta$ (0.1 and 0.05). The stopping $\delta$ ($\delta_f$) and the smoothing level $\lambda$ were selected using supervised model selection. Table 6.2 shows the performance of these trials, with the different initializers. Notice that the choice of $\delta_0$ is very important, and that "playing it safe" by starting with $\delta_0 \ll 0$ does not necessarily yield the best performance, and that the choice of $\Delta\delta$ seems to have little effect on accuracy, at least for the two values tested.

As for $\delta_f$, comparing Tables 6.2 and 6.3 shows that simply stopping at $\delta = 0$ performs nearly as well as selecting the stopping value with development data. However, continuing training slightly past $\delta = 0$ seems to be advantageous. This might be surprising, since positive $\delta$ is a poor choice (Table 6.1). This suggests that annealing $\delta$ can trap the learner at a model that prefers local attachments *too much* and is therefore less accurate. Continuing annealing past 0, then, pulls the learner away from this tendency.

Comparing, for example, the trees produced in the $\delta_0 = -0.6$, $\Delta\delta = 0.1$, $\lambda = 10$, Zero initializer condition, the (slight) improvement in performance between stopping at $\delta = 0$ and $\delta = 0.1$ involved 143 corrected attachments, 89 of which were reattachments to a more distant parent. Of the 91 reattachments that increased error (i.e., parsing with the model trained to $\delta = 0$ gave the correct attachment, but after training to $\delta = 0.1$, the attachment was made differently), 70 were reattachments to a more distant parent. The later ($\delta \to 0.1$) model is more inclined to make longer attachments when applied to parsing, and more often than not this improves error (though the balance is slight).

Figure 6.8 shows performance at different potential stopping points for some conditions. Generally speaking, the better the initial setting, the better the annealed model will be; there is usually a slight increase in performance after $\delta$ passes 0. Notice that training too long (annealing too far) has catastrophic results, as continued training past the "peak" leads to a sharp decline in performance.

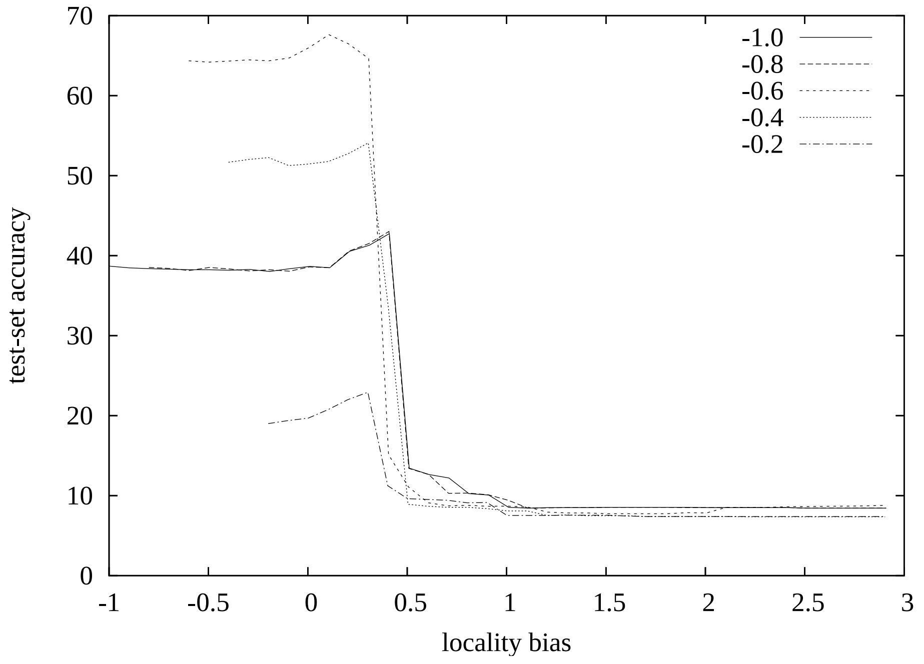
Further error analysis on the selected annealed-$\delta$ condition ($\lambda = 10$, $\vec{\theta}^{(0)} =$ Zero,

| | | Zero | | | | K&M | | | | Local | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | test accuracy | | | | test accuracy | | | | test accuracy | |
| $\delta_0$ | $\Delta\delta$ | $\lambda$ | $\delta_f$ | directed | undirected | $\lambda$ | $\delta_f$ | directed | undirected | $\lambda$ | $\delta_f$ | directed | undirected |
| – | – | 10 | – | 23.8 | 58.9 | $10^{-2/3}$ | – | 41.6 | 62.2 | 10 | – | 24.4 | 59.4 |
| – | – | 10 | -0.6 | 61.8 | 69.4 | 10 | -0.2 | 49.6 | 66.1 | 0 | -1.0 | 31.3 | 56.4 |
| -1.0 | 0.05 | 1 | 0.15 | 49.7 | 65.9 | 1 | 0.2 | 50.0 | 65.4 | 1 | 0.2 | **44.7** | **63.6** |
| -1.0 | 0.1 | 1 | 0.1 | 48.8 | 66.1 | 1 | 0.1 | 48.7 | 66.0 | $10^{-1/3}$ | 0.1 | **42.7** | **63.6** |
| -0.8 | 0.05 | $10^{-1/3}$ | 0.15 | 49.1 | 65.0 | $10^{-1/3}$ | 0.15 | 49.2 | 65.3 | 1 | 0.2 | **44.6** | **63.6** |
| -0.8 | 0.1 | 1 | 0.1 | 48.8 | 66.1 | 1 | 0.1 | 48.7 | 66.0 | $10^{1/3}$ | 0.3 | **43.9** | **62.0** |
| -0.6 | 0.05 | 10 | 0.1 | **66.7** | **73.1** | $10^{1/3}$ | 0.15 | **50.3** | **67.2** | 1 | 0.2 | **44.4** | **63.4** |
| -0.6 | 0.1 | 10 | 0.1 | **66.7** | **73.1** | 1 | 0.1 | 48.3 | 65.5 | $10^{1/3}$ | 0.3 | **43.9** | **62.0** |
| -0.4 | 0.05 | 10 | 0.3 | 53.8 | 66.2 | 10 | 0.15 | **51.5** | **67.9** | $10^{-2/3}$ | 0.05 | **33.2** | **66.3** |
| -0.4 | 0.1 | 10 | 0.3 | 53.4 | 66.0 | 10 | 0.1 | **51.2** | **67.6** | $10^{-2/3}$ | 0.1 | **32.8** | **65.4** |
| -0.2 | 0.05 | 0 | 0.05 | 37.4 | 63.7 | $10^{2/3}$ | 0 | 49.7 | 66.3 | 1 | 0.05 | **33.4** | **66.8** |
| -0.2 | 0.1 | $10^{-2/3}$ | 0.0 | 37.7 | 63.9 | 10 | 0.1 | **50.2** | **66.9** | 0 | 0.1 | 32.5 | **66.1** |
| unsupervised model selection: $\vec{\theta}^{(0)} = $ K&M, $\lambda = 10^{-2/3}$, $\delta_0 = -0.2$, $\Delta\delta = 0.1$, $\delta_f = 0.1$ | | | | | | | | | | | | 43.6 | 62.8 |
| supervised model selection: $\vec{\theta}^{(0)} = $ Zero, $\lambda = 10$, $\delta_0 = -0.6$, $\Delta\delta = 0.1$, $\delta_f = 0.1$ | | | | | | | | | | | | 66.7 | 73.1 |

Table 6.2: Model L trained with structural annealing on $\delta$ with different initializers and bias schedules $\langle \delta_0, \Delta\delta \rangle$. Supervised model selection was applied to select $\lambda$ and $\delta_f$ (the final value of $\delta$ during annealing). The first two lines correspond, respectively, to EM/Model A with supervised selection of $\lambda$, and to locality-biased EM/Model L (without annealing) with supervised selection of $\lambda$ and $\delta$. Boldface marks trials that achieved significantly better accuracy (sign test, $p < 0.05$) than the locality-biased trial in the same column (line 2). The last line selects a model (supervisedly) across $(\vec{\theta}^{(0)}, \lambda, \delta_0, \Delta\delta, \delta_f)$. For a graphical comparison against other methods over six languages, see Figure 8.2 (page 213) and Figure 8.3 (page 214).

|  |  | Zero |  |  | K&M |  |  | Local |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | test accuracy |  |  | test accuracy |  |  | test accuracy |  |
| $\delta_0$ | $\Delta\delta$ | $\lambda$ | directed | undirected | $\lambda$ | directed | undirected | $\lambda$ | directed | undirected |
| – | – | 10 | 23.8 | 58.9 | $10^{-2/3}$ | 41.6 | 62.2 | 10 | 24.4 | 59.4 |
| – | – | 10 ($\delta = -0.6$) | 61.8 | 69.4 | 10 ($\delta = -0.2$) | 49.6 | 66.1 | 0 ($\delta = -1.0$) | 31.3 | 56.4 |
| -1.0 | 0.05 | $10^{-2/3}$ | 39.2 | 60.1 | $10^{-1/3}$ | 38.7 | 59.6 | $10^{-1/3}$ | 32.0 | **57.0** |
| -1.0 | 0.1 | $10^{-1/3}$ | 38.8 | 59.8 | $10^{-2/3}$ | 39.2 | 60.2 | 1 | 32.0 | **57.0** |
| -0.8 | 0.05 | 1 | 38.8 | 59.9 | $10^{-1/3}$ | 38.5 | 59.4 | 0 | **32.9** | **57.6** |
| -0.8 | 0.1 | $10^{-1/3}$ | 38.8 | 59.8 | $10^{-1/3}$ | 39.0 | 59.9 | 0 | **32.8** | **57.6** |
| -0.6 | 0.05 | 10 | **65.6** | **72.2** | $10^{1/3}$ | 38.7 | 59.9 | 0 | **32.9** | **57.7** |
| -0.6 | 0.1 | 10 | **65.0** | **71.7** | $10^{-1/3}$ | 38.9 | 59.9 | 0 | **32.9** | **57.7** |
| -0.4 | 0.05 | 10 | 50.2 | 64.4 | 10 | **50.6** | **67.0** | $10^{-2/3}$ | 32.0 | **66.2** |
| -0.4 | 0.1 | 10 | 50.5 | 64.8 | 10 | **50.7** | **67.2** | $10^{-2/3}$ | 32.0 | **66.2** |
| -0.2 | 0.05 | $10^{-2/3}$ | 37.6 | 64.0 | $10^{2/3}$ | 49.7 | 66.3 | $10^{-2/3}$ | 30.3 | **66.9** |
| -0.2 | 0.1 | $10^{-2/3}$ | 37.7 | 63.9 | $10^{2/3}$ | 49.8 | 66.5 | $10^{-2/3}$ | 30.3 | **66.9** |

Table 6.3: Model L trained with structural annealing on $\delta$ with different initializers and bias schedules $\langle \delta_0, \Delta\delta \rangle$. selection was applied to select $\lambda$, and training stopped after $\delta = 0$. The first two lines correspond, respectively, to EM/Model A with supervised selection of $\lambda$, and to locality-biased EM/Model L (without annealing) with supervised selection of $\lambda$ and $\delta$. Boldface marks trials that achieved significantly better accuracy (sign test, $p < 0.05$) than the locality-biased trial in the same column (line 2).



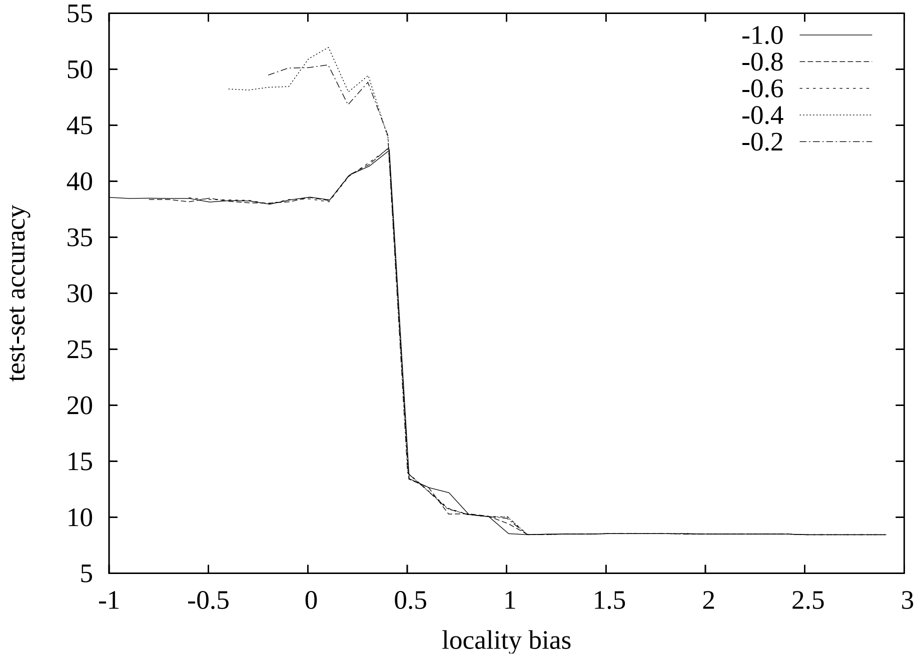


Figure 6.8: Test-set directed attachment accuracy of models at different potential stopping values of $\delta_f$. Here we show the Zero-initialized, $\lambda = 10$, $\Delta\delta = 0.1$ trials (above) and the K&M-initialized, $\lambda = 10$, $\Delta\delta = 0.1$ trials (below). Each curve corresponds to a different starting value $\delta_0$; time proceeds from left to right. The $x$-axis does not track time linearly; it plots accuracy after each $\delta$-epoch (earlier epochs typically take longer than later ones).

$\delta_0 = -0.6$, $\Delta\delta = 0.1$, $\delta_f = 0.1$) is shown in Tables 6.4, 6.5, and 6.6. The first of these shows undirected accuracy broken down by the most frequent unordered tag-pair types. The most notable problem has to do with NNP-NNP links, the conventional dependency structure of which is difficult to learn without lexicalization (they correspond to sequences of proper nouns, usually, in which lexical features would be necessary to learn patterns). Similarly to CE (Table 4.3, page 102), adjectives are too frequently linked to determiners (instead of nouns; DT/JJ *vs.* DT/NN, for example). Considering the distribution over link types hypothesized by this model, the undirected distribution is comparable to CE's (0.05 bits of mean KL divergence to the mean with the true distribution) and the directed distribution is even better (0.10 bits, compared to CE's 0.15 and EM's 0.28).

Table 6.5 shows that, even more than CE (Table 4.4, page 103), structural annealing improves both precision and recall on longer attachments. This is contingent on stopping at the right time; the table shows the same statistics after earlier epochs (the first one, $\delta = -0.6$, and also after $\delta = -0.3$). Notice how recall on longer attachments improves over time. Table 6.6 shows that structural annealing's hypothesized trees tend to shorten the path between a given node and its true parent, as compared to EM and CE.

### 6.2.2 Further Analysis (Random Initializers)

Figure 6.9 shows the spread of accuracy performance of 100 random models (the same ones used in previous chapters) before training and after training initialized from each one, with EM/Model A and with SA/Model L ($\delta_0 = -0.6, \Delta\delta = 0.1, \delta_f = 0$). We consider here only one annealing schedule, noting that $\delta_0 = -0.6$ tended to work fairly well across trials, stopping $\delta_f$ near 0 tended to be reasonable (if not optimal), and $\Delta\delta$ seems to have little effect on accuracy.

Figure 6.10 further illustrates that SA, like EM, may select any of similar likelihood--valued local optima: 100 random initializers all find different local maxima with about the same function value and very different accuracies on test data. However, as we saw for EM/Model L, and illustrated for SA/Model L in Figure 6.11, SA/Model L training always results in a more accurate model than the initializer $\vec{\theta}^{(0)}$. This was not true in the case of EM/Model A (for the Local and Zero initializers; see Figure 3.8), and it was not true for CE/Model A (for some random initializers; see Figure 4.12). It did, however, hold for SDA/Model A (see Figure 3.8), which behaved similarly to SA/Model L. Figure 6.12

| tag types | | count in hyp. | precision | recall | count in gold |
|---|---|---|---|---|---|
| DT | NN | 236 | 82.6 | 85.5 | 228 |
| NNP | NNP | 209 | 60.3 | 70.4 | 179 |
| IN | NN | 129 | 88.4 | 89.1 | 128 |
| NN | VBD | 120 | 87.5 | 87.5 | 120 |
| JJ | NN | 53 | 69.8 | 31.4 | 118 |
| NN | VBZ | 117 | 84.6 | 90.8 | 109 |
| JJ | NNS | 65 | 95.4 | 76.5 | 81 |
| IN | NNS | 76 | 81.6 | 83.8 | 74 |
| NNS | VBD | 79 | 86.1 | 100.0 | 68 |
| NN | NN | 67 | 86.6 | 87.9 | 66 |
| NN | NNS | 47 | 85.1 | 65.6 | 61 |
| IN | VBD | 76 | 67.1 | 86.4 | 59 |
| RB | VBD | 59 | 78.0 | 82.1 | 56 |
| NN | NNP | 19 | 57.9 | 19.6 | 56 |
| NNS | VBP | 51 | 88.2 | 88.2 | 51 |
| NNP | VBD | 52 | 88.5 | 95.8 | 48 |
| CD | CD | 51 | 86.3 | 95.7 | 46 |
| RB | VBZ | 59 | 72.9 | 97.7 | 44 |
| NNP | VBZ | 43 | 90.7 | 88.6 | 44 |
| CD | NN | 37 | 67.6 | 58.1 | 43 |
| PRP | VBD | 35 | 91.4 | 78.0 | 41 |
| CD | IN | 38 | 63.2 | 60.0 | 40 |
| TO | VBD | 43 | 90.7 | 100.0 | 39 |
| PRP | VBZ | 36 | 97.2 | 89.7 | 39 |
| DT | NNS | 38 | 92.1 | 89.7 | 39 |
| MD | VB | 38 | 97.4 | 97.4 | 38 |
| TO | VB | 32 | 100.0 | 94.1 | 34 |
| CD | TO | 38 | 44.7 | 51.5 | 33 |
| CD | NNS | 28 | 67.9 | 57.6 | 33 |
| IN | NNP | 36 | 72.2 | 81.3 | 32 |
| DT | NNP | 39 | 23.1 | 60.0 | 15 |
| DT | JJ | 66 | 3.0 | 100.0 | 2 |

Table 6.4: Undirected precision and recall of Model L, trained with structural annealing on $\delta$, by the unordered tag-pair type. Note that attachments in either direction and with the tags in either order are counted together. Tag-pair types with a count $\geq 30$ in the hypothesized annotation or the gold-standard are listed. Compare with Table 3.3 on page 61, for EM.

| $\delta = -0.6$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| length: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| hyp. | 2506 | 371 | 160 | 65 | 31 | 16 | 5 | 2 | 2 |
| count: | 1249 1257 | 175 196 | 53 107 | 30 35 | 19 12 | 9 7 | 3 2 | 0 2 | 1 1 |
| precision: | 70.0 | 74.9 | 63.8 | 64.6 | 29.0 | 37.5 | 0.0 | 50.0 | 50.0 |
| | 73.9 49.6 | 65.1 70.9 | 39.6 69.2 | 46.7 71.4 | 26.3 25.0 | 22.2 57.1 | 0.0 0.0 | – 50.0 | 100.0 0.0 |
| recall: | 89.4 | 39.9 | 35.9 | 36.2 | 16.4 | 20.0 | 0.0 | 25.0 | 100.0 |
| | 74.1 87.4 | 36.4 36.3 | 19.8 41.6 | 28.0 37.9 | 14.7 14.3 | 14.3 25.0 | 0.0 – | 0.0 100.0 | 100.0 – |
| $\delta = -0.3$ | | | | | | | | | |
| hyp. | 2497 | 376 | 152 | 70 | 35 | 18 | 6 | 2 | 2 |
| count: | 1170 1327 | 189 187 | 52 100 | 34 36 | 22 13 | 12 6 | 3 3 | 0 2 | 1 1 |
| precision: | 70.0 | 77.9 | 65.1 | 60.0 | 37.1 | 33.3 | 0.0 | 50.0 | 50.0 |
| | 76.6 48.0 | 66.7 75.4 | 44.2 70.0 | 44.1 63.9 | 31.8 30.8 | 25.0 50.0 | 0.0 0.0 | – 50.0 | 100.0 0.0 |
| recall: | 89.2 | 42.1 | 34.9 | 36.2 | 23.6 | 20.0 | 0.0 | 25.0 | 100.0 |
| | 71.9 89.2 | 40.3 36.8 | 21.7 39.3 | 30.0 34.8 | 20.6 19.0 | 21.4 18.8 | 0.0 – | 0.0 100.0 | 100.0 – |
| $\delta = 0.1$ | | | | | | | | | |
| hyp. | 2158 | 509 | 233 | 128 | 67 | 39 | 17 | 6 | 1 |
| count: | 1154 1004 | 211 298 | 66 167 | 41 87 | 28 39 | 9 30 | 8 9 | 1 5 | 1 0 |
| precision: | 75.9 | 73.7 | 68.2 | 60.2 | 46.3 | 46.2 | 52.9 | 16.7 | 100.0 |
| | 80.2 56.0 | 68.7 68.1 | 62.1 65.3 | 53.7 52.9 | 60.7 28.2 | 55.6 40.0 | 62.5 11.1 | 0.0 20.0 | 100.0 – |
| recall: | 83.6 | 53.9 | 56.0 | 66.4 | 56.4 | 60.0 | 75.0 | 25.0 | 100.0 |
| | 74.2 78.7 | 46.3 53.0 | 38.7 61.2 | 44.0 69.7 | 50.0 52.4 | 35.7 75.0 | 45.5 100.0 | 0.0 100.0 | 100.0 – |
| gold | 1960 | 696 | 284 | 116 | 55 | 30 | 12 | 4 | 1 |
| count: | 1246 714 | 313 383 | 106 178 | 50 66 | 34 21 | 14 16 | 11 1 | 3 1 | 1 0 |

Table 6.5: Precision and recall by string distance between parent and child, for the supervisedly-selected trial of Model L trained by structural annealing on $\delta$. We show tables corresponding to models after training to different values of $\delta$. Notice that recall for longer dependencies generally increases as $\delta$ moves from $-0.6$ to $-0.3$ to $0.1$. The large-typeface numbers show the undirected attachment precision and recall of dependencies at different lengths. Small-typeface numbers give the directed attachment precision and recall for left children and right children, by length. Compare with Table 3.4 on page 62, for EM.

| path length | ATTACH-RIGHT % of tag tokens undirected | ATTACH-RIGHT % of tag tokens directed | MAP/EM % of tag tokens undirected | MAP/EM % of tag tokens directed | CE/DEL1ORTRANS1 % of tag tokens undirected | CE/DEL1ORTRANS1 % of tag tokens directed | MAP/SA % of tag tokens undirected | MAP/SA % of tag tokens directed |
|---|---|---|---|---|---|---|---|---|
| 1 | 55.3 | 36.0 | 64.9 | 47.3 | 66.9 | 57.1 | 74.2 | 68.7 |
| 2 | 20.0 | 9.6 | 25.2 | 21.1 | 21.3 | 15.3 | 18.9 | 14.6 |
| 3 | 9.8 | 5.0 | 7.1 | 5.8 | 7.7 | 5.3 | 4.9 | 3.1 |
| 4 | 5.3 | 3.6 | 2.0 | 1.3 | 2.7 | 1.8 | 1.4 | 0.8 |
| 5 | 3.1 | 2.5 | 0.7 | 0.5 | 1.2 | 0.8 | 0.5 | 0.2 |
| 6 | 2.8 | 2.4 | 0.0 | 0.0 | 0.2 | 0.1 | 0.1 | 0.1 |
| 7 | 1.8 | 1.8 | 0.0 | 0.0 | 0.1 | 0.1 | 0.1 | 0.1 |
| 8 | 1.1 | 1.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 9 | 0.6 | 0.6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $\infty$ | – | 37.4 | – | 23.9 | – | 19.5 | – | 12.6 |

Table 6.6: Path analysis of the selected MAP/EM model, the selected CE/DEL1ORTRANS1 model, and the selected structural annealing model (annealing on $\delta$). The macro-averaged undirected path length in hypothesized trees between a word and its gold-standard parent is 1.31 for MAP/EM, 1.27 for DEL1ORTRANS1, and 1.19 for SA; under a sign test the SA path averages were significantly shorter than both of the other conditions ($p < 0.000001$). Note that attachment accuracy is *not* identical to the first line because here we count paths to the wall (just another node in the tree), whereas our usual accuracy measures do not count attachments to the wall.

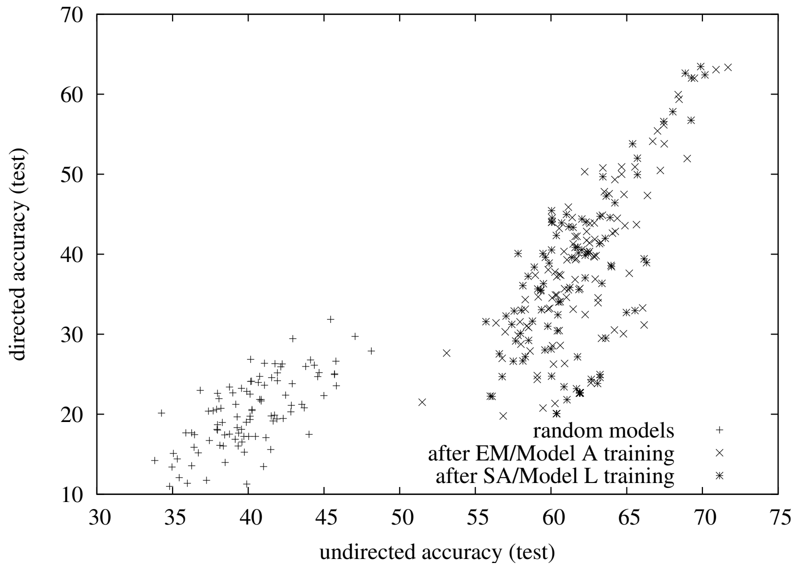


Figure 6.9: Accuracy on test data (directed *vs.* undirected): 100 random models before training, after EM training of Model A, and after SA training of Model L with $\delta_0 = -0.6, \Delta\delta = 0.1, \delta_f = 0$.

compares the amount of test-set accuracy improvement under Model L with the same measure under Model A (both with EM training). Neither consistently improves random models more than the other, though with selection across more annealing schedules and smoothing values the result could be different.

For more comparison among methods on random initializers, see Sections 4.6.5, 5.3.2, 6.1.4, and 7.4.

### 6.2.3 Semisupervised Training

In Section 3.4.2 we considered the use of the annotated development dataset to construct an initializer (i.e., train supervised and initialize unsupervised training with the learned model) and to construct a prior for MAP estimation—both as alternatives to our unsupervised estimation/supervised selection strategy. We found that, with enough annotated data, either method was preferable to our strategy, though none of the methods tested performed as well as simple supervised training on the development dataset (except when only very small amounts of annotated data were used). (See Figure 3.10 on page 57, Section 3.4.2). We argued in Section 3.4.2 that our method is nonetheless a reasonable way

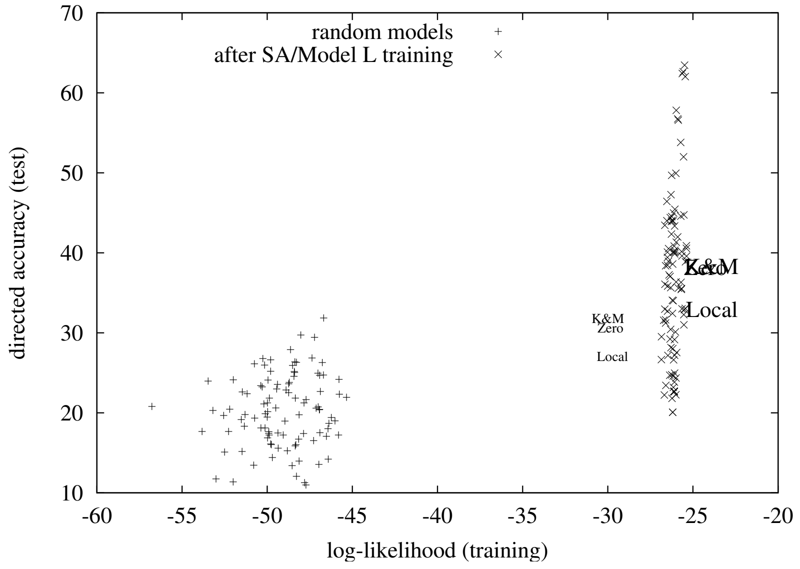


Figure 6.10: Accuracy on test data *vs.* log-likelihood on training data: 100 random models (and our three initializers) before and after SA training with Model L ($\delta_0 = -0.6, \Delta\delta = 0.1, \delta_f = 0$). Small labels correspond to our initializers before training, large labels to after training. Compare with Figure 3.7 on page 51.

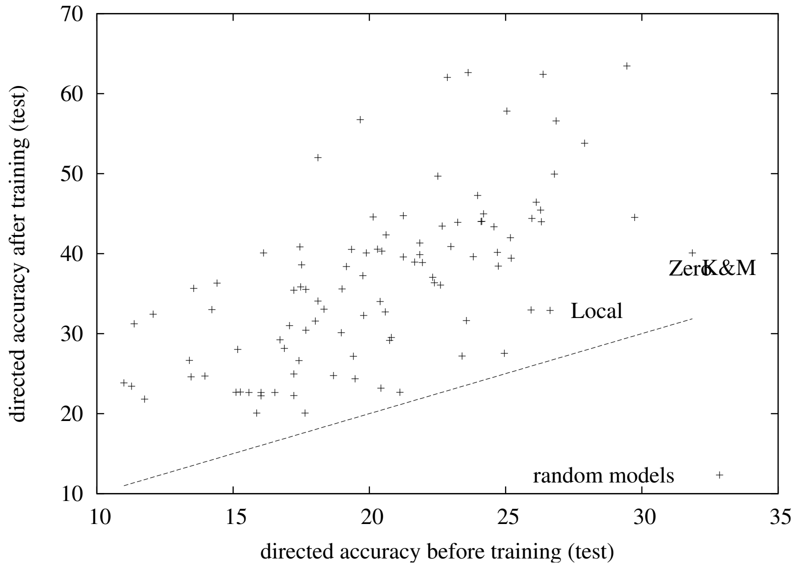


Figure 6.11: Directed accuracy on test data: 100 models after SA/Model L training *vs.* before training. The text labels correspond to the more carefully designed initializers. The line is $y = x$. Compare with Figure 3.8 on page 52.

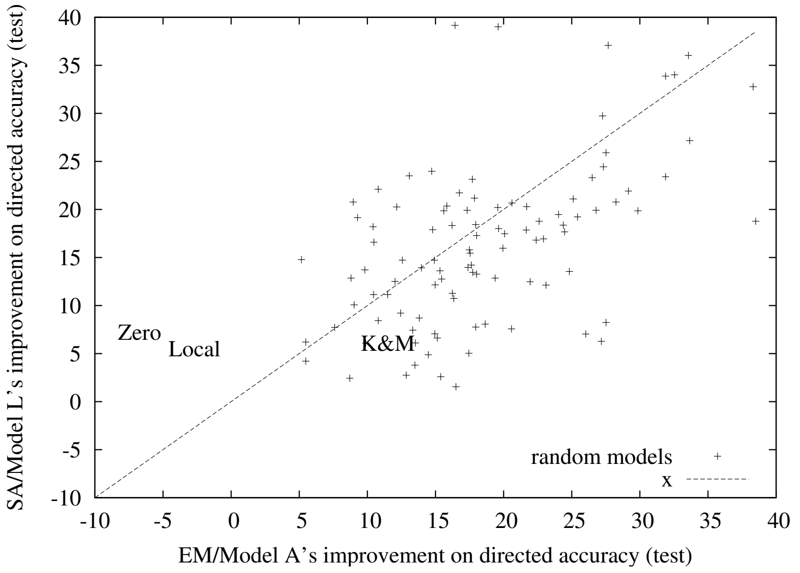


Figure 6.12: Absolute improvement (or reduction) of directed accuracy on test data via SA/Model L *vs.* EM/Model A. No regularization was applied and the annealing schedule was fixed at $\delta_0 = -0.6, \Delta\delta = 0.1, \delta_f = 0$. SA/Model L makes better use of good initializers Zero and Local and model selection than EM/Model A, but neither model consistently gives more improvement. Note that SA performs best with smoothing, and none of these models were smoothed. The line is $y = x$.

to evaluate performance of unsupervised estimation methods.

Here we consider how SA fares when the annotated data are used to build an initializer. Figure 6.13 illustrates the results, comparing the earlier methods with SA, for different values of $\delta_0$ ($\delta_f$ was fixed at 0, $\Delta\delta$ at 0.1, and $\lambda$ at 10). The differences between supervised initialization and supervised selection, for SA, are relatively small. The former outperforms the latter by a few accuracy points when larger amounts of annotated data are available, but the difference is less stark than for CE or EM. It is also interesting that supervised initialization with semisupervised training performs slightly worse than supervised initialization and *un*supervised training.

As in Sections 3.4.2 and 4.6.6, these results cast some doubt on our unsupervised training/supervised selection methodology. We suggest that the advantages of SA (and CE) over EM as an *unsupervised* estimator motivate further exploration on improved *semisupervised* estimation methods that make use of contrast in the objective function. Noting that none of the semisupervised methods explored here (and few explored elsewhere in the literature) outperform mere supervised training on the available annotated data, we leave the question of effective semisupervised parameter estimation to future work.

## 6.3 Segmentation Bias and Model S

A related way to focus on local structure early in learning is to *broaden* the hypothesis space $\mathcal{Y}_{\mathbf{x}}$ to include *partial* parse structures. Recall that the standard approach assumes $\mathbf{x}$ corresponds to the vertices in a single (here, projective) dependency tree that is hidden. The set $\mathcal{Y}_{\mathbf{x}}$ consists of all such trees. Instead, we will entertain every hypothesis in which $\mathbf{x}$ is a *sequence* of yields from *separate*, independently-generated trees. For example, $\langle x_1, x_2, x_3\rangle$ is the yield of one tree, $\langle x_4, x_5\rangle$ is the yield of a second tree, and $\langle x_6, ..., x_{|\mathbf{x}|}\rangle$ is the yield of a third. One extreme hypothesis is that $\mathbf{x}$ is $|\mathbf{x}|$ single-node trees. At the other end of the spectrum are the original hypotheses—single trees on the entirety of $\mathbf{x}$. Each has a nonzero probability.

Segmented analyses are intermediate representations that may be helpful for a learner to use to formulate notions of probable local structure, without committing to full trees.[2] Here we only *allow* unobserved breaks, never positing a hard segmentation of the training sentences. Over time, we increase the bias against broken structures, forcing the learner to

[2]Partial parsing has also been treated as a task in its own right: see Hindle (1990), *inter alia*.

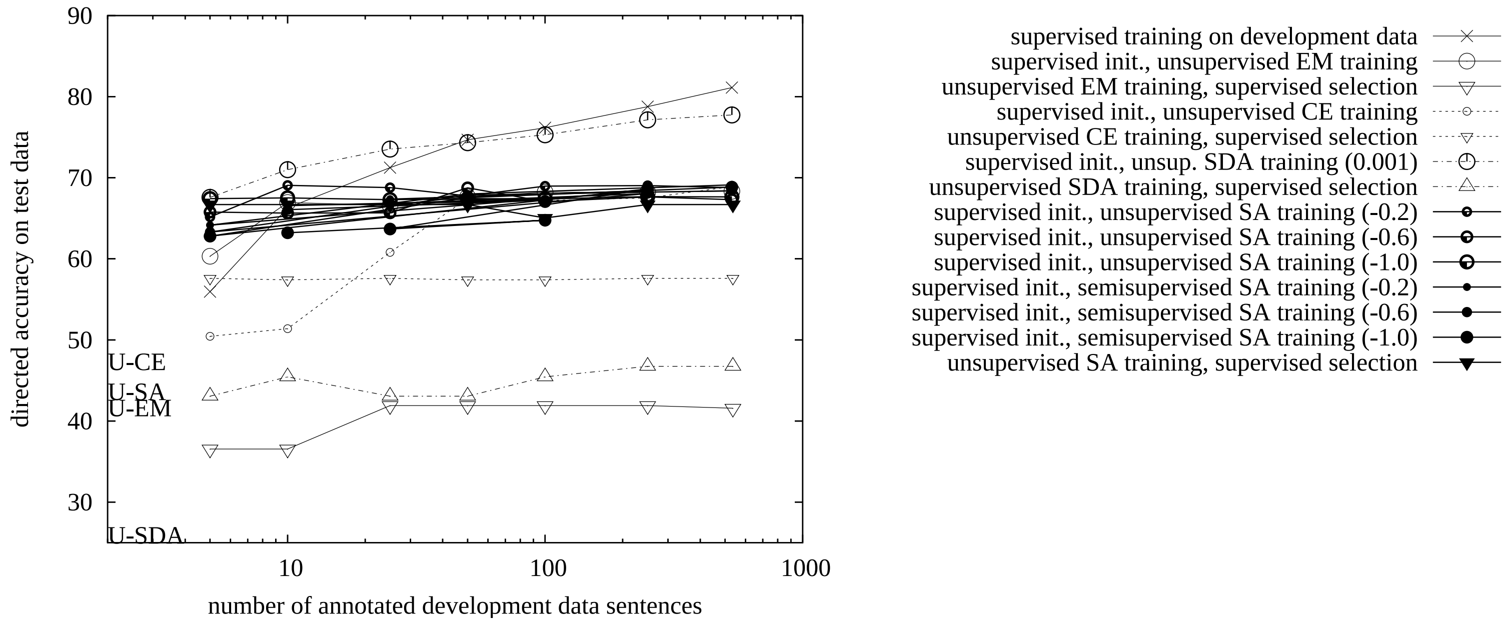


Figure 6.13: Different uses of the development dataset, as the size of the development dataset changes. The plot compares our standard setting of unsupervised estimation/supervised selection (with MAP/EM, CE/DEL1ORTRANS1, MLE/SDA, and MAP/SA) against the use of the annotated development dataset to construct an *initializer* for MLE/EM, unregularized CE/DEL1ORTRANS1, MLE/SDA, or MAP/SA (fixed $\lambda = 10, \delta_f = 0$; the parenthesized values refer to $\delta_0$) training. (Smoothing/regularization had very little effect on accuracy on EM and CE conditions.) The "U" labels to the left correspond to unsupervised model selection. When the initializer is not named, it has been selected over.

commit most of its probability mass to full trees.

To carry out this idea, we need to modify our model so that it can generate segmented structures. Like Model L, the new model, called Model S, is very similar to Model A. It includes the same set of parameters $\vec{\theta}$, and the inference equations for dynamic programming are very similar (see Appendix A). When Model S generates a sequence of trees, each tree is generated according to Model A, starting out by sampling the root node from $\lambda_{\mathbf{root}(\cdot)}$. Notice that $\mathcal{Y}_{\mathbf{x}}$ for Model S, which contains all partial and complete dependency parses of $\mathbf{x}$, is different from $\mathcal{Y}_{\mathbf{x}}$ for Models A and L (which contains only complete parses).

Model S has one additional parameter, $\beta$, which we will explain presently. Similar to $\delta$ in Model L, $\beta$ is not trained and is not present at decoding time; it is used only to aid in estimation. Like Model L, Model S can be thought of as a variant of Model A that is used to train $\vec{\theta}$.

### 6.3.1 Modeling Segmentation

In Model S, the total probability of a sequence $\mathbf{x}$, marginalizing over its hidden structures, sums up not only over trees, but over segmentations of $\mathbf{x}$. For completeness, we must include a probability model over the number of trees generated, which could be anywhere from $1$ to $|\mathbf{x}|$. The model over the number $T$ of trees given a sentence of length $n$ will take the following log-linear form:

$$p_\beta(T = t \mid n) = e^{t\beta} \Bigg/ \sum_{i=1}^{n} e^{i\beta} \tag{6.4}$$

where $\beta \in \mathbb{R}$ is the sole parameter. When $\beta = 0$, every value of $T$ is equally likely. For $\beta \ll 0$, the model prefers larger structures with few breaks. At the limit ($\beta \to -\infty$), we achieve Model A, which must explain $\mathbf{x}$ using a single tree. We start, however, at $\beta \gg 0$, where the model prefers smaller trees with more breaks. Gradually, we decrease $\beta$, preferring each word in $\mathbf{x}$ to be its own tree. Indeed, Model S is just Model A with one additional "brokenness" feature:

$$f_{\mathbf{brokenness}}(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} (\text{\# of connected components in } \mathbf{y}) - 1 \tag{6.5}$$

The weight of this feature is $\beta$. $\beta$ is chosen extrinsically (and time-dependently), rather than empirically—just as was done with $\delta$ in Model L in Section 6.1.

To be perfectly rigorous, Model S is problematic to describe as a stochastic process. Suppose we have a sequence $\mathbf{x}$ and $|\mathbf{x}| = 10$. Under this model, the number of trees (segments) is dependent upon the value of $|\mathbf{x}|$—we cannot have 11 trees, for example. But $|\mathbf{x}|$ depends on the number of trees, as well, since it is really just a random variable that is a deterministic function of the stochastic tree-generation process (how many times did any **child** feature/production rule fire?). To describe Model S as a stochastic process requires a cyclic dependency between $T$ and $|\mathbf{x}|$. All this means is that the model is deficient.[3]

One way to get rid of the deficiency is to replace Equation 6.4 with an exponential decay model that includes a stopping probability. This is almost exactly the same as Equation 6.4 (let $e^{\beta}$ be the probability of generating another tree and $1 - e^{\beta}$ be the probability of stopping), but does not allow us to *favor* more segmentation ($\beta > 0$). We have chosen the deficient model because we want more flexibility than exponential decay gives us—we sometimes want *more segmentation* to be *more probable*. Model S *does* give us exactly what we need for EM training: a proper posterior distribution $q$ over segmented trees.

### 6.3.2 Vine Parsing

At first glance broadening the hypothesis space to entertain all $2^{|\mathbf{x}|-1}$ possible segmentations may seem expensive. In fact, for Model S the dynamic programming computation is almost the same as summing or maximizing over connected dependency trees, as in Model A. For the latter, we use a dynamic programming algorithm that computes bottom-up a score for every parse tree by computing scores of partial structures.

Now note that any *sequence* of partial trees over $\mathbf{x}$ can be constructed by combining the same partial structures into trees. The only difference is that we are willing to consider unassembled sequences of these partial trees as hypotheses, in addition to the fully connected trees.

One way to accomplish this employs the special wall node, $x_0 = \$$. In Model A, $\mathbf{y}(0) = \{x_r\}$, the single root of the sentence said to be the child of \$. In the segmentation setting, we allow $\mathbf{y}(0)$ to have multiple children, instead of just one. Here, these children are independent of each other (e.g., generated by a unigram Markov model). Formally, to

[3]To say that a stochastic model is **deficient**, in simple terms, means that the stochastic process could generate outputs that are not well-formed. In other words, probability mass is "leaked" onto outcomes that are not valid structures. Log-linear models are one way to get around this, provided there are efficient summation algorithms for inference, which is not an option here, as noted in Section 3.5. It is unknown whether deficiency is anything to worry about on empirical or formal grounds, but it is a sign of sloppiness.

the production rules of Model A (Equation 2.2), we add, $\forall x \in \Sigma$:

$$e^{\beta} \cdot \lambda_{\mathbf{root}(x)} \quad S \rightarrow N[x]\, S \tag{6.6}$$

In supervised dependency parsing, Eisner and Smith (2005) showed that imposing a hard constraint on the whole structure—specifically that each non-\$ dependency arc cross fewer than $k$ words—can give guaranteed $O(|\mathbf{x}|k^2)$ runtime with little to no loss in accuracy (for simple models). If we had required a single tree with one root attached to \$, this constraint could lead to highly contrived parse trees, or none at all, for some sentences—both are avoided by the allowance of segmentation into a sequence of trees (each attached to \$). The construction of the "vine" (sequence of \$'s children) takes only $O(n)$ time once the chart has been assembled. Model S is essentially a weighted vine grammar (exactly equivalent to "Model B" in Eisner and Smith (2005)), with a unigram distribution over \$'s children. We do not impose any hard constraints on dependency length. The Dyna equations for Model S are given in Section A.5.

### 6.3.3 Experiments

We first trained Model S with different fixed values of $\beta$ in $[-1, 1]$ at increments of 0.2. As before, we tested MLE training and MAP training with the same set of six $\lambda$ values, and the same three initializers. This leads to 231 conditions. Supervised model selection gives the K&M, $\lambda = 1$, $\beta = 0.2$ setting, achieving 55.6% labeled accuracy and 67.0% unlabeled accuracy. Both are significantly better than the supervised-selected Model A MAP condition (K&M, $\lambda = 10^{-2/3}$). Unsupervised model selection gives the K&M, $\lambda = 10^{-2/3}$, $\beta = -1$ setting, achieving 42.0% labeled accuracy and 63.4% unlabeled accuracy; only the latter is significantly better than Model A trained with MAP.

We also experimented with structural annealing on $\beta$. We tested values of $\beta_0$ in $\{-1/2, 0, 1/2\}$, and used $\Delta\beta = -0.1$. Training continued until $\beta = -3$. Model selection over $\lambda$ and $\beta_f$ was applied; see Table 6.7 for results. If $\beta_0$ is sufficiently high, performance improvements are possible over training with fixed $\beta$. These results are not as impressive as those obtained with annealing on $\delta$ (Section 6.2), for this dataset, but the trend is similar and the experiment demonstrates that other kinds of structural annealing can be useful.

|  | Zero |  |  |  | K&M |  |  |  | Local |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | test accuracy |  |  |  | test accuracy |  |  |  | test accuracy |  |
| $\beta_0$ | $\lambda$ | $\beta_f$ | directed | undirected | $\lambda$ | $\beta_f$ | directed | undirected | $\lambda$ | $\beta_f$ | directed | undirected |
| – | 10 | – | 23.8 | 58.9 | $10^{-2/3}$ | – | 41.6 | 62.2 | 10 | – | 24.4 | 59.4 |
| – | $10^{2/3}$ | 0 | 30.1 | 54.7 | 1 | 0.2 | 55.6 | 67.0 | $10^{2/3}$ | 0 | 29.3 | 54.2 |
| -0.5 | 10 | -0.6 | 28.2 | 55.2 | $10^{-1/3}$ | -0.6 | 44.1 | 64.1 | 10 | -0.5 | 29.1 | **55.6** |
| 0.0 | 10 | -0.9 | 29.7 | 54.2 | $10^{1/3}$ | -3.0 | 55.4 | 67.1 | 10 | -0.9 | 29.8 | 54.3 |
| 0.5 | $10^{1/3}$ | -0.4 | **44.5** | **63.2** | $10^{-2/3}$ | 0.0 | **58.4** | **68.8** | $10^{1/3}$ | -0.5 | **34.8** | **58.0** |
| unsupervised model selection: $\vec{\theta}^{(0)}$ = K&M, $\lambda = 10^{-2/3}$ $\beta_0 = 0$, $\beta_f = -3$ |  |  |  |  |  |  |  |  |  |  | 42.1 | 63.0 |
| supervised model selection: $\vec{\theta}^{(0)}$ = K&M, $\lambda = 10^{-2/3}$, $\beta_0 = 0.5$, $\beta_f = 0$ |  |  |  |  |  |  |  |  |  |  | 58.4 | 68.8 |

Table 6.7: Model S trained with structural annealing on $\beta$, with different initializers and bias schedules $\langle \beta_0 \rangle$. Supervised model selection was applied to select $\lambda$ and $\beta_f$ (the final value of $\beta$ during annealing). The first two lines correspond, respectively, to Model A trained using EM with supervised selection of $\lambda$, and to Model S trained using EM (without annealing) with supervised selection of $\lambda$ and $\beta$. Boldface marks trials that achieved significantly better accuracy (sign test, $p < 0.05$) than the segmentation-biased trial in the same column (line 2).

## 6.4 Related Work

Annealing $\beta$ resembles the popular **bootstrapping** technique (Yarowsky, 1995), which starts out aiming for high precision, and gradually improves coverage over time. The idea is to first use a few labeled examples (or a few highly predictive features) to build a supervised classifier. The classifier is used to annotate a pool of unlabeled examples, and to select the ones it is most confident about. These are added to the annotated dataset, and a new classifier is learned. The annotation, selection, and dataset augmentation steps are repeated.

When the bias is strong ($\beta \gg 0$), we seek a model that maintains high dependency precision on (non-\$) attachments by attaching most tags to \$. Over time, as this is iteratively weakened ($\beta \to -\infty$), we hope to improve coverage (dependency recall). Bootstrapping was applied to syntax learning by Steedman, Osborne, Sarkar, Clark, Hwa, Hockenmaier, Ruhlen, Baker, and Crim (2003). Our approach differs in being able to remain partly agnostic about each tag's true parent (e.g., by giving 50% probability to attaching to \$), whereas Steedman *et al.* make a hard decision to retrain on a whole *sentence* fully or leave it out fully.

In earlier work, Brill and Marcus (1992) adopted a strategy for discovering phrase structure that emphasizes local relationships between part-of-speech tags, similar in spirit to our annealing of $\delta$. Learning from part-of-speech tagged corpora (like us), their idea was to score context-free production rules of the form $x \to x' \ x''$ by the distributional similarity of sequence $x$ (a single tag) and the sequence $x' \ x''$ (measured by KL divergence). The rules were then pruned, giving a weighted CFG. The strategy, while greedy and difficult to justify as a parameter estimation method, is notable for its focus on local tag-tag relationships.

Another related piece of work is the parameter estimation used for the IBM Candide translation models (Brown *et al.*, 1993). The IBM models consist of a set of progressively more complex models that define probability distributions $p(\mathbf{f} \mid \mathbf{e})$ over French sentences given English sentences using different stochastic processes. The first model predicts only a bag of words; later fertility (how many French words each English word generates) and distortion (the re-ordering of the French words *contra* the English) are modeled. With the exception of the simplest, these models cannot be efficiently estimated, even with EM. Many approximations were made. The relevant point here is that Brown *et al.* first estimated the simplest model, then used it to initialize training with the next, and so on. Al-Onaizan, Cuřin, Jahr, Knight, Lafferty, Melamed, Smith, Och, Purdy, and Yarowsky (1999) described a particular training regimen (how many iterations per model) that is still widely used with those models. Structural annealing is a softer alternative to that approach, though applying it directly to the Candide models would be very computationally intensive and might interact badly with the approximations required for inference with those models.

## 6.5 Future Work

We presented two features upon which structural annealing for Model A is straightforward. These are a sum-of-dependency-lengths feature and a "brokenness" feature. Others are imaginable, for this task and others. We mentioned machine translation above; we can imagine annealing locality bias in word alignment problems, or a bias toward unconnectedness (many words linked to the NULL word).

We also suggest a related idea: dynamic, soft feature selection. In structural annealing we gradually change feature weights to change learning bias over time. We might instead manipulate a *prior* over feature weights. This would allow any of the following strategies, in the log-linear setting:[4]

- Initially, the weight of a feature (or set of features) is kept close to zero. Gradually the prior is relaxed (e.g., $\sigma^2$ in a Gaussian prior is pushed toward $+\infty$), allowing these features to influence learning. In grammar induction, we might start with an unlexicalized model like Model A, and gradually anneal in lexical features. This would allow the model to learn a coarse structure first, then refine it with distinct lexicalized patterns that violate (or reinforce) the unlexicalized patterns.

- Features used early in learning could be phased out by doing the *opposite*: pushing feature weights toward zero by making the prior incrementally stronger with mode zero. For example, the constituent-context model of Klein and Manning (2001a) works very well for learning unlabeled, binary-branching constituent structure. Unfortunately, its feature set is huge and cannot generalize well to large datasets or long sentences, since it includes two features for every substring of part-of-speech

[4]These ideas were proposed in more detail in Smith (2004).

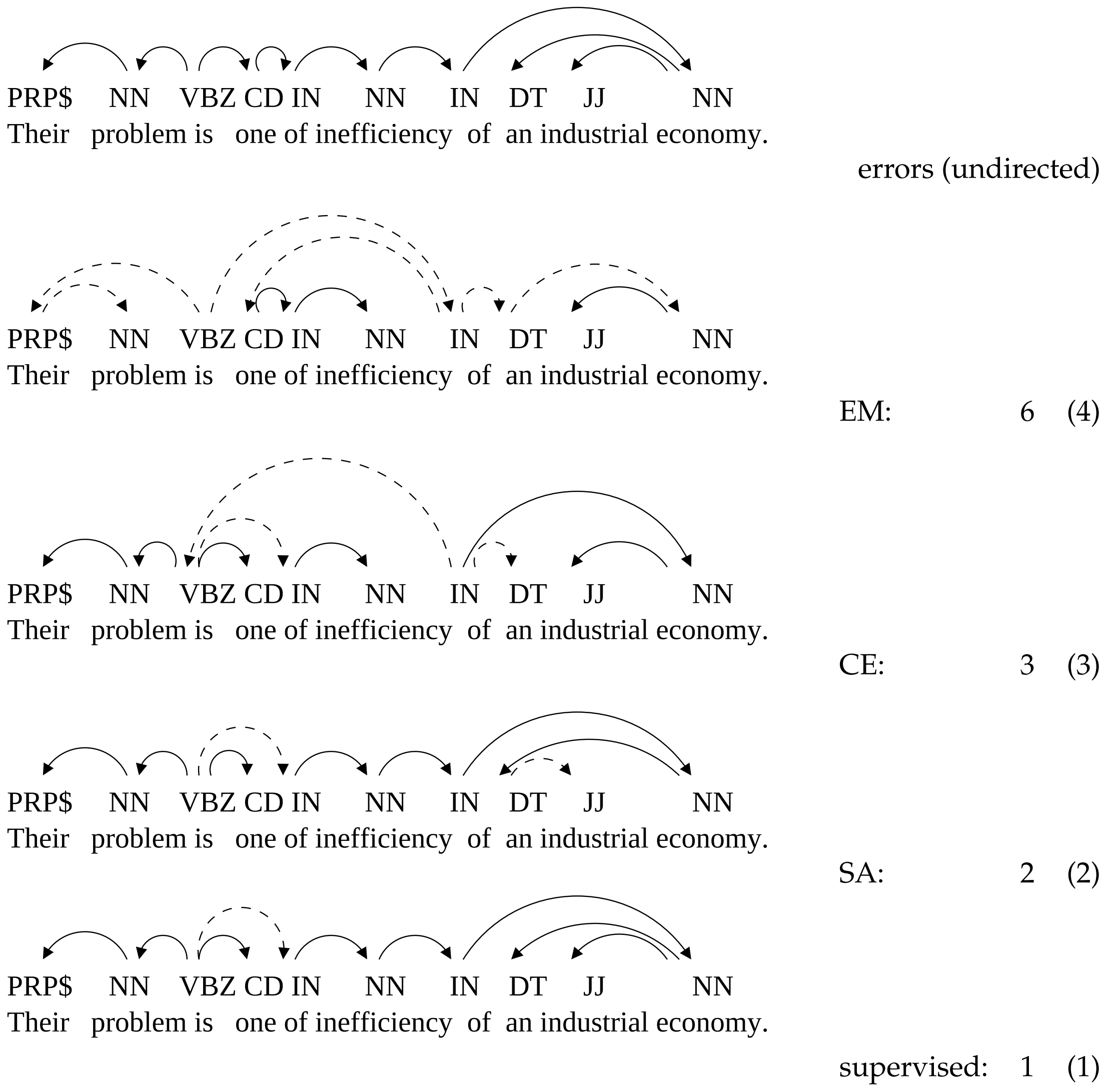

PRP$ NN VBZ CD IN NN IN DT JJ NN
Their problem is one of inefficiency of an industrial economy.
errors (undirected)
PRP$ NN VBZ CD IN NN IN DT JJ NN
Their problem is one of inefficiency of an industrial economy.
EM: 6 (4)
PRP$ NN VBZ CD IN NN IN DT JJ NN
Their problem is one of inefficiency of an industrial economy.
CE: 3 (3)
PRP$ NN VBZ CD IN NN IN DT JJ NN
Their problem is one of inefficiency of an industrial economy.
SA: 2 (2)
PRP$ NN VBZ CD IN NN IN DT JJ NN
Their problem is one of inefficiency of an industrial economy.
supervised: 1 (1)

Figure 6.14: An example sentence from our test set, drawn from the Penn Treebank, with four dependency parses: the gold standard, supervisedly selected EM, CE, and SA models, and a supervised model. This example was chosen because performance on it was typical for each parser (close to the parser's average). The numbers shown are the attachment errors and the undirected attachment errors. Errors with respect to the gold standard are shown as dashed lines. See Figures 1.1 (page 13) and 9.1 (page 9.1) for more examples.

tags seen in training. These features are useful for learning, but a weighted grammar is a preferred output of a grammar induction system. Klein and Manning (2004) experimented with combining the constituent-context model with a dependency grammar (the Model A used here); perhaps further stages of learning could phase out the constituent-context features and force the weighted grammar to do all of the explaining.

- Noting that the diagonal, single-$\sigma^2$ Gaussian priors used here are a special case of Gaussian priors that allow arbitrary covariances, features could be *tied* together (forced to have the same weight) using a large covariance parameter in the prior.[5] We could start out with features that are tied and gradually allow their weights to move apart. Initially, all verbs are the same, but over time, relevant feature covariances are moved toward zero, and the verbs are allowed (for instance) to select for different arguments. We could also do the reverse, assigning negative covariances to push feature weights apart (though it is harder to imagine why one would do this).

This approach could be carried out with a prescribed pattern (and different patterns compared experimentally), or perhaps empirical methods could be developed to drive the change in the prior over time.

Other kinds of structural annealing might be attempted. In Section 4.8 we described Model U, which scores *un*directed dependency trees, similar to link grammar (Sleator and Temperley, 1993), but without cycles. One could imagine starting with link grammar and annealing on a cyclicity feature. Over time the penalty toward acyclicity increases, until the model favors trees. Directionality could also be annealed in. This would help the model avoid settling too soon on a particular link direction (e.g., the determiner-noun problem discussed in Section 3.4.3).

Finally, as we mentioned in Section 3.7, our experimental results suffer from the limitation that we have trained and tested predominantly on short sentences. (This is consistent with recent work by others (Klein and Manning, 2002a).) Because longer sentences are likely to contain constructions not in our corpus, we cannot assume the induced grammar will perform well at parsing less restricted data (but see Section 7.1). In an annealing framework, we might start out by training a model on shorter sentences, and gradually encourage the model to pay attention to longer ones; the annealed parameter(s) would

[5]This idea is due to David Smith, personal communication.

|  |  | test accuracy |  |  |
|---|---|---|---|---|
|  |  | directed | undirected |  |
|  | ATTACH-RIGHT | 39.5 | 62.1 |  |
| Model A | MLE/EM (s-sel.) | 41.7 | 62.1 |  |
|  | MAP/EM (s-sel.) | 41.6 | 62.2 |  |
|  | CE/DEL1ORTRANS1 (s-sel.) | 57.6 | 69.0 | (Chapter 4) |
| Model L | MAP/fixed $\delta$ EM (s-sel.) | 61.8 | 69.4 |  |
|  | MAP/SA (s-sel.) | 66.7 | 73.1 |  |
| Model S | MAP/fixed $\beta$ EM (s-sel.) | 55.6 | 67.0 |  |
|  | MAP/SA (s-sel.) | 58.4 | 68.8 |  |

Table 6.8: Summary of baselines and results in this chapter.

control $\tilde{p}$, starting with a distribution favoring short (or otherwise "easy") sentences, gradually moving toward the actual empirical distribution in the corpus. Of course, as we pointed out earlier (Section 3.7), the Inside-Outside algorithm for these models has cubic runtime and may be prohibitive on very long sentences without fast approximations. An incremental variant that randomly samples from $\tilde{p}$ (while $\tilde{p}$ changes over time) might be appropriate; this is similar to example reweighting (Elidan *et al.*, 2002, discussed in Section 5.4.2) but exploits an intuitive notion of easy and hard examples. In Section 7.1 we consider models trained on smaller numbers of longer sentences, keeping the number of tags in the training set roughly stable.

## 6.6 Summary

This chapter presented two ways to augment Model A with simple structural features meant to aid in learning. We showed that a good choice of the weight for either feature can improve learning by biasing the learner toward certain kinds of structures. We then showed how gradually changing this feature weight during learning—structural annealing on it—produces even greater improvements. (We note again that the task on which improvements were demonstrated, the grammar induction task evaluated against gold-standard dependency annotations, is a relatively artificial task that may not directly bear on applications; see Section 1.4.) The biases explored here include a locality bias that makes the model favor trees with shorter dependencies, and a segmentation bias that makes the model favor "broken" or unconnected trees.

# Chapter 7

# Variations and Comparisons

*The hardest thing is to go to sleep at night, when there are so many urgent things needing to be done. A huge gap exists between what we know is possible with today's machines and what we have so far been able to finish.*

—Donald Knuth (1938–)

This chapter compares the EM baseline (Chapter 3) with contrastive estimation (Chapter 4), skewed deterministic annealing (Chapter 5), and structural annealing (Chapter 6) in a few additional ways. We consider training and testing on datasets that include longer sentences (Section 7.1). We test the robustness of the various learners to the quality of the input by training on induced (rather than gold-standard) part-of-speech tags (Section 7.2). We measure the quality of the learned parsers on the PARSEVAL scores by transforming the dependency trees into unlabeled constituent trees (Section 7.3). We plot the directed accuracy of the different training methods pairwise, for our standard initializers and the 100 random ones encountered throughout the thesis (Section 7.4). Finally, we combine contrastive estimation with structural bias and annealing (Section 7.5).

## 7.1 Sentence Length Effects

In this section we consider the performance of EM, CE, and SA on datasets that include longer sentences. First we discuss the performance of the selected models trained as previously described (on sentences of ten words or fewer), when applied to longer test sentences. We then show how performance is affected when we *train* on datasets that include longer sentences.

| dataset | size (sentences) | MAP/EM | supervised selection<br>CE/DEL1ORTRANS1 | MAP/SA |
|---|---|---|---|---|
| test, $\|\mathbf{x}\| \leq 10$ | 530 | 41.6 | 57.6 | 66.7 |
| $10 < \|\mathbf{x}\| \leq 20$ | 18,101 | 37.6 | 45.6 | 57.3 |
| $10 < \|\mathbf{x}\| \leq 40$ | 39,963 | 35.1 | 42.1 | 53.3 |
| $10 < \|\mathbf{x}\|$ | 41,786 | 34.7 | 41.8 | 52.8 |
| entire WSJ | 49,208 | 35.1 | 42.4 | 53.4 |

Table 7.1: Performance of selected models on unseen test datasets of longer sentences. All sentences of ten or fewer words are filtered out, to avoid overlap with training or development data, except in the last line, which includes all sentences in the Treebank. Note that punctuation was stripped in every case.

Table 7.1 shows the performance of the supervisedly-selected EM, CE, and SA models, trained on sentences of ten or fewer words, perform on datasets containing longer sentences. The first row shows performance on the original test set. The following three rows show performance on datasets of sentences with more than ten words (disjoint from the original training, development, and test datasets), with length cutoffs of 20, 40, and $\infty$. As we would expect, performance falls off as longer sentences are included, because longer sentences are more ambiguous and harder to parse, and also because the models were not trained on long sentences. Nonetheless, the same trend holds: SA outperforms CE (by a greater margin than on the original test set), and CE outperforms EM (with a decreased margin).

We also considered *training* models using longer sentences. Throughout the thesis we have used a training dataset taken from the set of sentences in the Treebank of ten or fewer words, after the removal of punctuation. We trained models using EM and SA (under the same smoothing and annealing conditions as before), with different length cutoffs (20, 40, and 100). Supervised model selection was applied using development data, and test data was used to evaluate performance. The development and test data were taken using the same cutoffs. In each case, the training dataset size was kept roughly the same in the number of *words*, though the number of sentences diminished as larger cutoffs were used.

Results are shown in Table 7.2, which also shows the performance of the parser on the entire WSJ dataset (which includes the training and development data). Consistently, models learned using SA outperform those learned by EM. On the full dataset, the best performance is achieved by SA trained only on short ($|\mathbf{x}| \leq 10$) sentences. Though a

|  | supervised selection |  |  |  |
|---|---|---|---|---|
|  | test set |  | entire WSJ |  |
| maximum length | MAP/EM | MAP/SA | MAP/EM | MAP/SA |
| 10 | 41.6 | 66.7 | 35.1 | 53.4 |
| 20 | 46.6 | 48.4 | 41.6 | 41.7 |
| 40 | 42.6 | 45.9 | 40.4 | 42.4 |
| 100 | 33.7 | 45.2 | 33.6 | 41.7 |

Table 7.2: Performance of models trained on datasets including longer sentences. In the "test set" columns, the models are evaluated on different test sets (with the same maximum length as the training set), so those scores are not directly comparable across rows. The scores in the "entire WSJ" columns include the training and development data, but are comparable.

performance hit is observed when longer sentences are used, performance is steady from 20 to 40 to 100 as a maximum length. Interestingly, EM gets better when a corpus of more diverse sentences is used. It appears, then, that the ten-tag maximum is not the optimal choice for designing a grammar induction corpus, and that different learning methods are more or less robust to corpus filtering. We leave further exploration to future work.

## 7.2 Training Parsers from Induced Tags

Recall that Model A ignores words completely, modeling only sequences of POS tags and the syntactic dependencies among them. In this section we consider the performance of Model A when the tags have been *learned* rather than hand-annotated.

Using the same training set of 5,301 sentences, we applied the part-of-speech induction methods of Section 4.7. Specifically, nine models were trained: three tagging dictionaries (the full-knowledge dictionary, the count $\geq 2$ dictionary, and the count $\geq 3$ dictionary; see Section 4.7.1 for a more full description), each under (a.) MLE/EM training of the base trigram HMM, (b.) CE/DEL1ORTRANS1 training with the same HMM, and (c.) CE/DEL1ORTRANS1 training with the HMM augmented with spelling features. The POS tagging accuracy of these models is shown in Table 7.3; notice that the accuracy tends to be higher than in experiments in the last section, since we are dealing with a simpler dataset. Nonetheless, similar trends are visible (CE outperforms EM, adding spelling features gives further improvement).

Using the nine POS tagging models trained this way, we trained Model A (our de-

| | | tagging dictionary | | |
|---|---|---|---|---|
| model | estimation | all train & dev. | count $\geq 2$ | count $\geq 3$ |
| trigram HMM | MLE/EM | 88.4 | 76.6 | 70.1 |
| trigram HMM | CE/DEL1ORTRANS1 | 92.5 | 82.2 | 77.4 |
| trigram HMM + spelling | CE/DEL1ORTRANS1 | 92.0 | 86.4 | 84.2 |

Table 7.3: Tagging accuracy (all words) of tagging models trained from the sentences in the parsing experiment training dataset, evaluated on the training data. No smoothing or regularization, and therefore no model selection, was applied.

pendency syntax model) using EM and using contrastive estimation. As in earlier experiments, we used three initializers (Zero, K&M, and Local) and a variety of smoothing/regularization settings (the same as before). The results are reported in Table 7.4.

First consider supervised model selection. None of the learned models are quite as good as those learned from gold-standard POS tags using CE. However, the directed dependency accuracy of EM-trained Model A actually improves over *gold-standard* tags, consistently, when the POS tags are learned using CE. When the POS tags are learned using EM, there is slight, predictable degradation. Also striking is that the best-performing EM models are learned using tags selected using CE and a diluted dictionary. Models learned using CE are consistently worse with learned tags than with gold-standard tags, but under some conditions the performance is close. The trends are less clear, but note that only when EM training and the most diluted dictionary (count $\geq 3$) was used did CE training result in a parsing model that underperformed the 41.6% directed accuracy achieved by the MAP/EM baseline. MAP/SA consistently outperforms MAP/EM and usually outperforms CE/DEL1ORTRANS1. For the most part, using CE to induce POS tags results in a more accurate parser than using EM to induce the tags.

Turning to unsupervised model selection, improvements are visible for EM as under supervised model selection, if less pronounced. CE training with induced tags is more volatile however, with unsupervised model selection sometimes choosing models performing in the 20s. Note that when CE is used to train a standard HMM (no spelling features), and then used again for Model A on the HMM's most probable analyses, a five-point improvement is possible over CE on gold-standard tags. Recall that MAP/SA tends to find the best likelihood (the selection criterion) when no annealing is done at all; unsupervised selection of a MAP/SA model led to the same choice as among MAP/EM-trained models. We see here that unsupervised selection (based on likelihood of development data) does not lead to consistent results with MAP/SA.

It may come as a surprise that the use of induced tags could actually improve the accuracy of an unsupervisedly-learned parser over the use of gold-standard tags. There are two possible explanations. The first is that, while the tagging model has learned *incorrect* tags, it has nonetheless learned *useful* category assignments for the words in the training set, and those "pseudo-tags" provide enough information to learn parsing models, in some cases providing better clues about the correct syntactic structure than the correct tags themselves would provide. The second explanation, for which we have tried to control in our experimental design (with model selection over several initializers and several smoothing/regularization settings), but which cannot be ruled out, is the role of chance in accuracy of the local optimum of the likelihood or contrastive objective function being searched.

## 7.3 Evaluation with PARSEVAL

The PARSEVAL scores (Black *et al.*, 1991) have become the community standard for evaluating parser output against a gold standard (e.g., a treebank). The scores compare a hypothesized phrase-structure tree with a gold-standard parse tree. More precisely, they compare the sets of constituents (either labeled or unlabeled) posited in the hypothesized and gold-standard trees. Bracketing precision and recall are typically computed, as well a sentence statistics such as the fraction of perfect matches, the average number of crossing brackets per sentence, and the fraction of sentences $\leq c$ crossing brackets, for different values of $c$ (usually 0 and 2).

We evaluated the English models learned using the various methods presented in this thesis under the PARSEVAL scores. To obtain constituents from the hypothesized dependency attachments, we take the maximal span of every word to be an unlabeled constituent. For the gold standard, we can use the *original* Penn Treebank trees for the test dataset sentences. Note that if we apply head rules to the gold standard (to get dependency structures), then convert back to phrase-structure trees (by taking maximal spans of all heads to be constituents), we arrive at an upper bound on performance—69.9% recall and 100.0% precision—the reason recall is less than 100% is that dependency structures tend to be "flatter" than standard Treebank trees. We also use these flatter "maximal spans" trees

| POS training conditions | | | supervised model selection | | | | | | unsupervised model selection | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MAP/EM accuracy | | CE/D1T1 accuracy | | MAP/SA accuracy | | MAP/EM accuracy | | CE/D1T1 accuracy | | MAP/SA accuracy | |
| dict. | model | estimation | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| (gold-standard tags) | | | 41.6 | 62.2 | 57.6 | 69.0 | 61.8 | 69.4 | 41.6 | 62.2 | 47.3 | 63.6 | 41.6 | 62.2 |
| all train & dev. | trigram HMM | MLE/EM | 41.2 | 61.8 | 54.6 | 68.3 | 61.9 | 66.9 | 41.5 | 62.0 | 20.2 | 60.8 | 45.1 | 63.7 |
| | trigram HMM | CE/D1T1 | 43.7 | 62.6 | 53.8 | 67.4 | 63.6 | 70.8 | 41.0 | 62.3 | 52.6 | 66.7 | 43.9 | 62.7 |
| | trigram HMM + spelling | CE/D1T1 | 41.1 | 62.7 | 48.4 | 64.0 | 57.1 | 67.4 | 41.1 | 62.7 | 42.7 | 61.2 | 39.2 | 60.5 |
| count $\geq 2$ | trigram HMM | MLE/EM | 40.4 | 61.0 | 45.9 | 65.8 | 54.6 | 65.9 | 40.6 | 61.3 | 22.4 | 61.8 | 39.5 | 60.8 |
| | trigram HMM | CE/D1T1 | 42.3 | 62.3 | 50.3 | 64.5 | 55.8 | 65.7 | 42.1 | 63.1 | 42.4 | 62.1 | 42.5 | 62.5 |
| | trigram HMM + spelling | CE/D1T1 | 53.4 | 65.2 | 45.6 | 63.7 | 58.4 | 68.3 | 50.8 | 65.2 | 22.5 | 61.8 | 29.1 | 54.7 |
| count $\geq 3$ | trigram HMM | MLE/EM | 38.9 | 60.7 | 40.4 | 62.4 | 40.5 | 60.1 | 38.6 | 60.7 | 26.5 | 62.9 | 39.8 | 60.6 |
| | trigram HMM | CE/D1T1 | 52.1 | 64.6 | 55.0 | 66.1 | 54.9 | 66.2 | 41.0 | 61.5 | 41.6 | 61.3 | 31.8 | 57.7 |
| | trigram HMM + spelling | CE/D1T1 | 51.8 | 65.5 | 49.9 | 65.5 | 52.4 | 64.4 | 51.8 | 65.5 | 24.9 | 60.0 | 33.1 | 58.5 |

Table 7.4: Accuracy of Model A when trained on learned (rather than gold-standard) POS tags. Supervised (left) and unsupervised (right) model selection were applied over initializers, smoothing/regularization settings, and annealing schedules.

| | Treebank gold standard | | | | | | maximal spans gold standard | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | unlabeled bracketing | | sentence statistics | | | | unlabeled bracketing | | sentence statistics | | | |
| | recall | precision | complete match | ave. crossing brackets | 0 crossing brackets | ≤2 crossing brackets | recall | precision | complete match | ave. crossing brackets | 0 crossing brackets | ≤2 crossing brackets |
| ATTACH-LEFT | 60.6 | 44.3 | 11.7 | 1.84 | 26.4 | 69.3 | 55.3 | 28.3 | 7.7 | 1.59 | 35.5 | 73.8 |
| ATTACH-RIGHT | 17.6 | 12.9 | 8.11 | 3.75 | 11.9 | 30.75 | 20.6 | 10.5 | 7.9 | 2.68 | 19.1 | 46.6 |
| MAP/EM (u-sel. $\lambda, \vec{\theta}^{(0)}$) | 50.1 | 52.3 | 7.9 | 0.55 | 68.9 | 94.5 | 66.8 | 48.9 | 17.0 | 0.39 | 77.0 | 97.0 |
| CE/DEL1 (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 0.9 | 29.8 | 9.8 | 0.05 | 94.9 | 100.0 | 1.3 | 29.8 | 14.0 | 0.04 | 96.4 | 100.0 |
| CE/TRANS1 (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 46.4 | 46.9 | 9.4 | 0.87 | 52.3 | 88.7 | 58.5 | 41.3 | 14.7 | 0.72 | 59.4 | 91.1 |
| CE/DEL1ORTRANS1 (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 37.0 | 34.6 | 8.9 | 1.61 | 32.1 | 72.5 | 45.0 | 29.5 | 12.3 | 1.43 | 36.4 | 77.7 |
| CE/LENGTH (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 49.7 | 55.0 | 9.1 | 0.48 | 71.9 | 96.2 | 62.3 | 48.1 | 17.6 | 0.45 | 73.2 | 96.6 |
| CE/DYNASEARCH (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 46.2 | 47.0 | 9.1 | 0.81 | 55.5 | 90.0 | 59.6 | 42.4 | 14.3 | 0.63 | 65.1 | 92.3 |
| CE/TRANS2 (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 48.4 | 48.3 | 9.4 | 0.81 | 54.2 | 90.6 | 64.8 | 45.2 | 16.8 | 0.62 | 64.9 | 92.6 |
| CE/DEL1ORTRANS2 (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 51.1 | 52.5 | 9.1 | 0.55 | 68.5 | 94.2 | 66.3 | 47.6 | 14.5 | 0.46 | 73.4 | 95.5 |
| CE/DEL1SUBSEQ (u-sel. $\sigma^2, \vec{\theta}^{(0)}$) | 2.4 | 26.6 | 10.2 | 0.19 | 81.7 | 100.0 | 3.1 | 23.7 | 14.5 | 0.14 | 86.6 | 100.0 |
| MLE/DA (u-sel. $\beta_0, \gamma, \vec{\theta}^{(0)}$) | 30.0 | 28.3 | 10.8 | 2.06 | 29.3 | 60.8 | 26.6 | 17.5 | 8.9 | 1.85 | 33.6 | 64.9 |
| MLE/SDA (u-sel. $\beta_0, \gamma, \vec{\theta}^{(0)}$) | 30.1 | 28.4 | 10.8 | 2.06 | 29.1 | 60.4 | 26.8 | 17.7 | 8.9 | 1.84 | 33.6 | 64.5 |
| Model L: | | | | | | | | | | | | |
| MAP/EM (u-sel. $\delta, \lambda, \vec{\theta}^{(0)}$) | 50.1 | 52.3 | 7.9 | 0.55 | 68.9 | 94.5 | 66.8 | 48.9 | 17.0 | 0.39 | 77.0 | 97.0 |
| MAP/SA (u-sel. $\delta_0, \delta_f, \Delta\delta, \lambda, \vec{\theta}^{(0)}$) | 48.2 | 52.3 | 8.3 | 0.46 | 71.1 | 96.2 | 64.4 | 48.1 | 17.0 | 0.44 | 74.9 | 96.4 |
| upper bound (given head rules) | *69.9* | *100.0* | *15.1* | *0.00* | *100.0* | *100.0* | *100.0* | *100.0* | *100.0* | *0.00* | *100.0* | *100.0* |

Table 7.5: PARSEVAL performance of various English models. Unsupervised model selection was applied over relevant hyperparameters (listed in the left column). The left side uses the Penn Treebank trees (with punctuation removed) as the gold standard; the right side uses the flatter, "maximal span" trees constructed from the dependency representation.

as a gold standard for comparing performance of different methods.

Table 7.5 shows PARSEVAL scores for *unsupervisedly* selected models (that is, selection based on the training objective score on unannotated development data). First, note that these are not state-of-the-art phrase-structure induction results on this dataset.[1] Indeed, we would not expect particularly good performance on these scores, given that our dependency models do not directly predict constituent structure. While many methods rival the scores obtained by EM, none are notably better. (The anomalous cases of very low crossing bracket rates and low recall for DEL1 and DEL1SUBSEQ are due to extremely flat hypotheses where nearly every tag is the child of a single root.)

Of course, with annotated development data, supervised model selection can be applied. Table 7.6 shows performance where the model selected from among those trained is the one with the highest PARSEVAL $F_1$-measure. Some improvement is observed over unsupervised model selection (Table 7.5), but none of the alternative estimation methods in the thesis offer striking advantages over EM on this evaluation measure. They tend to achieve higher recall and lower precision than EM, meaning that the trees are less flat than EM's trees.

The fact that many of the alternative methods do not perform notably *worse* than EM is important. Worse PARSEVAL performance would have cast doubt on the claim that our new estimation methods were learning more linguistically apt structures. Of course, our model is not designed to uncover constituent structure. The estimation methods are general and could be applied to a model that *is* designed to capture constituents (see, e.g., Klein and Manning (2004)).

We leave that exploration for future work, with two notes. Klein and Manning's constituent-context model (Klein and Manning, 2002a), which, under EM, is akin to clustering of sequences to find good bracketings, is a poor fit for contrastive estimation. The reason is that the parameters in the model correspond to subsequences of the observed training data. To model neighborhood sequences, as in CE, many more parameters will be needed, since neighborhood sequences tend to include subsequences that are never observed. Indeed, the CE objective can always be maximized by simply driving up the weight of the feature corresponding to the whole observed sequence for each training sen-

[1]Klein and Manning (2004) report 88.0% unlabeled recall and 69.3% unlabeled precision on the full WSJ10 dataset, from which our training, test, and development data are drawn, using a product model that includes the dependency model we use here (Model A) and a model of constituent structure, trained using EM. Using only Model A, they report 59.2% recall and 46.6% precision, which is comparable in $F_1$-measure to our result.

tence. To work with CE, this model must be reparameterized, perhaps using a factored sequence model.

Second, our structural locality bias (used in Model L) is not appropriate for models without heads. For an unheaded phrase-structure model, different notions of structural bias need to be developed. One possibility is the segmentation bias used in Model S; another is a parameter that penalizes balanced trees.

To sum up this section, we have evaluated our unsupervisedly-selected models using PARSEVAL, a standard measure of constituent-based parser performance, and carried out supervised model selection using PARSEVAL. The results do not show a clear superiority of any method over EM, though CE (with some neighborhoods) and SA do rival EM.

## 7.4 Cross-Method Comparison with Random Initializers

Using 100 random initializers, we compared the improvement each of our novel estimation methods made (over the initializer) to the improvement made by EM in Sections 4.6.5, 5.3.2, 6.1.4, and 6.2.2.

For completeness, we show here the directed accuracy performance of each of those four methods (MLE/EM/Model A, CE/DEL1ORTRANS1/Model A, MLE/SDA/Model A, and MLE/SA/Model L) compared to each of the others, on the 100 initializers as well as Zero, K&M, and Local. (MLE/Model L is also shown in comparison to MLE/SA/Model L.) Figure 7.1 plots each pair of these. Unlike the earlier plots, the absolute directed accuracy (not the improvement) is shown for each pair. The only notable trends are that CE/DEL1ORTRANS1 tends to be worse than all the other methods, and Model L with EM and SA tend to be more closely coupled than any other pair.

## 7.5 Combining Structural Annealing and Contrastive Estimation

Recall that contrastive estimation (Chapter 4) is a way of redefining the objective function used for learning. A contrastive objective function takes into account, for each training example $\mathbf{x}^t$, a set of implicitly negative instances $\mathcal{N}(\mathbf{x}^t) \subseteq \Sigma^*$.

Structural annealing, presented in Chapter 6, is a method that gradually changes the objective function over time, to guide the learner to a more accurate model. Our applications of structural annealing have involved the addition of a single feature to Model A

| | Treebank gold standard | | | | | | maximal spans gold standard | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | unlabeled bracketing | | sentence statistics | | | | unlabeled bracketing | | sentence statistics | | | |
| | recall | precision | complete match | ave. crossing brackets | 0 crossing brackets | $\leq$2 crossing brackets | recall | precision | complete match | ave. crossing brackets | 0 crossing brackets | $\leq$2 crossing brackets |
| ATTACH-LEFT | 60.6 | 44.3 | 11.7 | 1.84 | 26.4 | 69.3 | 55.3 | 28.3 | 7.7 | 1.59 | 35.5 | 73.8 |
| ATTACH-RIGHT | 17.6 | 12.9 | 8.11 | 3.75 | 11.9 | 30.75 | 20.6 | 10.5 | 7.9 | 2.68 | 19.1 | 46.6 |
| MAP/EM (s-sel. $\lambda, \vec{\theta}^{(0)}$) | 51.0 | 53.2 | 8.5 | 0.51 | 71.1 | 95.1 | 66.3 | 48.7 | 17.0 | 0.38 | 76.8 | 97.2 |
| CE (s-sel. $\sigma^2, \vec{\theta}^{(0)}, \mathcal{N}$) | 53.6 | 51.3 | 9.1 | 0.75 | 59.3 | 90.2 | 77.1 | 48.8 | 17.0 | 0.62 | 68.9 | 90.2 |
| MLE/DA (s-sel. $\beta_0, \gamma, \vec{\theta}^{(0)}$) | 54.6 | 49.0 | 13.2 | 1.27 | 37.2 | 82.6 | 49.5 | 31.0 | 9.3 | 1.15 | 43.2 | 84.2 |
| MLE/SDA (s-sel. $\beta_0, \gamma, \vec{\theta}^{(0)}$) | 56.0 | 48.2 | 13.2 | 1.37 | 34.7 | 80.8 | 70.5 | 50.8 | 17.4 | 0.37 | 77.6 | 96.8 |
| Model L: | | | | | | | | | | | | |
| MAP/EM (s-sel. $\delta, \lambda, \vec{\theta}^{(0)}$) | 60.7 | 46.2 | 11.9 | 1.62 | 29.1 | 75.3 | 77.7 | 50.9 | 17.7 | 0.56 | 70.6 | 92.5 |
| MAP/SA (s-sel. $\delta_0, \delta_f, \Delta\delta, \lambda, \vec{\theta}^{(0)}$) | 55.3 | 53.8 | 9.1 | 0.70 | 60.2 | 93.0 | 77.5 | 51.3 | 19.8 | 0.56 | 70.2 | 93.2 |
| upper bound (given head rules) | *100.0* | *100.0* | *100.0* | *0.00* | *100.0* | *100.0* | *69.9* | *100.0* | *15.1* | *0.00* | *100.0* | *100.0* |

Table 7.6: PARSEVAL performance of various English models. Supervised model selection was applied over relevant hyperparameters (listed in the left column). The left side uses the Penn Treebank trees (with punctuation removed) as the gold standard; the right side uses the flatter, "maximal span" trees constructed from the dependency representation. The CE model selected on the left had been trained using TRANS1; the CE model selected on the right had been trained using TRANS2.

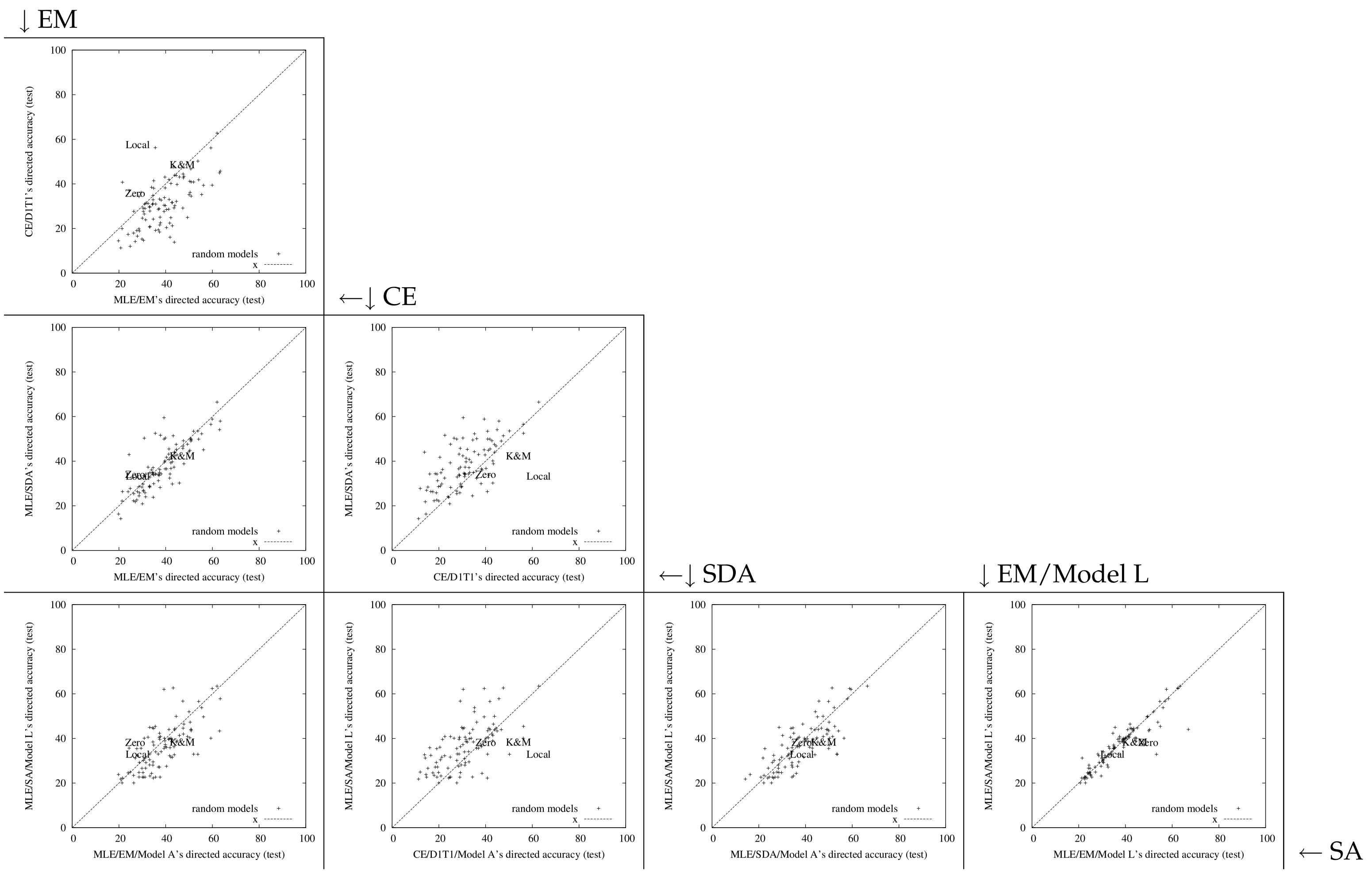


Figure 7.1: Cross-method comparison of directed accuracy on the test dataset. The methods compared are unsmoothed MLE/EM, unregularized CE/DEL1ORTRANS1, unsmoothed MLE/SDA ($\beta = 0.01, \gamma = 1.5$), unsmoothed MLE/Model L ($\delta = -0.4$), and unsmoothed SA/Model L ($\delta_0 = -0.6, \Delta\delta = 0.1, \delta_f = 0$).

(giving Models L and S), and manipulating that feature's weight extrinsic to the training of the other parameters. So far we have used only maximum likelihood to estimate those parameters.

The two techniques—contrastive estimation and structural annealing—are orthogonal. In this section, we experiment with a combination of the two, finding that Model L and CE together outperform MAP estimation (on Model L) and Model A (with CE). *Annealing* the structural bias $\delta$ improves this result further, essentially matching (but not surpassing) the performance of structurally annealed EM training.

### 7.5.1 CE and Model L

Experimental results for Model L (Section 6.1) trained with CE/DEL1ORTRANS1 are presented in Table 7.7. The table shows how performance of unregularized CE varies with a fixed value of $\delta$, for the three initializers. Notice that, for any of the initializers, substantial improvement over Model A is available by setting $\delta$ to an appropriate negative value.

Supervised model selection was applied, and is shown to outperform Model A trained with CE (the $\delta = 0$ case). Selecting (supervisedly) across *regularized* CE/DEL1ORTRANS1 trials gives a performance boost over MAP/EM training on the same model. Figure 7.2 shows, for several initializer/regularization settings, how performance of CE varies with $\delta$.

Next we apply structural annealing to Model L during contrastive estimation training with the DEL1ORTRANS1 neighborhood function. Selecting across $\vec{\theta}^{(0)}$, $\sigma^2$, $\beta_0$, and $\beta_f$:

- Unsupervised selection ($\vec{\theta}^{(0)}$ = Zero, $\sigma^2 = 1$, $\beta_0 = -0.2$, $\beta_f = 0$): 41.0% directed, 63.8% undirected.
- Supervised selection ($\vec{\theta}^{(0)}$ = K&M, $\sigma^2 = 10^{-2/3}$, $\beta_0 = -0.2 \to 0$): 65.5% directed, 72.3% undirected. These results are not signficantly different (under a sign test) from those obtained using structural annealing with EM (instead of CE; 66.7% and 73.1%).

### 7.5.2 CE and Model S

Our experiments found that combining Model S with CE/DEL1ORTRANS1 training performed very badly, hypothesizing extremely local parse trees. Typically over 90% of dependencies were of length 1 and pointed in the same direction (compared with around 60%

| | Zero | | | | K&M | | | | Local | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | test accuracy | | | | test accuracy | | | | test accuracy | | |
| $\delta$ | evaluations | $f$ | directed | undirected | evaluations | $f$ | directed | undirected | evaluations | $f$ | directed | undirected | |
| MAP/EM, s-sel.: $\vec{\theta}^{(0)}$ = Zero, $\delta = -0.6$, $\lambda = 10$ | | | | | | | | | 44 | -9.40 | 61.8 | 69.4 | $*$ |
| -1.0 | 105 | -3.62 | 59.5 | 69.2 | 105 | -3.80 | 47.6 | 60.5 | 56 | -3.65 | 44.1 | 63.5 | |
| -0.8 | 104 | -3.56 | 60.2 | 69.2 | 105 | -3.70 | 46.5 | 60.0 | 103 | -3.56 | 44.7 | 64.1 | |
| -0.6 | 104 | -3.50 | 56.0 | 66.3 | 106 | -3.57 | 61.1 | 69.1 | 104 | -3.62 | 61.7 | 69.8 | |
| -0.4 | 104 | -3.45 | 57.0 | 68.4 | 106 | -3.54 | 59.0 | 68.0 | 101 | -3.53 | 58.4 | 67.4 | |
| -0.2 | 108 | -3.47 | 45.8 | 65.6 | 105 | -3.50 | 60.5 | 70.5 | 106 | -3.48 | 59.7 | 69.6 | |
| (≡Model A) 0.0 | 103 | -3.79 | 35.8 | 62.2 | 103 | -3.90 | 48.6 | 64.9 | 104 | -3.83 | 57.6 | 69.0 | $**$ |
| 0.2 | 106 | -3.64 | 23.9 | 54.1 | 106 | -3.52 | 51.1 | 66.3 | 104 | -3.55 | 27.9 | 56.1 | |
| unsupervised model selection: $\vec{\theta}^{(0)}$ = Local, $\delta = -0.2$ | | | | | | | | | 106 | -3.48 | 59.7 | 69.6 | |
| supervised model selection: $\vec{\theta}^{(0)}$ = Local, $\delta = -0.6$ | | | | | | | | | 104 | -3.62 | 61.7 | 69.8 | |
| with regularization: | | | | | | | | | | | | | |
| unsupervised model selection: $\vec{\theta}^{(0)}$ = Zero, $\delta = -0.8$, $\sigma^2 = 1$ | | | | | | | | | 94 | -3.70 | 57.9 | 68.2 | |
| supervised model selection: $\vec{\theta}^{(0)}$ = K&M, $\delta = -0.4$, $\sigma^2 = 10^{-1/3}$ | | | | | | | | | 108 | -3.80 | 63.5 | 71.5 | |

Table 7.7: Model L trained using CE/DEL1ORTRANS1 with different initializers and biases $\delta$. Note that $\delta = 0$ is essentially equivalent to Model A. No regularization was done here, except for the last two lines (see text). $f$ is the objective value (log of the contrastive likelihood) on the training data, under Model A (i.e., calculated with $\delta = 0$). Model L trained with regularized CE improves over Model L trained by MAP/EM (marked $*$) and over Model A trained with CE/DEL1ORTRANS1 (marked $**$).

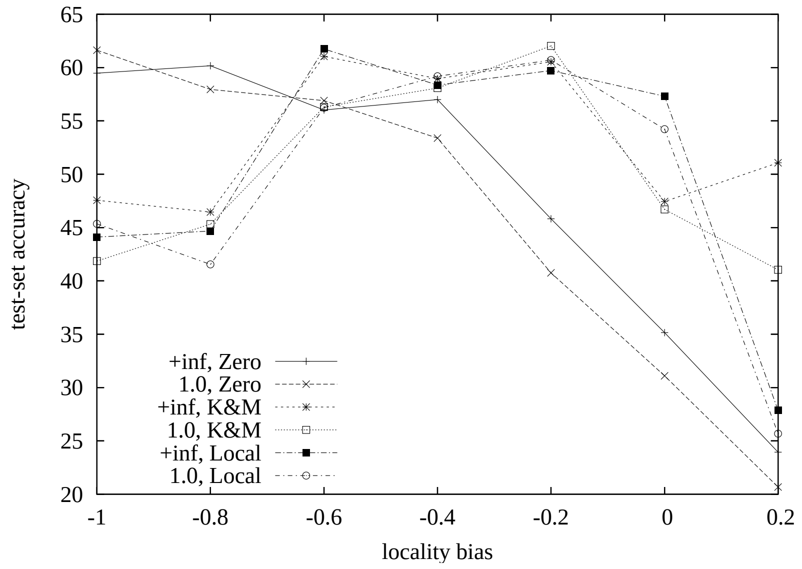


Figure 7.2: Test-set directed attachment accuracy of Model L trained by CE/DEL1ORTRANS1, at different, fixed values of $\delta$. Each curve corresponds to a different value of $\sigma^2$ and $\vec{\theta}^{(0)}$. Notice that the highest point on most curves is not at $\delta = 0$.

in the gold standard data). To understand why, consider that the CE goal is to maximize the score of a sentence *and* all its segmentations while minimizing the scores of neighborhood sentences and their segmentations. An $n$-gram model can accomplish this, since the same $n$-grams are present in all segmentations of $\mathbf{x}$, and (some) different $n$-grams appear in $\mathcal{N}(\mathbf{x})$ (for LENGTH and DEL1ORTRANS1). A bigram-like model that favors monotone branching, then, is not a bad choice for a CE learner on Model S.

Why doesn't CE on Model A resort to $n$-gram-like models? Inspection of models trained that way with transposition-based neighborhoods TRANS1 and DEL1ORTRANS1 *did* have high rates of length-1 dependencies, while the poorly-performing DEL1 models found *low* length-1 rates. This suggests that a bias toward locality ("$n$-gram-ness") is built into the former neighborhoods, and may partly explain why CE works when it does. We achieved a similar locality bias indirectly in the likelihood framework when we broadened the hypothesis space (moving from Model A to Model S), but doing so under CE *over*-focuses the model on local structures.

## 7.6 Summary

In this chapter we evaluated our learning methods in a few additional ways, including variations on the training data and the evaluation criteria. We found the general trend to be similar when training on datasets with longer sentences or imperfect tags as to earlier experiments: SA is superior to CE, and both are superior to EM. We found that the constituent trees implied by the learned dependency parses are less clearly differentiated in quality under the standard PARSEVAL measures. We compared five methods' directed accuracy on 100 random initializers and found only one trend of note, that CE/DEL1ORTRANS1 usually performs worse than the other methods, given a random initializer.

We then combined structural bias and structural annealing, with contrastive estimation, showing the same pattern of improvements seen when these were combined with MLE. Annealed contrastive estimation performs approximately as well as annealed MLE for Model L. The best *un*annealed results (requiring only a single round of numerical optimization) were achieved by CE with a structural locality bias.

# Chapter 8

# Multilingual Evaluation

*If it had been possible to build the Tower of Babel without ascending it, the work would have been permitted.*

—Franz Kafka (1883–1924)

*I speak two languages: Body and English.*

—Mae West (1892–1980)

This chapter applies techniques from Chapters 4, 5, and 6 to data in five additional languages: German, Bulgarian, Mandarin, Turkish, and Portuguese.[1] Many of these experiments were described in Smith and Eisner (2006). Because this chapter is focused on the breadth of applicability of the estimation methods, we will suppress many details about selected models for readability.

## 8.1 Datasets

Following the usual conventions (Klein and Manning, 2002a), our experiments use treebank POS sequences of length $\leq 10$, stripped of words and punctuation.

Our training datasets are as follows. The part-of-speech tagsets are given in Appendix B.

- 8,227 **German** sentences from the TIGER Treebank (Brants, Dipper, Hansen, Lezius,

[1]One reviewer of the thesis questioned the absence of Czech in these experiments. Like English and German, Czech might be counted as a "high-resource" language for which the state-of-the-art in parsing is already quite good (McDonald *et al.*, 2005b). We opted instead to choose Bulgarian as a representative of the Slavic group, since there is more potential benefit to the development of Bulgarian NLP tools.

and Smith, 2002) (51 tag types, unigram tag distribution in training data has 4.01 bits of entropy),

- 5,301 **English** sentences from the WSJ Penn Treebank (Marcus *et al.*, 1993) (35 tag types, 4.12 bits),
- 4,929 **Bulgarian** sentences from the BulTreeBank (Simov, Popova, and Osenova, 2002; Simov and Osenova, 2003; Simov, Osenova, Simov, and Kouylekov, 2004a) (48 tag types, 4.15 bits),
- 2,775 **Mandarin** sentences from the Penn Chinese Treebank (Xue, Xia, Chiou, and Palmer, 2004) (33 tag types, 3.18 bits),
- 2,576 **Turkish** sentences from the METU-Sabancı Treebank (Atalay, Oflazer, and Say, 2003; Oflazer, Say, Hakkani-Tür, and Tür, 2003) (27 tag types, 3.06 bits), and
- 1,676 **Portuguese** sentences from the Bosque portion of the Floresta Sintá(c)tica Treebank (Afonso, Bick, Haber, and Santos, 2002) (19 tag types, 3.39 bits).

The Bulgarian, Turkish, and Portuguese datasets come from the CoNLL-X shared task Buchholz and Marsi (2006); we thank the organizers.

Recall from Section 2.5 that precision and recall measures can be defined when comparing a hypothesized dependency tree $\mathbf{y}$ to a gold standard tree $\mathbf{y}^*$. These measures are equivalent (and equal to the accuracy measures used so far) if both trees are fully connected, with $n - 1$ dependency links between the $n$ words. In some of the gold standard corpora used in this chapter, some trees are not fully connected and have multiple roots. For this reason, we report the $F_1$-accuracy (harmonic mean between precision and recall). The test set consists of around 500 sentences (in each language), as does the development set used for model selection. All reported results are on the unseen test data; the development data was used for model selection. As in previous chapters, supervised and unsupervised model selection were applied.

## 8.2 Baselines and EM (Chapter 3)

Table 8.1 shows the performance of the ATTACH-LEFT and ATTACH-RIGHT baselines, MLE/EM, and MAP/EM. The MLE models are selected from among the three initializers

| | German accuracy | | English accuracy | | Bulgarian accuracy | | Mandarin accuracy | | Turkish accuracy | | Portuguese accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | **62.1** | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | **68.6** | 36.2 | **65.7** |
| ATTACH-RIGHT | 47.0 | 55.2 | **39.5** | **62.1** | 23.8 | 61.0 | 42.9 | 56.1 | **61.8** | 68.3 | 29.5 | **65.7** |
| MLE (u-sel. $\vec{\theta}^{(0)}$) | –* | –* | 22.7 | 59.0 | –* | –* | 40.4 | 56.5 | 31.9 | 58.3 | **42.5** | 64.3 |
| MLE (s-sel. $\vec{\theta}^{(0)}$) | 39.6 | 60.9 | **41.6** | **62.2** | 43.2 | 62.9 | 48.5 | 60.0 | 42.2 | 58.7 | **42.5** | 64.3 |
| MLE (oracle $\vec{\theta}^{(0)}$) | *39.6* | *60.9* | *41.6* | *62.2* | *43.2* | *62.9* | *48.5* | *60.0* | *42.2* | *58.7* | *42.9* | *64.4* |
| MAP (u-sel. $\vec{\theta}^{(0)}, \lambda$) | 19.8 | 55.2 | **41.6** | **62.2** | 44.6 | **63.1** | 37.2 | 56.1 | 41.2 | 57.8 | 37.4 | 62.2 |
| MAP (s-sel. $\vec{\theta}^{(0)}, \lambda$) | **54.4** | **71.9** | **41.6** | **62.2** | **45.6** | **63.6** | **50.0** | **60.9** | 48.0 | 59.1 | **42.3** | 64.1 |
| MAP (oracle $\vec{\theta}^{(0)}, \lambda$) | *54.4* | *71.9* | *42.0* | *62.3* | *45.6* | *63.6* | *50.0* | *60.9* | *51.4* | *63.8* | *43.0* | *64.4* |

Table 8.1: Performance of Model A trained with EM on six languages. Unsupervised ("u-sel.") and supervised ("s-sel.") model selection were applied; the performance of oracle model selection is also shown. Boldface marks the best performance in each column, and any trial in the column not shown to be significantly worse than it under a sign test ($p \not< 0.05$). *Unsupervised model selection could not be applied, because all of these (unsmoothed) models scored $-\infty$ on the development data.

used before (Zero, K&M, Local); the MAP models are selected from initializers $\times$ seven values of $\lambda$: $\{0, 10^{-2/3}, 10^{-1/3}, 1, 10^{1/3}, 10^{2/3}, 10\}$. First note the high performance of the untrained baselines for Turkish; nothing outperforms ATTACH-RIGHT. Portuguese undirected accuracy is also best with the baselines, and (as noted earlier) no English model trained this way significantly outperforms ATTACH-RIGHT. The second important thing to notice is that supervised model selection tends to perform very close to the oracle model. Further, unsupervised model selection usually performs much worse (note, especially, German). The usefulness of even a small quantity of annotated data for model selection is evident across languages.

For five languages, the best performance among these trials is by MAP/EM with supervised model selection. In three cases, unsupervised model selection still outperformed the baseline.

## 8.3 Contrastive Estimation (Chapter 4)

Regularized contrastive estimation (CE; Chapter 4) was applied to all six datasets, using five neighborhoods and with $\sigma^2 \in \{10^{-2/3}, 10^{-1/3}, 1, 10^{1/3}, 10^{2/3}, 10, +\infty\}$. Results are shown in Table 8.2 (oracle model selection), Table 8.3 (supervised selection), and Table 8.4 (unsupervised selection).

The oracle results (Table 8.2) show that CE can, in principle, achieve better results than EM on four languages (the exceptions are Bulgarian and Mandarin). This holds up well under supervised model selection (Table 8.3). Note that on German, directed accuracy significantly improved, but not undirected accuracy (if we apply supervised selection on undirected accuracy, CE achieves 72.6% against MAP/EM's 71.9%, which is not quite significantly better under a sign test). On Turkish, CE shows large improvements over MAP/EM, but still not over the ATTACH-RIGHT baseline. Large, significant improvements were obtained for Portuguese (with DEL1ORTRANS1) but CE did not match EM on Mandarin or Bulgarian.

Turning to *unsupervised* model selection, although performance generally decreases, CE offers more consistent improvement across languages over MAP/EM (also with unsupervised selection), provided that a good neighborhood is used. Bulgarian remains hopeless, but on every other language some neighborhood outperforms MAP/EM, in most cases significantly.

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| MAP (oracle $\vec{\theta}^{(0)}, \lambda$) | 54.4 | 71.9 | 42.0 | 62.3 | 45.6 | 63.6 | 50.0 | 60.9 | 51.4 | 63.8 | 43.0 | 64.4 |
| CE (oracle $\vec{\theta}^{(0)}, \sigma^*$) | 63.1 | 66.5 | 57.6 | 69.0 | 43.1 | 62.5 | 48.3 | 58.8 | 59.0 | 64.9 | 71.8 | 78.4 |
| | (DEL1ORTRANS1) | | (DEL1ORTRANS1) | | (DEL1ORTRANS1) | | (DYNASEARCH) | | (LENGTH) | | (DEL1ORTRANS1) | |

Table 8.2: Performance of Model A trained with CE on six languages. Oracle model selection was applied to choose the neighborhood $\mathcal{N}$, initializer $\vec{\theta}^{(0)}$, and regularizer $\sigma^2$.

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| $\Sigma^*$ (MAP/EM) | 54.4 | 71.9 | 41.6 | 62.2 | 45.6 | 63.6 | 50.0 | 60.9 | 48.0 | 59.1 | 42.3 | 64.1 |
| DEL1 | 34.4 | 49.3 | 39.7 | 53.5 | 17.7 | 33.8 | 43.4 | 49.8 | 42.1 | 45.1 | 28.0 | 43.1 |
| TRANS1 | 45.6 | 59.0 | 41.2 | 62.5 | 40.1 | 57.9 | 41.1 | 56.1 | 47.2 | **63.4** | 35.9 | **65.8** |
| DEL1ORTRANS1 | **63.4** | 66.5 | **57.6** | **69.0** | 40.5 | 61.5 | 41.1 | 56.9 | **58.2** | **66.4** | **71.8** | **78.4** |
| LENGTH | **57.3** | 65.1 | **45.5** | **64.9** | 38.3 | 63.4 | 26.2 | 44.9 | **59.0** | **64.9** | 33.6 | 65.3 |
| DYNASEARCH | 45.7 | 58.6 | **47.6** | **65.3** | 34.0 | 58.0 | 47.9 | 60.6 | 44.9 | **62.7** | 40.9 | 64.4 |
| s-sel. ($\mathcal{N}$) | **63.4** | 66.5 | **57.6** | **69.0** | 40.5 | 61.5 | 41.1 | 56.1 | **59.0** | **64.9** | **71.8** | **78.4** |

Table 8.3: Performance of Model A trained with CE on six languages. Supervised model selection was applied to choose the initializer $\vec{\theta}^{(0)}$ and regularizer $\sigma^2$. Boldface marks trials significantly better than MAP/EM (third line) under a sign test ($p < 0.05$). Note that, on Turkish, CE still does not outperform ATTACH-RIGHT. The last line selects across neighborhoods (supervisedly).

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| $\Sigma^*$ (MAP/EM) | 19.8 | 55.2 | 41.6 | 62.2 | 44.6 | 63.1 | 37.2 | 56.1 | 41.2 | 57.8 | 37.4 | 62.2 |
| DEL1 | **26.5** | 43.7 | 18.3 | 33.9 | 13.1 | 31.5 | 25.6 | 41.4 | 41.8 | 45.2 | 39.9 | **67.2** |
| TRANS1 | 17.9 | 53.0 | 29.4 | 57.8 | 23.8 | 61.0 | 22.7 | 56.3 | 27.7 | **59.4** | 36.0 | **65.5** |
| DEL1ORTRANS1 | **59.3** | **72.6** | **47.3** | **63.6** | 24.2 | 60.0 | 22.6 | 58.2 | **46.5** | **62.9** | 36.0 | **65.4** |
| LENGTH | **49.2** | **64.1** | **45.5** | **64.9** | 27.0 | 60.1 | 16.5 | 43.4 | 34.4 | 57.6 | 31.9 | 59.4 |
| DYNASEARCH | 16.0 | 53.0 | 39.7 | 61.9 | 23.8 | 61.0 | **48.3** | **58.8** | 44.9 | **62.7** | 37.9 | 62.3 |

Table 8.4: Performance of Model A trained with CE on six languages. Unsupervised model selection was applied to choose the initializer $\vec{\theta}^{(0)}$ and regularizer $\sigma^2$. Boldface marks trials significantly better than MAP/EM (third line) under a sign test ($p < 0.05$).

## 8.4 Skewed Deterministic Annealing (Chapter 5)

We applied skewed deterministic annealing (SDA; Section 5.3) to the six datasets. Annealing schedules with $\beta_0 \in \{0.1, 0.01, 0.001\}$ and $\gamma \in [1.25, 2.5]$ were tested. Model selection across initializers $\vec{\theta}^{(0)}$ was applied; no smoothing was applied. Table 8.5 presents the results.

As a general trend, we see that SDA sometimes improves over MLE/EM and (with the exception of German unlabeled $F_1$-accuracy) sometimes improves over MAP/EM. The gains are generally not overwhelming, however, and in cases where EM did not beat an untrained baseline, neither did SDA.

## 8.5 Structural Annealing (Chapter 6)

Experimental results of Model L, trained with values of $\delta \in [-1, 0.2]$, are shown in Table 8.6. With supervised model selection, across languages we see that a nonzero $\delta$ improves directed accuracy relative to Model A ($\delta = 0$). German improves most with a *positive* $\delta$. Here we finally see Turkish results that catch up to the ATTACH-RIGHT baseline, with $\delta = -0.2$. Unsupervised model selection is less successful with this method, because local optima on the Model L likelihood surface tend not to score well on the unsupervised selection criterion (Model A likelihood of development data).

Table 8.7 shows results with structural annealing on $\delta$, for different starting $\delta_0$. $\Delta\delta$ was fixed at 0.1 for these experiments, and supervised model selection was applied for the stopping $\delta_f$, smoothing $\lambda$, and the initializer. Consistent improvements are seen for languages other than Turkish and Portuguese, where annealing was approximately as accurate as the unannealed Model L.

Unsupervised model selection was applied, as well; see Table 8.8. Although three languages show improvement over MAP/EM (with unsupervised selection), and four languages show improvement for some value of $\delta_0$ (fixed and not selected), these results are less impressive. The explanation is, as with fixed $\delta$: the likelihood of development data (or training data) under Model A is not expected to be high for models trained using Model L with $\delta \neq 0$.

Table 8.9 shows some additional error statistics: the macro-averaged path length for the (supervisedly) selected models in each language, and the mean KL divergence to the

| | | German test accuracy | | | English test accuracy | | | Bulgarian test accuracy | | | Mandarin test accuracy | | | Turkish test accuracy | | | Portuguese test accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\vec{\theta}^{(0)}$ | directed | undirected | $\vec{\theta}^{(0)}$ | directed | undirected | $\vec{\theta}^{(0)}$ | directed | undirected | $\vec{\theta}^{(0)}$ | directed | undirected | $\vec{\theta}^{(0)}$ | directed | undirected | $\vec{\theta}^{(0)}$ | directed | undirected |
| ATTACH-LEFT | | | 8.2 | 59.1 | | 22.6 | 62.1 | | 37.2 | 61.0 | | 13.1 | 56.1 | | 6.6 | 68.6 | | 36.2 | 65.7 |
| ATTACH-RIGHT | | | 47.0 | 55.2 | | 39.5 | 62.1 | | 23.8 | 61.0 | | 42.9 | 56.1 | | 61.8 | 68.3 | | 29.5 | 65.7 |
| $\beta_0$ | $\gamma$ | | | | | | | | | | | | | | | | | | |
| 0.1 | 2.5 | Zero | **58.4** | **66.7** | K&M | **44.6** | 61.2 | K&M | 38.0 | 61.3 | K&M | 30.5 | 51.9 | K&M | 35.3 | 55.5 | Local | 35.9 | 62.3 |
| 0.1 | 2.25 | K&M | 32.7 | 58.1 | K&M | 40.7 | 57.4 | K&M | 33.7 | 59.6 | K&M | 29.7 | 52.1 | Zero | 30.0 | 53.8 | K&M | 31.0 | 58.3 |
| 0.1 | 2.0 | Zero | **55.1** | **64.3** | K&M | **46.7** | **64.3** | K&M | **46.4** | **65.8** | Zero | 44.0 | 59.2 | K&M | **50.1** | **60.5** | Zero | **43.7** | 64.3 |
| 0.1 | 1.75 | Zero | **56.7** | **64.7** | K&M | **42.2** | **62.6** | K&M | **45.0** | **64.3** | Local | **50.5** | **61.5** | K&M | **50.3** | **60.2** | Zero | **43.6** | **64.9** |
| 0.1 | 1.5 | Zero | **55.9** | **65.2** | K&M | **45.4** | 62.7 | K&M | **46.3** | **65.7** | Local | 42.3 | 57.5 | K&M | **48.3** | **59.9** | Zero | **43.3** | 63.8 |
| 0.1 | 1.25 | Zero | **57.0** | **64.6** | K&M | **42.4** | **62.7** | K&M | 42.7 | 63.4 | Local | **50.7** | **61.6** | K&M | **53.0** | **61.8** | Zero | **43.4** | 64.8 |
| 0.01 | 2.5 | Zero | **55.0** | **63.6** | K&M | 41.7 | 62.0 | K&M | **44.4** | **64.1** | Local | 48.6 | 59.9 | K&M | 42.4 | 58.8 | Zero | **43.3** | 64.6 |
| 0.01 | 2.25 | Zero | **54.9** | **64.9** | K&M | 43.1 | 60.0 | K&M | 35.2 | 60.1 | K&M | 30.6 | 52.4 | Local | 31.0 | 55.5 | Local | 34.2 | 60.4 |
| 0.01 | 2.0 | Zero | **59.4** | **67.0** | K&M | **44.1** | 61.5 | K&M | 39.0 | 61.8 | Local | 36.4 | 53.9 | K&M | 38.1 | 55.9 | K&M | 38.6 | 63.8 |
| 0.01 | 1.75 | Zero | **55.7** | **64.2** | K&M | 41.9 | 62.3 | K&M | **45.4** | **64.6** | Local | **49.1** | **60.7** | K&M | **51.7** | **61.6** | Zero | **43.7** | 64.7 |
| 0.01 | 1.5 | Zero | **56.3** | **64.7** | K&M | 42.1 | 62.4 | K&M | **45.6** | **64.9** | Local | **50.3** | **61.2** | K&M | **53.2** | **62.7** | Zero | **43.7** | 64.4 |
| 0.01 | 1.25 | Zero | **56.4** | **64.7** | K&M | 42.0 | 62.3 | K&M | 40.1 | 62.0 | Local | 47.2 | **61.0** | K&M | **53.6** | **62.8** | Zero | **43.9** | 64.0 |
| 0.001 | 2.5 | Zero | **55.8** | **65.6** | K&M | **44.3** | 61.3 | K&M | 36.9 | 60.9 | Local | 34.8 | 52.9 | Local | 32.5 | 55.3 | K&M | 36.6 | 62.8 |
| 0.001 | 2.25 | Zero | **59.5** | **67.1** | K&M | **44.2** | 61.6 | K&M | 40.3 | 62.5 | Local | 37.3 | 53.8 | K&M | 39.0 | 56.1 | Local | 36.8 | 63.0 |
| 0.001 | 2.0 | Zero | 34.9 | 60.6 | K&M | 41.3 | 57.9 | K&M | 33.8 | 59.6 | K&M | 29.6 | 52.1 | Zero | 29.9 | 53.9 | K&M | 31.1 | 58.6 |
| 0.001 | 1.75 | Zero | **55.3** | **64.2** | K&M | 41.5 | 62.0 | K&M | **46.4** | **65.5** | Zero | 45.3 | 59.7 | K&M | **51.5** | **62.0** | Zero | **43.6** | 64.0 |
| 0.001 | 1.5 | Zero | **58.3** | **64.1** | K&M | **42.2** | 62.4 | K&M | 37.5 | 59.9 | Local | **50.1** | **61.1** | K&M | **42.9** | 58.9 | Zero | **43.2** | 64.6 |
| 0.001 | 1.25 | Zero | **60.0** | **66.5** | K&M | 41.7 | 62.3 | K&M | 40.4 | 62.3 | Local | 47.6 | 59.1 | K&M | **51.1** | **61.3** | Zero | **43.7** | 64.2 |

Table 8.5: Performance of Model A trained with SDA on six languages. SDA training was without smoothing, so the most direct comparison is with MLE (third line). Supervised model selection was applied to choose the initializer $\vec{\theta}^{(0)}$. Boldface marks trials significantly better than MLE/EM (third line) under a sign test ($p < 0.05$). Note that we did not use smoothing in the SDA trials, so direct comparison to MAP/EM (fourth line) is not fair.

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| MAP/Model A (s-sel.) | 54.4 | 71.9 | 41.6 | 62.2 | 45.6 | 63.6 | 50.0 | 60.9 | 48.0 | 59.1 | 42.3 | 64.1 |
| $\delta =$ | | | | | | | | | | | | |
| -1.0 | 50.1 | 57.1 | 37.7 | 59.0 | 47.0 | **65.2** | 49.4 | 58.2 | **61.8** | **68.5** | 36.9 | **65.6** |
| -0.8 | 50.5 | 57.3 | 38.2 | 59.1 | 47.7 | 63.8 | 47.3 | 57.6 | **62.3** | **68.5** | 38.0 | **66.8** |
| -0.6 | **60.7** | 62.9 | **61.8** | **69.4** | **48.2** | 64.1 | 43.1 | 56.1 | **61.0** | **67.9** | **46.7** | **67.6** |
| -0.4 | **58.2** | 62.2 | **50.5** | **64.6** | **49.1** | **64.9** | 51.1 | 59.7 | **60.9** | **67.8** | **50.4** | **69.4** |
| -0.2 | **58.6** | 71.1 | **49.6** | **66.1** | **49.2** | 64.3 | **54.3** | 62.1 | **62.3** | **67.5** | **46.3** | **66.9** |
| 0.0 | 54.4 | 71.9 | 41.6 | 62.3 | 45.6 | 63.6 | 50.0 | 60.8 | 48.0 | 59.1 | 42.4 | 64.2 |
| 0.2 | **61.3** | 70.7 | **44.9** | **64.4** | 32.1 | 59.9 | 46.6 | 59.2 | **56.2** | **63.8** | 40.5 | **66.3** |
| supervised selection | **61.3** | 70.7 | **61.8** | **69.4** | **49.2** | 64.3 | 51.1 | 59.7 | **62.3** | **67.5** | **50.4** | **69.4** |

Table 8.6: Performance of Model L trained with MAP on six languages. Supervised model selection were applied to choose the initializer $\vec{\theta}^{(0)}$ and smoothing value $\lambda$. The last two lines show unsupervised and supervised model selection on $\vec{\theta}^{(0)}$, $\lambda$, and $\delta$. Boldface marks trials significantly better than MAP/EM on Model A (with supervised selection, fourth line) under a sign test ($p < 0.05$). The third line and the $\delta = 0$ line are equivalent up to slight differences in implementation. Even when Model L did not outperform Model A *significantly*, it was no worse.

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| MAP/Model A (s-sel.) | 54.4 | 71.9 | 41.6 | 62.2 | 45.6 | 63.6 | 50.0 | 60.9 | 48.0 | 59.1 | 42.3 | 64.1 |
| MAP/Model L (s-sel.) | **61.3** | 70.7 | **61.8** | **69.4** | **49.2** | 64.3 | 51.1 | 59.7 | **62.3** | **67.5** | **50.4** | **69.4** |
| $\delta_0 =$ | | | | | | | | | | | | |
| -1.0 | **66.2** | **72.0** | **48.8** | **66.1** | **55.8** | **69.8** | **58.0** | **64.3** | **60.7** | **63.8** | 43.5 | 59.5 |
| -0.8 | **66.3** | 72.1 | **48.8** | **66.1** | **55.2** | **69.4** | **55.8** | 63.6 | **60.6** | **63.8** | 42.1 | 59.5 |
| -0.6 | **71.8** | **74.5** | **66.7** | **73.1** | **58.7** | **72.3** | 53.8 | 62.7 | **62.0** | **67.9** | **46.8** | **67.4** |
| -0.4 | **70.2** | **75.1** | **53.4** | **66.0** | **58.3** | **71.7** | **54.1** | 62.9 | **61.9** | **67.9** | **50.5** | **69.6** |
| -0.2 | **65.6** | 72.2 | **50.2** | **66.9** | **58.1** | **71.7** | **51.3** | 60.8 | **62.3** | **67.5** | **45.2** | 64.4 |
| supervised selection | **71.8** | **74.5** | **66.7** | **73.1** | **58.7** | **72.3** | **58.0** | **64.3** | **62.3** | **67.5** | **50.5** | **69.6** |

Table 8.7: Performance of Model L trained with structural annealing on $\delta$, on six languages. Supervised model selection was applied to choose the initializer $\vec{\theta}^{(0)}$, smoothing value $\lambda$, and $\delta_f$; the final row also selects over $\delta_0$. Boldface marks trials significantly better than MAP/EM (third line) under a sign test ($p < 0.05$).

| | German test accuracy | | English test accuracy | | Bulgarian test accuracy | | Mandarin test accuracy | | Turkish test accuracy | | Portuguese test accuracy | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected | directed | undirected |
| ATTACH-LEFT | 8.2 | 59.1 | 22.6 | 62.1 | 37.2 | 61.0 | 13.1 | 56.1 | 6.6 | 68.6 | 36.2 | 65.7 |
| ATTACH-RIGHT | 47.0 | 55.2 | 39.5 | 62.1 | 23.8 | 61.0 | 42.9 | 56.1 | 61.8 | 68.3 | 29.5 | 65.7 |
| MAP (u-sel.) | 19.8 | 55.2 | 41.6 | 62.2 | 44.6 | 63.1 | 37.2 | 56.1 | 41.2 | 57.8 | 37.4 | 62.2 |
| $\delta_0 =$ | | | | | | | | | | | | |
| -1.0 | **64.9** | **68.5** | **44.1** | **62.9** | 41.7 | 62.1 | 29.6 | 54.8 | **57.9** | **66.7** | 33.3 | 60.0 |
| -0.8 | **23.8** | 53.8 | **44.2** | **63.0** | 44.5 | 63.4 | **39.8** | **58.4** | **62.3** | **68.3** | 33.4 | 60.0 |
| -0.6 | **24.1** | 54.2 | **45.1** | **63.4** | 41.0 | 61.5 | 32.5 | 55.0 | **59.8** | **67.6** | 33.5 | 60.1 |
| -0.4 | **23.5** | 53.6 | **43.5** | 62.6 | 41.2 | 61.4 | 32.3 | 54.0 | **60.9** | **67.7** | 33.3 | 59.9 |
| -0.2 | **21.3** | 55.7 | **43.6** | 62.8 | 41.8 | 62.8 | 32.1 | 54.0 | **61.6** | **67.6** | 33.2 | 59.9 |
| unsupervised selection | **23.5** | 53.6 | **43.6** | 62.8 | 41.0 | 61.5 | 32.1 | 54.0 | **60.9** | **67.7** | 33.2 | 59.9 |

Table 8.8: Performance of Model L trained with structural annealing on $\delta$, on six languages. Unsupervised model selection was applied to choose the initializer $\vec{\theta}^{(0)}$, smoothing value $\lambda$, and $\delta_f$; the final row also selects over $\delta_0$. Boldface marks trials significantly better than MAP/EM (third line) under a sign test ($p < 0.05$).

| | Attach-Left | | Attach-Right | | MAP/EM | | CE | | SA | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ave. path length | **D** (bits) | ave. path length | **D** (bits) | ave. path length | **D** (bits) | ave. path length | **D** (bits) | ave. path length | **D** (bits) |
| German | 1.58 | 0.175 | 1.52 | 0.175 | 1.23 | 0.107 | 1.23 | 0.101 | **1.15** | **0.091** |
| English | 1.50 | 0.107 | 1.50 | 0.107 | 1.31 | 0.128 | 1.27 | 0.055 | **1.19** | **0.051** |
| Bulgarian | 1.46 | 0.120 | 1.53 | 0.120 | 1.34 | 0.110 | 1.42 | 0.117 | **1.20** | **0.089** |
| Mandarin | 1.62 | 0.112 | 1.43 | 0.112 | 1.28 | 0.096 | 1.44 | 0.120 | **1.24** | **0.049** |
| Turkish | 1.46 | **0.057** | 1.20 | **0.057** | 1.25 | 0.078 | 1.21 | 0.061 | **1.18** | 0.063 |
| Portuguese | 1.30 | 0.118 | 1.39 | 0.118 | 1.26 | 0.112 | **1.10** | **0.047** | 1.18 | 0.118 |

Table 8.9: Error statistics for supervisedly selected models for different languages. The statistics are the average undirected path length between a word and its true parent, through the hypothesized tree, macro-averaged by test data sentences; and the KL divergence between the empirical link type distribution and that hypothesized by the model on the test data. In this table, smaller numbers are always better. Boldface marks the best scores in each row.

mean between the hypothesized link type distribution and the gold-standard distribution. Smaller numbers are better in both cases; these scores are shown to bolster the case that our methods are an improvement over EM.

## 8.6 On Supervised Model Selection

Many experiments throughout the thesis have made use of annotated development datasets of about 500 sentences for model selection. Here we see how performance would be affected if this set were reduced in size. Figure 8.1 shows that, with only a few exceptions, supervised model selection *within* a training setting (MAP/EM, CE, SDA, or SA) or *across* settings would perform very similarly even if the development set were substantially reduced in size. These plots also show curves for supervised learning on the development dataset, which is consistently a better use of annotated examples (when 500 are available, as we have assumed) than supervised selection.

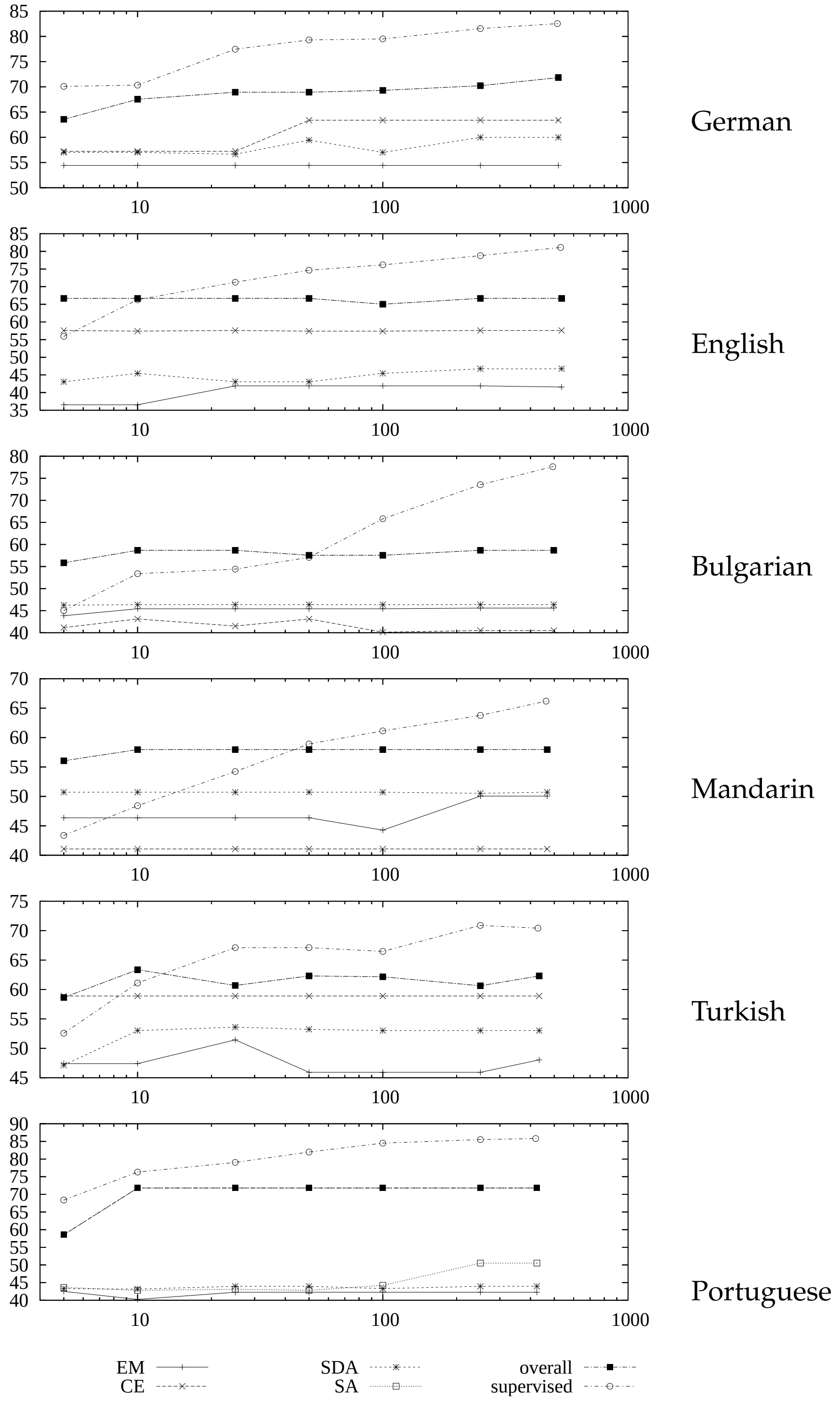


Figure 8.1: The effect of smaller development sets on supervised model selection. These plots show how test set directed accuracy changes as the amount of annotated development data used for model selection is diminished. The rightmost point corresponds to results shown earlier. Selection "overall" always coincides with CE (Portuguese) or SA (other languages). Supervised curves correspond to the use of the development data (alone) for supervised training; they illustrate that our supervised model selection method is *not* the best way to use the annotated examples in most cases.

| | existing baseline | | new result | |
|---|---|---|---|---|
| German | 54.4 | EM/Model A<br>(K&M, $\lambda = 10$) | 71.8 | SA/Model L<br>(Zero, $\lambda = 1$, $\delta = -0.6 \rightarrow 0.5$) |
| English | 41.6 | EM/Model A<br>(K&M, $\lambda = 10^{-2/3}$) | 66.7 | SA/Model L<br>(Zero, $\lambda = 10$, $\delta = -0.6 \rightarrow 0.1$) |
| Bulgarian | 45.6 | EM/Model A<br>(K&M, $\lambda = 10^{-1/3}$) | 58.3 | SA/Model L<br>(K&M, $\lambda = 10$, $\delta = -0.4 \rightarrow 0.2$) |
| Mandarin | 50.0 | EM/Model A<br>(K&M, $\lambda = 10^{1/3}$) | 58.0 | SA/Model L<br>(K&M, $\lambda = 10^{2/3}$, $\delta = -1 \rightarrow 0.1$) |
| Turkish | 61.8 | ATTACH-RIGHT | 62.4 | EM/Model L<br>(Zero, $\lambda = 1$, $\delta = -0.2$) |
| Portuguese | 42.5 | EM/Model A<br>(Zero, $\lambda = 0$) | 71.8 | CE/DEL1ORTRANS1/Model A<br>(Local, $\sigma^2 = 10^{-1/3}$) |

Table 8.10: Summary of improvements on directed attachment accuracy of grammar induction in six languages. The left column shows the best baseline methods that existed prior to this thesis, the right column shows the top performance achieved with the new techniques. Supervised model selection was applied to choose the trials presented in *both* columns.

## 8.7 Summary

In this chapter we applied contrastive estimation, skewed deterministic annealing, and structural annealing to five additional languages. Structural annealing (or at least the use of a structural bias that is chosen appropriately) is the "safest" method—the one least likely to hurt performance. Figures 8.2 and 8.3 compare all methods across languages (under supervised and unsupervised selection, respectively), and Table 8.10 describes the best baseline performance and the best performance achieved in this thesis (and how each was obtained). We summarize by addressing each new language in turn.

**German** Using structural annealing we improved performance on German by 17 points. Our methods all behaved in similar patterns on German as on English, with each technique demonstrating improvements over EM. German displays less of a locality tendency than English (57% of true dependencies in the training corpus are length-1, compared to 63% for English). Note that the locality bias trials (both fixed and annealed $\delta$) performed best with relatively high values of the locality bias $\delta$—yet introducing the bias was clearly helpful.

**Bulgarian** Performance on Bulgarian was improved by 12 points using structural an-

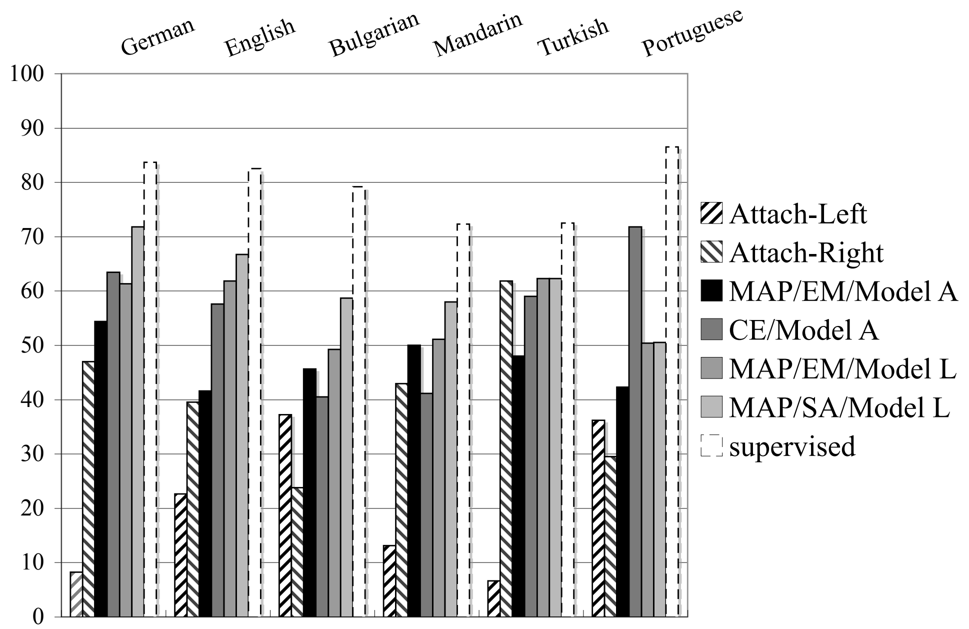


Figure 8.2: Summary of results achieved by baselines, novel methods, and supervised upper bound in six languages. *Supervised* model selection was applied, where appropriate. CE results refer to the DEL1ORTRANS1 neighborhood.

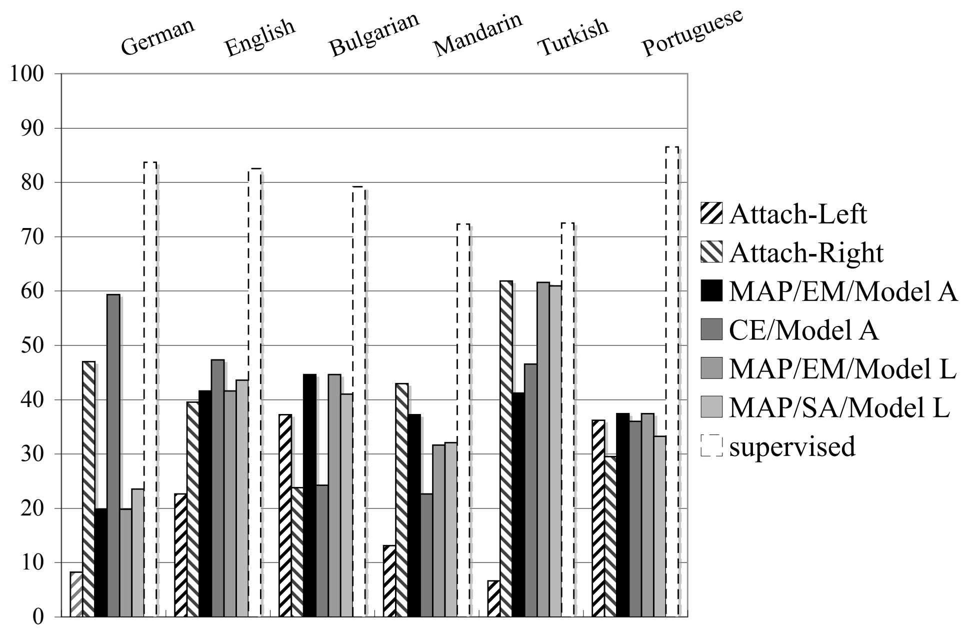


Figure 8.3: Summary of results achieved by baselines, novel methods, and supervised upper bound in six languages. *Unsupervised* model selection was applied, where appropriate. CE results refer to the DEL1ORTRANS1 neighborhood.

nealing. Contrastive estimation with the neighborhoods we tested did not improve performance over EM. (They also did not lag terribly behind.) Bulgarian has relatively free word order, so the neighborhoods that perform well for more fixed-order languages (like English), especially the ones involving transpositions like DEL1ORTRANS1 and DYNASEARCH, are not likely to provide the right kind of implicit negative evidence.

**Mandarin** We improved performance on Mandarin by eight points using structural annealing. Notably, contrastive estimation with the tested neighborhoods did not improve over EM for Mandarin (though DYNASEARCH was not terribly far behind). This corpus contains many noun compounds, the dependency structure of which is nearly impossible to learn without lexical features (which Models A and L do not have). Indeed, our selected model achieves only 37% undirected recall on NN/NN links. Future work on Mandarin will require lexicalization—a challenge because spelling and morphological features are not available in Mandarin. Better-informed neighborhoods for contrastive estimation, perhaps combined with structural annealing, might also be useful.

**Turkish** Our smallest gains were on Turkish at less than one percentage point, but significant under a sign test ($p < 0.05$). The best baseline system is ATTACH-RIGHT, and it was only surpassed by structural annealing. That this baseline performs so well is somewhat surprising, given that Turkish is free word order language with a complex morphology that makes it relatively difficult to parse (see results in the recent dependency parsing shared task for Turkish, Buchholz and Marsi, 2006); *supervised* statistical parsing has only very recently been applied to Turkish (Eryiğit and Oflazer, 2006). The (relatively) good performance of ATTACH-RIGHT is probably due in part to the distribution of sentence lengths (the mean in this corpus, after selecting sentences of length at most ten, is 5.3). Further experimentation with Turkish should take into account a larger corpus containing longer sentences with more varied constructions.

**Portuguese** This corpus was the smallest tested (fewer than 2,000 sentences). Contrastive estimation was found to increase performance by 30 points over EM, using the DEL1ORTRANS1 neighborhood. Other methods gave the same pattern of gains observed for English, though annealing the structural bias $\delta$ did not improve over performance with a fixed $\delta$ for annealing schedules that we tested. A small dataset like this one might require slower annealing, which would prevent over-committing to particular hypotheses when stepping between different $\delta$ values. A model with a reduced hypothesis space (like Model U in Section 4.8), perhaps for the first stage of learning only, might also be appropriate for small datasets.

To conclude, we have not succeeded in building a "universal" language learner, but we did not expect to. Model A simply does not capture many of the features that would be required for successful learning in these diverse languages: morphology, lexical patterns (see Section 2.3.3), and flexibility of word order, for example. The techniques have, however, improved over state-of-the-art baselines for all six languages and (more importantly) opened up the possibility of better model development in future work, by allowing more diverse features in unsupervised estimation.

## Chapter 9

# Conclusion

*When you give for an academic audience a lecture that is crystal clear from* $\mathbf{A}$ *to* $\boldsymbol{\Omega}$*, your audience feels cheated and leaves the lecture hall commenting to each other: "That was rather trivial, wasn't it?" The sore truth is that complexity sells better.*

—Edsger Dijkstra (1930–2002)

We have presented three novel techniques to improve upon Expectation-Maximization (Chapter 3; Dempster *et al.*, 1977) for learning useful hidden structure in natural language data:

- **Contrastive estimation** (Chapter 4; Smith and Eisner, 2005a,b), which mathematically formulates implicit negative evidence into the objective function and solves computational problems faced by unsupervised learning of log-linear models over infinite discrete spaces.

- **Skewed deterministic annealing** (Chapter 5; Smith and Eisner, 2004), which is similar to deterministic annealing (Rose, 1998) in using "cautious" search that starts with an easy concave problem, gradually morphing into the original likelihood problem, but does not ignore a good initializer.

- **Structural annealing** (Chapter 6; Smith and Eisner, 2006), which gradually manipulates a single structural feature added to the model to bias the learner initially toward simple structures, decreasing the bias over time.

While we focused on a grammar induction task, these methods have been described in generality and are expected to be useful in many applications. We have described how contrastive estimation provides a way to carry out *task-focused* structure learning that makes use of additional domain knowledge not available to maximum likelihood estimation, and how the two annealing techniques can guide a learner from easy to harder problems, where, again, “ease” is defined using information about the domain (here, the tendency for syntactic relations to be string-local).

We have shown that contrastive estimation performs is far more effective than EM at learning to disambiguate part-of-speech tags from unannotated text. We have shown that CE, skewed deterministic annealing, and structural annealing significantly outperform EM on dependency parsing accuracy when used to train a state-of-the-art model designed for unsupervised learning (the dependency model with valency features of Klein and Manning, 2004, which is our Model A). Further, we have shown significant improvements on grammar induction for five additional, diverse languages (German, Bulgarian, Mandarin, Turkish, and Portuguese). Our best results were obtained using a small amount of annotated data for model selection (the EM baseline was given the same treatment), though in many scenarios unsupervised model selection with the new methods also improved over MLE-trained models.

The grammar induction task as defined here and in prior work, we have pointed out (Section 1.4), is somewhat artificial: the datasets are relatively small, the data are assumed to be clean and correctly tagged (in most of our experiments), only short sequences are used in training and testing (in most experiments, at most ten words), and the parsers are evaluated against a gold standard rather than in a real application. It is left to future work to explore how these methods (including EM) can be scaled up to larger datasets and more interesting probabilistic models, and evaluated in the context of larger systems. Such “Olympic sport” tasks continue to play an important role in the field, however, allowing direct comparison of core techniques on standard datasets. In response to this criticism of the task, we argue that our methods have substantially improved the quality of parsing models learned from unannotated data. Throughout the thesis we have presented error analysis and alternative means of evaluation.

Many aspects of learning linguistic structure using unannotated data have been explored here. We briefly note some of the most interesting experimental results. While very expensive in practice, we showed in Section 3.3.2 that multiple random initializers to EM can be used, with supervised model selection, to achieve far higher accuracy than the standard “clever” initializers. We also saw in Sections 3.4.2, 4.6.6, 5.3.3, 6.2.3, and 8.6 that significantly reducing the amount of annotated data used for model selection has very little

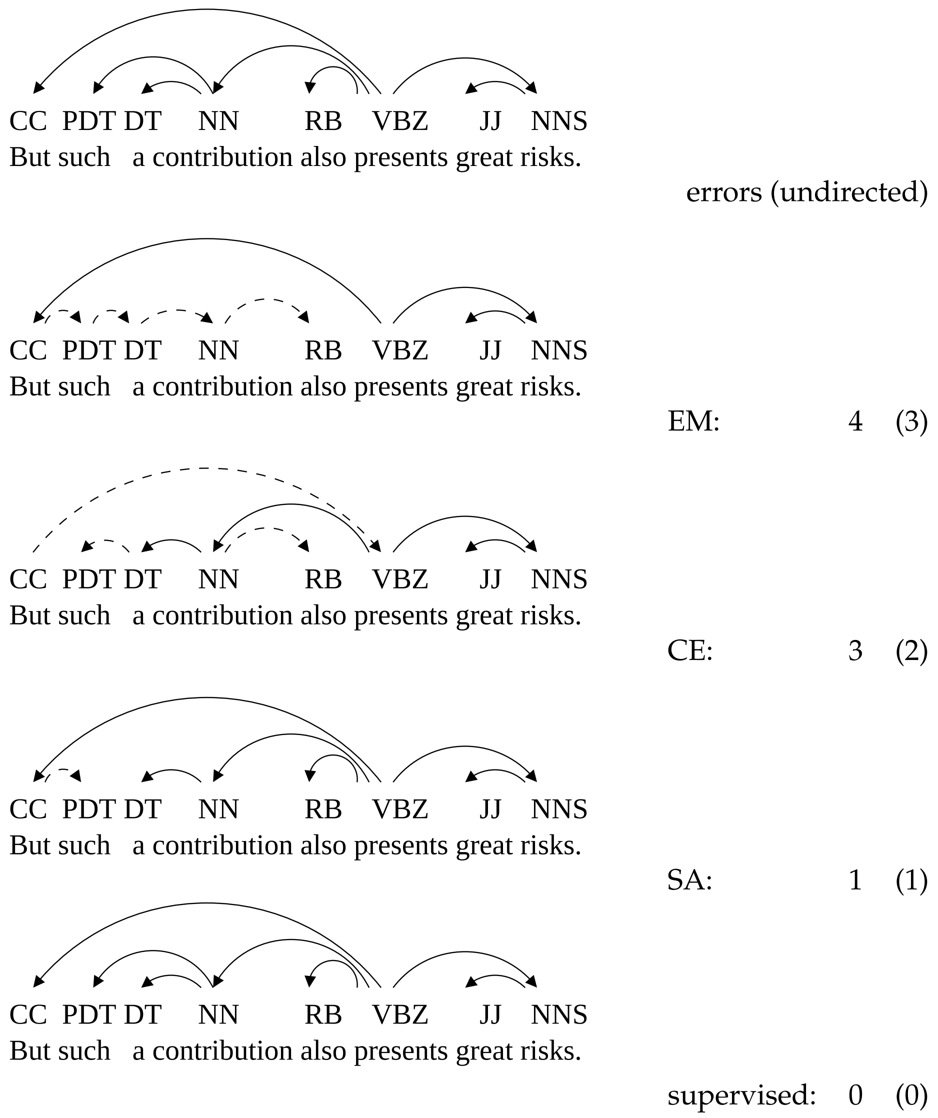

CC PDT DT NN RB VBZ JJ NNS
But such a contribution also presents great risks.
errors (undirected)
CC PDT DT NN RB VBZ JJ NNS
But such a contribution also presents great risks.
EM: 4 (3)
CC PDT DT NN RB VBZ JJ NNS
But such a contribution also presents great risks.
CE: 3 (2)
CC PDT DT NN RB VBZ JJ NNS
But such a contribution also presents great risks.
SA: 1 (1)
CC PDT DT NN RB VBZ JJ NNS
But such a contribution also presents great risks.
supervised: 0 (0)

Figure 9.1: An example sentence from our test set, drawn from the Penn Treebank, with four dependency parses: the gold standard, the baseline method (EM, supervisedly selected), two of our methods (contrastive estimation and structural annealing, supervisedly selected), and a supervised model. This example was chosen because performance on it was typical for each parser (close to the parser's average). The numbers shown are the attachment errors and the undirected attachment errors. Errors with respect to the gold standard are shown as dashed lines. See Figures 1.1 (page 13) and 6.14 (page 6.14) for more examples.

effect on the accuracy of any of the unsupervised training methods we applied. However, when more than trivial amounts of annotated data are available, our combination of unsupervised training with supervised selection (of initializers, regularization, and annealing parameters) is notably less effective than supervised training alone. One promising result does stand out: in Section 5.3.3 we saw that skewed deterministic annealing is the best among the various semisupervised methods, outperforming MLE on the annotated data alone with up to 50 annotated examples. We showed in Section 7.2 that different the differences among unsupervised learning algorithms can carry through to later stages in a cascade of learners; we first learned to disambiguate part-of-speech tags, then used the hypothesized tags to induce syntactic dependencies.

In addition to using standard and obvious initialization in training our models, we carried out experiments using *random* initializers. These served to highlight the severity of the local maximum problem—even for learning methods that explicitly seek to avoid local maxima, like skewed deterministic annealing—each of one hundred differently-initialized models consistently led to a different, locally optimal parameter estimate, for each training method. These results also showed the role of chance in learning; methods that perform better than EM with well-reasoned initializers do not consistently perform better than EM with random initializers—CE/DEL1ORTRANS1 is the most notable such example, as seen in Section 4.6.5 and Section 7.4. While this casts some doubt on a general claim that the new methods are preferred, we note that unsupervised learning nearly always, in practice, uses reasonable initializers, and our methods *do* tend to make better use of those initializers.

Consistently throughout the thesis, our results illustrated that, for good performance, regularization and initialization must be chosen carefully and together, and a good choice will depend heavily on the training algorithm. These hyperparameters can have a very large effect on the quality of the learned model.

Ultimately, when a small amount of labeled data are present, the best experimental results achieved in this thesis were obtained using structural annealing to train an improved model (Model L), equivalent to the original model but for the addition of a single locality feature. (See Table 8.10 (page 212) and Figures 8.2 (page 213) and 8.3 (page 214) for a summary of the improvements by language and a graphical comparison of all methods on all six languages tested.) Our results, then, give another example of improved machine learning by an improved choice of features. This is not surprising or revolutionary, though annealing the feature's importance is a novel approach that had a large effect on performance.

When we move to *purely unsupervised* model learning (with even unsupervised model selection), contrastive estimation was shown to be the best method for English. This was also true for German (the other language tested with more than 5,000 training sentences). Contrastive estimation, then, has not been completely surpassed in all scenarios and remains an interesting area for future research.

A key contribution of the thesis is that now, new features *can* be added to models used for unsupervised learning, whereas before, there was no clean, principled way to train the weights of nontraditional features (e.g., in the case of tagging and parsing grammars, those that do not correspond to grammar rules) from unannotated data.[1] Indeed, contrastive estimation and structural annealing are two entirely novel options for adding new features to a model over discrete structures. The former modifies the estimation criterion, avoiding numerical and computational difficulties posed by the classical MLE approach. CE also allows the incorporation of implicit negative evidence in an entirely novel way, as we have noted. Some neighborhoods exist that are essentially approximations for MLE, so while implicit negative evidence can improve performance, it is not required for using CE. Further, as we have seen, even without additional features CE can outperform MLE and MAP on many problems.

Structural annealing treats one additional feature's weight as a bias that is outside the learned parameters—depending on the feature's role in the model, it can be changed over time (annealed) to guide the learner from simple or easy structures to more complex ones. Both CE and SA were shown to be highly successful in comparison to the standard Expectation-Maximization approach to unsupervised learning.

[1]These methods are not just for grammars, of course, and our exposition has been sufficiently general that it should be clear how to apply them to other kinds of models over other kinds of discrete data.

# Appendix A

# Dynamic Programming Equations

We give semiring dynamic programming equations for Model A (a specialized variant of the SHAG recognition algorithm in Eisner and Satta (1999)), Model A on lattices, Model U (seen in Chapter 4), Model L, and Model S (both seen in Chapter 6). These are presented as Dyna programs (Eisner *et al.*, 2004), though we abstract over the semiring and use $\oplus$ and $\otimes$. Each program computes the value of $\mathrm{goal}$, which has a different interpretation depending on the semiring. In the $\langle \max, \times \rangle$ semiring, it is the score of the most probable parse. In the $\langle +, \times \rangle$ semiring, it is the total score of all parses, $\ddot{Z}_{\vec{\lambda}}(\{\mathbf{x}\} \times \mathcal{Y}_{\mathbf{x}})$ (also written $\ddot{Z}_{\vec{\lambda}}(\mathbf{x})$). In practice these algorithms are implemented in the logarithmic domain, replacing $+$ with LOGADD and $\times$ with $+$. While slightly more expensive, this prevents underflow and, in the case of log-linear models, overflow.

Our procedural implementations of these declarative programs use the algorithms described in detail in Eisner *et al.* (2005).

## A.1 Model A

Let $\mathbf{x}$ be the sentence, and let $n$ be the length of the sentence.

$$\mathrm{rconstit}(I-1, I, \mathbf{false}) \quad = \quad 1 \text{ whenever } I < n. \tag{A.1}$$

$$\mathrm{lconstit}(I-1, I, \mathbf{false}) \quad = \quad 1 \text{ whenever } I < n. \tag{A.2}$$

$$\mathrm{rconstit}(I, K, \mathbf{true}) \quad \oplus= \quad \mathrm{rtrpzd}(I, J, B) \otimes \mathrm{rconstit}'(J-1, K). \tag{A.3}$$

$$\mathrm{lconstit}(I, K, \mathbf{true}) \quad \oplus= \quad \mathrm{lconstit}'(I, J) \otimes \mathrm{ltrpzd}(J-1, K, B). \tag{A.4}$$

$$\text{rconstit}'(I, J) \quad \oplus= \quad \text{rconstit}(I, J, B) \otimes \lambda_{\mathbf{stop}(x_{I+1},\text{right},B)}. \tag{A.5}$$

$$\text{lconstit}'(I, J) \quad \oplus= \quad \text{lconstit}(I, J, B) \otimes \lambda_{\mathbf{stop}(x_{J-1},\text{left},B)}. \tag{A.6}$$

$$\begin{aligned}\text{rtrpzd}(I, K, B) \quad \oplus= \quad & \text{rconstit}(I, J, B) \otimes \text{lconstit}'(J, K) \\ & \otimes\lambda_{\mathbf{child}(x_{I+1},\text{right},x_K)} \otimes \lambda_{\mathbf{continue}(x_{I+1},\text{right},B)}.\end{aligned} \tag{A.7}$$

$$\begin{aligned}\text{ltrpzd}(I, K, B) \quad \oplus= \quad & \text{rconstit}'(I, J) \otimes \text{lconstit}(J, K, B) \\ & \otimes\lambda_{\mathbf{child}(x_K,\text{left},x_{I+1})} \otimes \lambda_{\mathbf{continue}(x_K,\text{left},B)}.\end{aligned} \tag{A.8}$$

$$\text{goal} \quad \oplus= \quad \text{lconstit}'(0, I) \otimes \text{rconstit}'(I-1, n) \otimes \lambda_{\mathbf{root}(x_I)}. \tag{A.9}$$

## A.2 Model A on Lattices

The equations for Model A, run on lattices, are very similar to those for sequences (above). The equations change very little, except that now the variables $I$, $J$, and $K$ refer to states in the lattice rather than positions between symbols in the sequence. Define $\text{arc}(X, I, J)$ to be **true** if and only if there is an arc in the lattice between states $I$ and $J$. Let state 0 be the start state, and let $\text{finalstate}(I)$ be **true** iff state $I$ is a final state. Some additional bookkeeping is done within the terms; note that terms can be nested (a nested term is not evaluated to its value).

$$\text{rconstit}(\text{arc}(X, I, J), J, \mathbf{false}) \quad = \quad 1 \text{ whenever } \text{arc}(X, I, J). \tag{A.10}$$

$$\text{lconstit}(\text{arc}(X, I, J), I, \mathbf{false}) \quad = \quad 1 \text{ whenever } \text{arc}(X, I, J). \tag{A.11}$$

$$\begin{aligned}\text{rconstit}(\text{arc}(X, I, J), M, \mathbf{true}) \quad \oplus= \quad & \text{rtrpzd}(\text{arc}(X, I, J), \text{arc}(X', K, L), B) \\ & \otimes\text{rconstit}'(\text{arc}(X', K, L), M).\end{aligned} \tag{A.12}$$

$$\begin{aligned}\text{lconstit}(\text{arc}(X, L, M), I, \mathbf{true}) \quad \oplus= \quad & \text{lconstit}'(\text{arc}(X', J, K), I) \\ & \otimes\text{ltrpzd}(\text{arc}(X, L, M), \text{arc}(X', J, K), B).\end{aligned} \tag{A.13}$$

$$\text{rconstit}'(\text{arc}(X, I, J), K) \quad \oplus= \quad \text{rconstit}(\text{arc}(X, I, J), K, B) \otimes \lambda_{\mathbf{stop}(X,\text{right},B)}. \tag{A.14}$$

$$\text{lconstit}'(\text{arc}(X, J, K), I) \quad \oplus= \quad \text{lconstit}(\text{arc}(X, J, K), I, B) \otimes \lambda_{\mathbf{stop}(X,\text{left},B)}. \tag{A.15}$$

$$\text{rtrpzd}(\text{arc}(X, I, J), \text{arc}(X', L, M), B) \quad \oplus= \quad \text{rconstit}(\text{arc}(X, I, J), K, B)$$

$$\otimes \text{lconstit}'(\text{arc}(X', L, M), K)$$

$$\otimes \lambda_{\textbf{child}(X,\text{right},X')} \otimes \lambda_{\textbf{continue}(X,\text{right},B)}. \quad \text{(A.16)}$$

$$\text{ltrpzd}(\text{arc}(X, L, M), \text{arc}(X', I, J), B) \oplus= \text{rconstit}'(\text{arc}(X', I, J), K) \otimes \text{lconstit}(\text{arc}(X, I, J), K, B) \otimes \lambda_{\textbf{child}(X,\text{left},X')} \otimes \lambda_{\textbf{continue}(X,\text{left},B)}. \quad \text{(A.17)}$$

$$\text{goal} \oplus= \text{lconstit}'(\text{arc}(X, I, J), 0) \otimes \text{rconstit}'(\text{arc}(X, I, J), n) \otimes \lambda_{\textbf{root}(X)}. \quad \text{(A.18)}$$

## A.3 Model U

$$\text{rconstit}(I, I+1, \textbf{false}) = 1 \text{ whenever } I \le n. \quad \text{(A.19)}$$

$$\text{lconstit}(I, I+1, \textbf{false}) = 1 \text{ whenever } I \le n. \quad \text{(A.20)}$$

$$\text{rconstit}(I, K, \textbf{true}) \oplus= \text{rtrpzd}(I, J, B) \otimes \text{rconstit}'(J-1, K). \quad \text{(A.21)}$$

$$\text{lconstit}(I, K, \textbf{true}) \oplus= \text{lconstit}'(I, J) \otimes \text{ltrpzd}(J-1, K, B). \quad \text{(A.22)}$$

$$\text{rconstit}'(I, J) \oplus= \text{rconstit}(I, J, B). \quad \text{(A.23)}$$

$$\text{lconstit}'(I, J) \oplus= \text{lconstit}(I, J, B). \quad \text{(A.24)}$$

$$\text{rtrpzd}(I, K, B) \oplus= \text{rconstit}(I, J, B) \otimes \text{lconstit}'(J, K) \otimes \lambda_{\langle x_{I+1}, x_K \rangle}. \quad \text{(A.25)}$$

$$\text{ltrpzd}(I, K, B) \oplus= \text{rconstit}'(I, J) \otimes \text{lconstit}(J, K, B) \otimes \lambda_{\langle x_{I+1}, x_K \rangle}. \quad \text{(A.26)}$$

$$\text{goal} \oplus= \text{lconstit}'(0, 1) \otimes \text{rconstit}'(0, n). \quad \text{(A.27)}$$

## A.4 Model L

Start with the equations for Model A, replacing Equations A.7 and A.8 with:

$$\text{rtrpzd}(I, K, B) \oplus= \text{rconstit}(I, J, B) \otimes \text{lconstit}'(J, K) \quad \text{(A.28)}$$

$$\otimes \lambda_{\mathbf{child}(x_{I+1},\text{right},x_K)} \otimes \lambda_{\mathbf{continue}(x_{I+1},\text{right},B)} \tag{A.29}$$
$$\otimes e^{\delta\cdot(K-I+1)}. \tag{A.30}$$
$$\text{ltrpzd}(I,K,B) \quad \oplus= \quad \text{rconstit}'(I,J) \otimes \text{lconstit}(J,K,B) \tag{A.31}$$
$$\otimes \lambda_{\mathbf{child}(x_K,\text{left},x_{I+1})} \otimes \lambda_{\mathbf{continue}(x_K,\text{left},B)} \tag{A.32}$$
$$\otimes e^{\delta\cdot(K-I+1)}. \tag{A.33}$$
$$\tag{A.34}$$

## A.5 Model S

Remove the goal equation from Model A (Equation A.9), and add instead the following:

$$\text{vine}(0) \quad = \quad 1. \tag{A.35}$$
$$\text{openvine}(X,J+1) \quad \oplus= \quad \text{vine}(I) \otimes \text{lconstit}'(I,I+1). \tag{A.36}$$
$$\text{openvine}(X,J) \quad \oplus= \quad \text{openvine}(X',I) \otimes \text{ltrpzd}(I-1,J,B)$$
$$\otimes \lambda_{\mathbf{stop}(x_J,\text{left},B)}. \tag{A.37}$$
$$\text{vine}'(X,J) \quad \oplus= \quad e^{\beta} \otimes \text{openvine}(X,J) \otimes \lambda_{\mathbf{root}(X)}. \tag{A.38}$$
$$\text{vine}'(X,J) \quad \oplus= \quad \text{vine}'(X',I) \otimes \text{rtrpzd}(I-1,J,B)$$
$$\otimes \lambda_{\mathbf{stop}(x_I,\text{right},B)}. \tag{A.39}$$
$$\text{vine}(I) \quad \oplus= \quad \text{vine}'(X,I) \otimes \text{rconstit}'(I-1,I). \tag{A.40}$$
$$\text{goal} \quad \oplus= \quad \text{vine}(n). \tag{A.41}$$

These equations were presented in Eisner and Smith (2005); see Figure 2b in that paper for a more graphical representation.

# Appendix B

# Part-of-Speech Tagsets

We briefly describe the tagsets for each of the datasets used in the thesis.

## B.1 Penn Treebank

The Penn Treebank is available from the Linguistic Data Consortium; see `http://www.ldc.upenn.edu`, catalog number LDC99T42. The tagset is shown in Table B.1. For more details, readers are directed to Santorini (1990). The coarse tags used in some POS tagging experiments in Chapter 4 are given in Table B.2.

## B.2 TIGER Treebank

The TIGER Treebank is free for scientific use; details are available at `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/`. The tagset is shown in Table B.3. For more details, see Smith (2003).

## B.3 BulTreeBank

The BulTreeBank is described in detail at `http://www.bultreebank.org`. The version of the data used in this thesis was as distributed by the organizers of the CoNLL-X shared task Buchholz and Marsi (2006). The tagset used here is shown in Table B.4. For more details, see Simov, Osenova, and Slavcheva (2004b).

| | |
|---|---|
| CC | coordinating conjunction |
| CD | cardinal number |
| DT | determiner |
| EX | existential *there* |
| FW | foreign word |
| IN | preposition or subordinating conjunction |
| JJ | adjective |
| JJR | adjective, comparative |
| JJS | adjective, superlative |
| LS | list item marker |
| MD | modal |
| NN | noun, singular or mass |
| NNS | noun, plural |
| NNP | proper noun, singular |
| NNPS | proper noun, plural |
| PDT | predeterminer |
| POS | possessive ending |
| PRP | personal pronoun |
| PRP$ | possessive pronoun |
| RB | adverb |
| RBR | adverb, comparative |
| RBS | adverb, superlative |
| RP | particle |
| SYM | symbol |
| TO | *to* |
| UH | interjection |
| VB | verb, base form |
| VBD | verb, past tense |
| VBG | verb, gerund or present participle |
| VBN | verb, past participle |
| VBP | verb, non-third person singular present |
| VBZ | verb, third person singular present |
| WDT | wh-determiner |
| WP | wh-pronoun |
| WP$ | possessive wh-pronoun |
| WRB | wh-adverb |

Table B.1: The Penn Treebank tagset. WP$ does not appear in sentences of ten words or fewer, and so is not included in models trained on that portion of the Treebank.

| coarse tag | Treebank tags |
|---|---|
| ADJ | $ # CD JJ JJR JJS PRP$ RB RBR RBS |
| CONJ | CC |
| DET | DT PDT |
| ENDPUNC | . |
| INPUNC | , : LS SYM UH |
| LPUNC | “ -LRB |
| N | EX FW NN NNP NNPS NNS PRP |
| POS | POS |
| PREP | IN |
| PRT | RP |
| RPUNC | ” -RRB- |
| TO | TO |
| V | MD VBD VBP VB VBZ VBG |
| VBN | VBN |
| W | WDT WP$ WP WRB |

Table B.2: Mapping of Penn Treebank tags onto our coarse tagset, used in part-of-speech tagging experiments with spelling features.

## B.4 Penn Chinese Treebank

The Chinese Treebank is available from the Linguistic Data Consortium (`http://www.ldc.upenn.edu`), catalog number LDC2005T01U01 for the latest version as of this writing. The tagset is shown in Table B.5; see Xia (2000) for more details.

## B.5 METU-Sabancı Treebank

The METU-Sabancı Turkish Treebank is available at `http://www.ii.metu.edu.tr/~corpus/treebank.html`; it is free for research purposes. The data and tagset used here is as formatted for the CoNLL-X shared task Buchholz and Marsi (2006). To quote the documentation provided with that version of the data:

> See the Appendix of [Oflazer, Say, Hakkani-Tür, and Tür (2003)] for a full list (“Minor Parts of Speech”). In most cases, the “Minor Part of Speech” fully determines the “Major Part of Speech.” In those cases, POSTAG simply contains the “Minor Part of Speech.” However, the “Minor Parts of Speech” PastPart and FutPart occur with nouns as well as adjectives. We have therefore prefixed them (and Inf and PresPart for consistency as well) with N and A respectively. If no “Minor Part of Speech” is given in the treebank, POSTAG is identical to [the “Major Part of Speech”].

| | |
|---|---|
| ADJA | adjective, attributive |
| ADJD | adjective, adverbial or predicative |
| ADV | adverb |
| APPR | preposition; circumposition left |
| APPRART | preposition with article |
| APPO | postposition |
| APZR | circumposition right |
| ART | definite or indefinite article |
| CARD | cardinal number |
| FM | foreign language material |
| ITJ | interjection |
| KOUI | subordinate conjunction with *zu* and infinitive |
| KOUS | subordinate conjunction with sentence |
| KON | coordinate conjunction |
| KOKOM | comparative conjunction |
| NN | common noun |
| NE | proper noun |
| PDS | substituting demonstrative pronoun |
| PDAT | attributive demonstrative pronoun |
| PIS | substituting indefinite pronoun |
| PIAT | attributive indefinite pronoun without determiner |
| PIDAT | attributive indefinite pronoun with determiner |
| PPER | non-reflexive personal pronoun |
| PPOSS | substituting possessive pronoun |
| PPOSAT | attributive possessive pronoun |
| PRELS | substituting relative pronoun |
| PRELAT | attributive relative pronoun |
| PRF | reflexive personal pronoun |
| PWS | substituting interrogative pronoun |
| PWAT | attributive interrogative pronoun |
| PWAV | adverbial interrogative or relative pronoun |
| PAV | pronominal adverb |
| PTKZU | *zu* before infinitive |
| PTKNEG | negative particle |
| PTKVZ | separable verbal particle |
| PTKANT | answer particle |
| PTKA | particle with adjective or adverb |
| SGML | SGML markup |
| SPELL | letter sequence |
| TRUNC | word remnant |
| VVFIN | finite verb, full |
| VVIMP | imperative, full |
| VVINF | infinitive, full |
| VVIZU | infinitive with *zu*, full |
| VVPP | perfect participle, full |
| VAFIN | finite verb, auxiliary |
| VAIMP | imperative, auxiliary |
| VAINF | infinitive, auxiliary |
| VAPP | perfect participle, auxiliary |
| VMFIN | finite verb, modal |
| VMINF | infinitive, modal |
| VMPP | perfect participle, modal |
| XY | non-word containing non-letter |

Table B.3: The TIGER Treebank tagset. Not all of these tags are present in the subset use for experiments (i.e., sentences with fewer than ten words).

| | | | |
|---|---|---|---|
| N | noun | Vpi | personal verb, imperfective |
| Nc | common noun | Vpp | personal verb, perfective |
| Np | proper noun | Vxi | auxiliary verb (*sum*), imperfective |
| A | adjective | Vyp | auxiliary verb (*buda*), perfective |
| Af | adjective, feminine | Dm | adverb of manner |
| Am | adjective, masculine | Dt | adverb of time |
| An | adjective, neuter | Dl | adverb of location |
| H | hybrid (family name or adjective) | Dq | adverb of quantity or degree |
| Hf | hybrid, feminine | Dd | adverb of modal nature |
| Hm | hybrid, masculine | Cc | coordinative conjunction |
| Hn | hybrid, neuter | Cs | subordinative conjunction |
| Pp | personal pronoun | Cr | repetitive coordinative conjunction |
| Pd | demonstrative pronoun | Cp | single and repetitive coordinative conjunction |
| Pr | relative pronoun | Ta | affirmative particle |
| Pc | collective pronoun | Tn | negative particle |
| Pi | interrogative pronoun | Ti | interrogative particle |
| Pf | indefinite pronoun | Tx | auxiliary particle |
| Pn | negative pronoun | Tm | modal particle |
| Ps | possessive pronoun | Tv | verbal particle |
| Mc | cardinal numeral | Te | emphasis particle |
| Mo | ordinal numeral | Tg | gradable particle |
| Md | adverbial numeral | R | preposition |
| My | fuzzy numeral about people | I | interjection |
| Vii | auxiliary verb (*bivam*), imperfective | | |
| Vni | impersonal verb, imperfective | | |
| Vnp | impersonal verb, perfective | | |

Table B.4: The BulTreeBank tagset. Because the tags are factored into fields, we list only those present in the data; for a complete description see Simov *et al.* (2004b).

| | |
|---|---|
| AD | adverb |
| AS | aspect marker |
| BA | *bǎ* in ba-construction |
| CC | coordinating conjunction |
| CD | cardinal number |
| CS | subordinating conjunction |
| DEC | *de* in a relative clause |
| DEG | associative *de* |
| DER | *dé* in V-de const. and V-de-R |
| DEV | *de* before VP |
| DT | determiner |
| ETC | for words *děng*, *děng děng* |
| FW | foreign words |
| IJ | interjection |
| JJ | other noun-modier |
| LB | *bèi* in long bei-const |
| LC | localizer |
| M | measure word |
| MSP | other particle |
| NN | common noun |
| NR | proper noun |
| NT | temporal noun |
| OD | ordinal number |
| ON | onomatopoeia |
| P | preposition excl. *bèi* and *bǎ* |
| PN | pronoun |
| PU | punctuation |
| SB | *bèi* in short bei-const |
| SP | sentence-nal particle |
| VA | predicative adjective |
| VC | *shì* |
| VE | *yǒu* as the main verb |
| VV | other verb |

Table B.5: The Penn Chinese Treebank tagset. Not all of these tags are present in the subset use for experiments (i.e., sentences with fewer than ten words).

For more details, see Oflazer *et al.* (2003). Because that paper does not give descriptions of the tags, we merely list those present in the dataset (note that the intended meanings are fairly clear); see Table B.6.

## B.6 Floresta Sintá(c)tica Treebank

The data used in these experiments is freely available at `http://nextens.uvt.nl/~conll/free_data.html`. The tagset is shown in Table B.7. For more details, see `http://visl.sdu.dk/visl/pt/symbolset-floresta.html`.

AFutPart
APastPart
APresPart
Adj
Adv
Card
Conj
DemonsP
Det
Distrib
Dup
Interj
NFutPart
NInf
NPastPart
Noun
Ord
PersP
Postp
Pron
Prop
Ques
QuesP
Real
ReflexP
Verb
Zero

Table B.6: The METU-Sabancı Treebank tagset. Not all of these tags are present in the subset use for experiments (i.e., sentences with fewer than ten words). The tags are listed but not described in Oflazer *et al.* (2003); we repeat the list.

| | |
|---|---|
| n | noun |
| prop | proper noun |
| adj | adjective |
| v-fin | finite verb |
| v-inf | infinitive |
| v-pcp | participle |
| v-ger | gerund |
| art | article |
| pron-pers | personal pronoun |
| pron-det | determiner pronoun (adjective-like) |
| pron-indp | independent pronoun (noun-like) |
| adv | adverb |
| num | numeral |
| prp | preposition |
| in | interjection |
| conj-s | subordinating conjunction |
| conj-c | coordinating conjunction |

Table B.7: The Floresta Sintá(c)tica Treebank tagset. Not all of these tags are present in the subset use for experiments (i.e., sentences with fewer than ten words).

# Bibliography

S. Abney. Statistical methods and linguistics. In J. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 1–26. MIT Press, Cambridge, Massachusetts, 1996.

S. Abney. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, MIT, 1987.

S. P. Abney, D. A. McAllester, and F. Pereira. Relating probabilistic grammars and automata. In *Proc. of ACL*, 1999.

W. P. Adriaans. *Language Learning from a Categorial Perspective*. Ph.D. thesis, Universiteit van Amsterdam, 1992.

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta sintá(c)tica: a treebank for Portuguese. In *Proc. of LREC*, 2002.

Y. Al-Onaizan, J. Cuřin, M. Jahr, K. Knight, J. Lafferty, I. D. Melamed, N. A. Smith, F.-J. Och, D. Purdy, and D. Yarowsky. Statistical Machine Translation. CLSP Research Notes No. 42, Johns Hopkins University, 1999.

J. F. Allen, B. W. Miller, E. K. Ringger, and T. Sikorski. A robust system for natural spoken dialogue. In *Proc. of ACL*, 1996.

E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, 1990.

H. Alshawi. Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of ACL*, 1996.

Y. Altun, M. Johnson, and T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*, 2003.

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–53, 2005.

D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–50, 1990.

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

N. B. Atalay, K. Oflazer, and B. Say. The annotation process in the Turkish treebank. In *Proc. of LINC*, 2003.

J. K. Baker. Trainable grammars for speech recognition. In *Proc. of the Acoustical Society of America*, pages 547–550, 1979.

L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3:1–8, 1972.

F. Beil, G. Carroll, D. Prescher, and M. Rooth. Inside-outside estimation of a lexicalized PCFG for German. In *Proc. of ACL*, 1999.

S. J. Benson and J. J. Moré. A limited memory variable metric method for bound constrained optimization. Technical Report ACSP909-0901, Argonne National Laboratory, 2001.

A. Berger. Convexity, maximum likelihood and all that, 1998. `http://www.cs.cmu.edu/afs/cs/user/aberger/www/ps/convex.ps`.

D. Bertsekas. *Nonlinear Optimization*. Athena Scientific, 1995.

D. M. Bikel. A distributional analysis of a lexicalized statistical parsing model. In *Proc. of EMNLP*, 2004.

J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report ICSI-TR-97-021, ICSI, 1997.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. of DARPA Workshop on Speech and Natural Language*, 1991.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, 1998.

T. L. Booth and R. A. Thompson. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, 22(5):442–450, May 1973.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER Treebank. In *Proc. of Workshop on Treebanks and Linguistic Theories*, 2002.

M. Brent. Advances in the computational study of language acquisition. *Cognition*, 61: 1–38, 1996.

M. R. Brent. Minimal generative models: a middle ground between neurons and triggers. In *Proc. of the Cognitive Science Society*, 1993.

J. Bresnan. *Lexical Functional Syntax*. Blackwell, 2001.

E. Brill. Discovering the lexical features of a language. In *Proc. of ACL*, 1991.

E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proc. of VLC*, 1995.

E. Brill and M. Marcus. Automatically acquiring phrase structure using distributional analysis. In *Proc. of DARPA Workshop on Speech and Natural Language*, 1992.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

P. F. Brown, V. J . Della Pietra, P. V. de Souza, J. C. Lai, and R. Mercer. Class-based $n$-gram models of natural language. *Computational Linguistics*, 18(4):467–79, 1992.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2): 263–311, 1993.

S. Buchholz. Unsupervised learning of subcategorisation information and its application ina parsing subtask. In *Proc. of Netherlands/Belgium Conference on AI*, 1998.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*, 2006.

R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *Proc. of HLT-EMNLP*, 2005.

S. A. Caraballo and E. Charniak. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–98, 1998.

X. Carreras and L. Màrquez. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proc. of CoNLL*, 2004.

X. Carreras and L. Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of CoNLL*, 2005.

G. Carroll and E. Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University, 1992.

G. Carroll and M. Rooth. Valence induction with a head-lexicalized PCFG. In *Proc. of EMNLP*, 1998.

F. Casacuberta and C. de la Higuera. Computational complexity of problems on probabilistic grammars and transducers. In L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications; 5th International Colloquium, ICGI 2000*, pages 15–24. Springer, 2000.

G. Casella and E. I. George. Explaining the Gibbs sampler. *American Statistician*, 46:167–174, 1992.

E. Charniak. A maximum-entropy-inspired parser. In *Proc. of NAACL*, 2000.

E. Charniak. Unsupervised learning of name structure from coreference data. In *Proc. of NAACL*, 2001.

E. Charniak. *Statistical Language Learning*. MIT Press, 1993.

E. Charniak and M. Johnson. Coarse-to-fine $n$-best parsing and maxent discriminative reranking. In *Proc. of ACL*, 2005.

E. Charniak, S. Goldwater, and M. Johnson. Edge-based best-first chart parsing. In *Proc. of VLC*, 1998.

C. Chelba and F. Jelinek. Exploiting syntactic structure for language modeling. In *Proc. of ACL*, pages 225–231, 1998.

S. Chen. Bayesian grammar induction for language modeling. In *Proc. of ACL*, 1995.

S. Chen and R. Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

Z. Chi. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–60, 1999.

Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305, 1998.

D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*, 2005.

N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1965.

A. S. Clark. *Unsupervised Language Acquisition: Theory and Practice*. Ph.D. thesis, University of Sussex, 2001.

J. Cocke and J. T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

T. Cohn and P. Blunsom. Semantic role labelling with tree conditional random fields. In *Proc. of CoNLL*, 2005.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*, 2002.

M. Collins. Discriminative reranking for natural language parsing. In *Proc. of ICML*, 2000.

M. Collins. A new statistical parser based on bigram lexical dependencies. In *Proc. of ACL*, 1996.

M. Collins. The EM algorithm, 1997a.

M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, 1999.

M. Collins. The EM algorithm, 1997b. `http://people.csail.mit.edu/mcollins/papers/wpeII.4.ps`.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. A statistical parser for Czech. In *Proc. of ACL*, 1999.

R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(2):52–67, 2002.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics & Decisions*, Supplement Issue No. 1:205–237, 1984.

A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proc. of ACL*, 2004.

I. Dagan, O. Glickman, and B. Magnini. The PASCAL recognising textual entailment challenge. In *Proc. of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–80, 1972.

C. de Marcken. Lexical heads, phrase structure and the induction of grammar. In *Proc. of the 3rd VLC*, pages 14–26, 1995a.

C. de Marcken. *Unsupervised Language Acquisition*. Ph.D. thesis, MIT, 1995b.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Y. Ding and M. Palmer. Machine translation using probabilistic synchronous dependency insertion grammar. In *Proc. of ACL*, 2005.

M. Dreyer and J. Eisner. Better informed training of latent syntactic features. In *Proc. of EMNLP*, 2006.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2): 94–102, 1970.

P. Edmonds. SENSEVAL: The evaluation of word sense disambiguation systems. *ELRA Newsletter*, 7(3), 2002.

J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*, 1996.

J. Eisner. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*, 1997.

J. Eisner. *Smoothing a Probabilistic Lexicon Via Syntactic Transformations*. Ph.D. thesis, University of Pennsylvania, 2001.

J. Eisner and G. Satta. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*, 1999.

J. Eisner and N. A. Smith. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*, 2005.

J. Eisner and R. W. Tromble. Local search with very large-scale neighborhoods for optimal permutations in machine translation. In *Proc. of HLT-NAACL Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

J. Eisner, E. Goldlust, and N. A. Smith. Dyna: A declarative language for implementing dynamic programs. In *Proc. of ACL* (companion volume), 2004.

J. Eisner, E. Goldlust, and N. A. Smith. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*, 2005.

G. Elidan and N. Friedman. The information bottleneck expectation maximization algorithm. In *Proc. of UAI*, 2003.

G. Elidan and N. Friedman. Learning hidden variable networks: the information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127, 2005.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *Proc. of AAAI*, 2002.

D. Elworthy. Does Baum-Welch re-estimation help taggers? In *Proc. of the 4th ANLP*, pages 53–58, 1994.

G. Eryiğit and K. Oflazer. Statistical dependency parsing for Turkish. In *Proc. of EACL*, 2006.

S. Finch and N. Chater. Bootstrapping syntactic categories. In *Proc. of the Cognitive Science Society*, 1992a.

S. Finch and N. Chater. Bootstrapping syntactic categories using statistical methods. In

W. Daelemans and D. Powers, editors, *Background and Experiments in Machine Learning of Natural Language*. Tilburg University: Institute for Language Technology and AI, 1992b.

J. R. Finkel, C. D. Manning, and A. Y. Ng. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*, 2006.

N. Fraser. Parsing and dependency grammar. In *University College London Working Papers in Linguistics 1*, 1989.

H. Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 8, 1965.

D. Gildea. Dependencies vs. constituents for tree-based alignment. In *Proc. of EMNLP*, 2004.

L. Gleitman. The structural sources of verb meanings. *Language Acquisition*, 1:3–55, 1990.

M. E. Gold. Language identification in the limit. *Information and Control*, 10:447–74, 1967.

M. Goldberg. An unsupervised model for statistically determining coordinate phrase attachment. In *Proc. of ACL*, 1999.

J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

C. P. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126:43–62, 2001.

J. Goodman. Exponential priors for maximum entropy models. In *Proc. of HLT-NAACL*, 2004.

J. Goodman. Parsing algorithms and metrics. In *Proc. of ACL*, 1996.

J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999.

U. Grenander. Syntax controlled probabilities. Technical report, Brown University Division of Applied Mathematics, 1967.

A. Gunawardana and W. Byrne. Discriminative speaker adaptation with conditional maximum likelihood linear regression. In *Proc. of EUROSPEECH*, 2001.

A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proc. of HLT-NAACL*, 2006.

Z. S. Harris. *Language and Information*. Columbia University Press, New York, 1988.

D. Hindle. Noun classification from predicate-argument structure. In *Proc. of ACL*, 1990.

D. Hindle and M. Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–20, 1993.

B. Hoffman. *The Computational Analysis of the Syntax and Interpretation of Free Word Order in*

*Turkish.* Ph.D. thesis, University of Pennsylvania, 1995.

J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

J. J. Horning. *A Study of Grammatical Inference*. Ph.D. thesis, Stanford University, 1969.

R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer, 1995.

R. Hudson. *Word Grammar*. Blackwell, 1984.

R. Hwa. Supervised grammar induction using training data with limited constituent information. In *Proc. of ACL*, 1999.

E. T. Jaynes. Information theory and statistical mechanics I. *Phys. Rev.*, 106:620–30, 1957.

T. Jebara. *Machine Learning: Discriminative and Generative*. Springer, 2003.

F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1997.

F. Jelinek, J. Lafferty, and R. Mercer. Basic methods of probabilistic context-free grammars. Technical Report Research Report RC 16374, IBM, 1991.

M. Johnson. Joint and conditional estimation of tagging and parsing models. In *Proc. of ACL*, 2001.

M. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24: 613–32, 1998.

M. Jordan and L. Xu. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8:1409–1431, 1996.

T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Lab, 1965.

D. Kazakov. Unsupervised learning of naïve morphology with genetic algorithms. In *Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing*, pages 105–112, 1997.

J. Kazama and J. Tsujii. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60:159–194, 2005.

M. D. Kernighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proc. of COLING*, 1990.

G. Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Ann. Phys. Chem.*, 72:497–508, 1847.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*, 2003a.

D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proc. of ACL*, 2003b.

D. Klein and C. D. Manning. Natural language grammar induction using a constituent-context model. In *Advances in NIPS 14*, 2001a.

D. Klein and C. D. Manning. Distributional phrase structure induction. In *Proc. of CoNLL*, 2001b.

D. Klein and C. D. Manning. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*, 2002a.

D. Klein and C. D. Manning. A$^*$ parsing: Fast exact Viterbi parse selection. In *Proc. of HLT-NAACL*, 2003c.

D. Klein and C. D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*, 2004.

D. Klein and C. D. Manning. Conditional structure vs. conditional estimation in NLP models. In *Proc. of EMNLP*, 2002b.

T. Koo and M. Collins. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*, 2005.

J. Kuhn. Experiments in parallel-text based grammar induction. In *Proc. of ACL*, 2004.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

J. Lafferty, D. Sleator, and D. Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *Proc. of the AAAI Conference on Probabilistic Approaches to Natural Language*, 1992.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4, 1990.

L. Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Harvard University, 1996.

C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9:171–85, 1995.

V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

J. D. Lewis and J. L. Elman. Learnability and the statistical structure of language: Poverty of stimulus arguments revisited. In *Proc. of Boston University Conference on Language Development*, 2001.

D. Lin. On the structural complexity of natural language sentences. In *Proc. of COLING*, 1996.

D. Lin. A dependency-based method for evaluating broad-coverage parsers. In *Proc. of IJCAI*, 1995.

D. Lin. Automatic identification of non-compositional phrases. In *Proc. of ACL*, 1999.

D. Lin and P. Pantel. DIRT—discovery of inference rules from text. In *Proc. of KDD*, 2001.

R. B. Lyngsø and C. N. S. Pedersen. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computing and System Science*, 65:545–569, 2002.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. Available at `http://www.inference.phy.cam.ac.uk/mackay`.

B. MacWhinney. *The CHILDES Database*. Discovery Systems, 3 edition, 1994.

D. Magerman. Statistical decision-tree models for parsing. In *Proc. of ACL*, 1995.

D. M. Magerman and M. P. Marcus. Parsing a natural language using mutual information statistics. In *Proc. of AAAI*, 1990.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

G. F. Marcus. Negative evidence in language acquisition. *Cognition*, 46:53–85, 1993.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.

T. Matsuzaki, Y. Miyao, and J. Tsujii. Probabilsitic CFG with latent annotations. In *Proc. of ACL*, 2005.

J. May and K. Knight. A better $n$-best list: Practical determinization of weighted finite tree automata. In *Proc. of HLT-NAACL*, 2006.

A. McCallum and W. Li. Early results for named-entity extraction with conditional random fields. In *Proc. of CoNLL*, 2003.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*, 2000.

D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proc. of HLT-NAACL*, 2006a.

D. McClosky, E. Charniak, and M. Johnson. Reranking and self-training for parser adaptation. In *Proc. of COLING-ACL*, 2006b.

R. McDonald and F. Pereira. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, 2006.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proc. of ACL*, 2005a.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*, 2005b.

I. D. Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

I. Mel'čuk. *Dependency Syntax: Theory and Practice*. SUNY Press, 1988.

B. Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20 (2):155–72, 1994.

T. Minka. Discriminative models, not discriminative training. Technical Report MSR-TR-2005-144, Microsoft Research, October 2005.

T. Minka. Expectation-maximization as lower bound maximization, 1998.

Y. Miyao and J. Tsujii. Maximum entropy estimation for feature forests. In *Proc. of HLT*, 2002.

M. Mohri, F. C. N. Pereira, and M. Riley. *A Rational Design for a Weighted Finite-State Transducer Library*. Number 1436 in Lecture Notes in Computer Science. Springer, 1998.

J. L. Morgan. *From Simple Input to Complex Grammar*. MIT Press, 1986.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

M.-J. Nederhof and G. Satta. Estimation of consistent probabilistic context-free grammars. In *Proc. of HLT-NAACL*, 2006.

K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proc. of Conference on Information and Knowledge Management*, 2000.

J. Nivre and J. Nilsson. Pseudo-projective dependency parsing. In *Proc. of ACL*, 2005.

J. Nivre and M. Scholz. Deterministic dependency parsing of English text. In *Proc. of COLING*, 2004.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

K. Oflazer, B. Say, D. Z. Hakkani-Tür, and G. Tür. Building a Turkish treebank. In A. Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer, 2003.

M. Osborne and T. Briscoe. Learning stochastic categorial grammars. In *Proc. of CoNLL*, 1997.

M. A. Paskin. Grammatical bigrams. In *Advances in NIPS 15*, 2002.

F. Pereira. Formal grammar and information theory: Together again. *Philosophical Transactions of the Royal Society, Series A*, 358:1239–53, 2000.

F. C. N. Pereira and Y. Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*, 1992.

F. C. N. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proc. of the 31st ACL*, pages 183–190, 1993.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*, 2006.

S. Pinker. How could a child use verb syntax to learn verb semantics? *Lingua*, 92:377–410, 1994.

C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications, 1994.

C. N. Potts and S. L. van de Velde. Dynasearch—iterative local improvement by dynamic programming. part i. the traveling salesman problem. Technical report, University of Twente, The Netherlands, 1995.

G. K. Pullum. Learnability, hyperlearning, and the poverty of the stimulus. In *Proc. of the Annual Meeting of the Berkeley Linguistics Society*, 1996.

G. K. Pullum and B. C. Scholz. Empirical assessment of stimulus poverty arguments. *Linguistic Review*, 19:9–50, 2002.

R. Raina, A. Haghighi, C. Cox, J. Finkel, J. Michels, K. Toutanova, B. MacCartney, M.-C. de Marneffe, C. D. Manning, and A. Y. Ng. Robust textual inference using diverse knowledge sources. In *Proc. of the PASCAL Challenges Workshop on Recognising Textual Entailment*, 2005.

L. Ramshaw, E. Boschee, S. Bratus, S. Miller, R. Stone, R. Weischedel, and A. Zamanian. Experiments in multi-modal automatic content extraction. In *Proc. of HLT*, 2001.

A. Rao and K. Rose. Deterministically annealed design of Hidden Markov Model speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 9(2):111–126, February 2001.

A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing, 1997.

A. Ratnaparkhi. Unsupervised statistical models for prepositional phrase attachment. In *Proc. of COLING*, 1998.

R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2), 1984.

P. Resnik and N. A. Smith. The Web as a parallel corpus. *Computational Linguistics*, 29(3): 349–80, 2003.

S. Riezler. *Probabilistic Constraint Logic Programming*. Ph.D. thesis, Universität Tübingen, 1999.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL*, 2000.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

B. Roark, M. Saraclar, M. Collins, and M. Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. of ACL*, 2004.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. Inducing a semantically annotated lexicon via EM-based clustering. In *Proc. of ACL*, 1999.

K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239, 1998.

K. Rose, E. Gurewitz, and G. C. Fox. Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948, 1990.

D. B. Rubin. The Bayesian bootstrap. *Annals of Statistics*, 9:130–4, 1981.

J. R. Saffran, E. L. Newport, and R. N. Aslin. Word segmentation: The role of distributional cues. *The Journal of Memory and Language*, 35:606–21, 1996.

B. Santorini. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing). Technical report, Linguistic Data Consortium, 1990.

Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. of COLING*, 1992.

G. Schneider. A linguistic comparison of constituency, dependency, and link grammar. Master's thesis, Universität Zürich, 1998.

H. Schütze. Part-of-speech induction from scratch. In *Proc. of ACL*, 1993.

M. S. Seidenberg and M. C. MacDonald. A probabilistic constraints approach to language acquisition and processing. *Cognitive Science*, 23(4):569–88, 1999.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, 2003.

S. Shieber and X. Tao. Comma restoration using constituency information. In *Proc. of NAACL*, 2003.

Y. Shinyama, S. Sekine, K. Sudo, and R. Grishman. Automatic paraphrase acquisition from news articles. In *Proc. of HLT*, 2002.

K. Sima'an. Computational complexity of probabilistic disambiguation. *Grammars*, 5:125–151, 2002.

K. Simov and P. Osenova. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of LINC*, 2003.

K. Simov, G. Popova, and P. Osenova. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In A. Wilson, P. Rayson, and T. McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pages 135–42. Lincom-Europa, 2002.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. Design and implementation of the Bulgarian HPSG-based Treebank. *Journal of Research on Language and Computation*, 2(4): 495–522, 2004a.

K. Simov, P. Osenova, and M. Slavcheva. BulTreeBank morphosyntactic tagset. Technical Report BTB-TR03, BulTreeBank Project, 2004b.

D. Sleator and D. Temperley. Parsing English with a link grammar. In *Proc. of IWPT*, 1993.

D. A. Smith and J. Eisner. Minimum risk annealing for training log-linear models. In *Proc. of COLING-ACL*, 2006a.

D. A. Smith and J. Eisner. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*, 2006b.

D. A. Smith and N. A. Smith. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*, 2004.

G. Smith. A brief introduction to the TIGER Treebank, version 1. Technical report, Universität Potsdam, 2003.

N. A. Smith. From words to corpora: Recognizing translation. In *Proc. of EMNLP*, 2002.

N. A. Smith. Discovering grammatical structure in unannotated text: Implicit negative evidence and dynamic feature selection. Technical report, Johns Hopkins University, 2004.

N. A. Smith and J. Eisner. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*, 2004.

N. A. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*, 2005a.

N. A. Smith and J. Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*, 2005b.

N. A. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proc. of COLING-ACL*, 2006.

N. A. Smith and M. Johnson. Weighted and probabilistic context-free grammars are equally expressive, in review.

N. A. Smith and R. W. Tromble. Sampling uniformly from the unit simplex. Technical report, Johns Hopkins University, 2004.

R. Snow, D. Jurafsky, and A. Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in NIPS 17*, 2004.

Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised context sensitive language acquisition from a large corpus. In *Advances in NIPS 16*, 2003.

S. Soule. Entropies of probabilistic grammars. *Information and Control*, 25(1):57–74, 1974.

M. Steedman. *The Syntactic Process*. MIT Press, 2000.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. Bootstrapping statistical parsers from small datasets. In *Proc. of EACL*, 2003.

A. Stolcke and S. Omohundro. Inducing probabilistic grammars by Bayesian model merging. In *Proc. of ICGI*, 1994.

R. Swanson and A. S. Gordon. A comparison of alternative parse tree paths for labeling semantic roles. In *Proc. of COLING-ACL*, 2006.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in NIPS 16*, 2003.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proc. of EMNLP*, 2004.

B. Taskar, S. Lacoste-Julien, and D. Klein. A discriminative matching approach to word alignment. In *Proc. of HLT-EMNLP*, 2005.

L. Tesnière. *Élément de Syntaxe Structurale*. Klincksieck, 1959.

K. R. Thickman. *Structural and Thermodynamic Analysis of the Essential pre-mRNA Splicing Factor U2AF$^{65}$*. Ph.D. thesis, Johns Hopkins University, 2006.

N. Tishby, F. C. N. Pereira, and W. Bialek. The information bottleneck method. In *Proc.*

*of the 37th Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

E. F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, 2002.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of CoNLL*, 2000.

E. F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, 2003.

E. F. Tjong Kim Sang and H. Déjean. Introduction to the CoNLL-2001 shared task: Clause identification. In *Proc. of CoNLL*, 2001.

A. Törn and A. Zilinskas. *Global Optimization*. Springer-Verlag, 1989.

N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2): 271–282, 1998.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. MMIE training of large vocabulary speech recognition systems. *Speech Communication*, 22(4):303–14, 1997.

M. M. van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. Ph.D. thesis, University of Leeds, 2001.

D. Vergyri. *Integration of Multiple Knowledge Sources in Speech Recognition using Minimum Error Training*. Ph.D. thesis, Johns Hopkins University, 2000.

P. Viola and M. Narasimhan. Learning to extract information from semi-structured text using a discriminative context free grammar. In *Proc. of ACM SIGIR*, 2005.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–9, April 1967.

S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. The latent maximum entropy principle. In *Proc. of ISIT*, 2002.

C. S. Wetherell. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12:361–379, 1980.

C. M. White, I. Shafran, and J.-L. Gauvain. Discriminative classifiers for language recognition. In *Proc. of ICASSP*, 2006.

P. C. Woodland, D. Pye, and M. J. F. Gales. Iterative unsupervised adaptation using maximum likelihood linear regression. In *Proc. of ICSLP*, 1996.

C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1): 95–103, 1983.

F. Xia. The part-of-speech tagging guidelines for the Penn Chinese Treebank (3.0). Technical report, Linguistic Data Consortium, 2000.

L. Xu and M. I. Jordan. On the convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8:129–151, 1996.

L. Xu, D. Wilkinson, F. Southey, and D. Schuurmans. Discriminative unsupervised learning of structured predictors. In *Proc. of ICML*, 2006.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30, 2004.

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*, 2003.

D. Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proc. of ACL*, 1994.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*, 1995.

D. Yarowsky and G. Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proc. of NAACL*, 2001.

D. H. Younger. Recognition and parsing of context-free languages in time $n^3$. *Information and Control*, 10(2), 1967.

D. Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, MIT, 1998.

L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*, 2005.

# Vita

Noah Smith was born in Washington, D.C. in 1979. After graduating with an unofficial "major" in clarinet performance from Glenelg High School in Glenelg, Maryland, in 1997, he completed a B.A. in Linguistics (with honors) and a B.S. in Computer Science (with high honors) at the University of Maryland, College Park, in 2001. He studied informatics at the University of Edinburgh during the first half of 2000. He completed his M.S.Eng. and his Ph.D., both in Computer Science, at Johns Hopkins University in 2004 and 2006, respectively.

Noah has spent time doing research at the Institute for Advanced Computer Studies at the University of Maryland, the Johns Hopkins University summer workshop, West Group's Computer Science research group, New York University, and the Text Mining, Search, and Navigation group at Microsoft Research. His graduate research (academic years 2001–6) was generously supported by a fellowship from the Fannie and John Hertz Foundation.

Noah is on the web at `http://www.cs.cmu.edu/~nasmith`; in the physical world he lives with his wife Karen Thickman and their cats, Sacada and Valentino. Concurrent with the completion of this thesis and Karen's (Thickman, 2006), they relocated to Pittsburgh, Pennsylvania, where Noah is an assistant professor in the Language Technologies Institute and Machine Learning Department, School of Computer Science, at Carnegie Mellon University.

UM... HOLD ON, PROF. SMITH... I KNOW I DID SOME WORK THIS WEEK...
YOU KNOW, SLACKENERNY...
YOU'VE BEEN WORKING FOR ME FOR QUITE A FEW YEARS... THERE'S NO NEED FOR FORMALISM. YOU DON'T HAVE CALL ME "PROF. SMITH"...
...YOU CAN JUST CALL ME "DR. SMITH."
WHATEVER YOU SAY, DR. SMI...
YOU'RE RIGHT. IT'S KIND OF AWKWARD. LET'S GO BACK TO "PROF."
JORGE CHAM ©THE STANFORD DAILY
phd.stanford.edu

From "Piled Higher and Deeper" by Jorge Cham (`http://www.phdcomics.com`, 4/10/2001). Reprinted with permission.