GRAMMAR INDUCTION AND PARSING WITH DEPENDENCY-AND-BOUNDARY MODELS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Valentin Ilyich Spitkovsky
December 2013

Preface

For Dan and Hayden, with even greater variances...

Unsupervised learning of hierarchical syntactic structure from free-form natural language text is an important and difficult problem, with implications for scientific goals, such as understanding human language acquisition, or engineering applications, including question answering, machine translation and speech recognition. As is the case with many unsupervised settings in machine learning, grammar induction usually reduces to a non-convex optimization problem. This dissertation proposes a novel family of head-outward generative dependency parsing models and a curriculum learning strategy, co-designed to effectively induce grammars despite local optima, by taking advantage of multiple views of data.

The dependency-and-boundary models are parameterized to exploit, as much as possible, any observable state, such as words at sentence boundaries, which limits the proliferation of optima that is ordinarily caused by presence of latent variables. They are also flexible in their modeling of overlapping subgrammars and sensitive to different kinds of input types. These capabilities allow training data to be split into simpler text fragments, in accordance with proposed parsing constraints, thereby increasing the numbers of visible edges. An optimization strategy then gradually exposes learners to more complex data.

The proposed suite of constraints on possible valid parse structures, which can be extracted from unparsed surface text forms, helps guide language learners towards linguistically plausible syntactic constructions. These constraints are efficient, easy to implement and applicable to a variety of naturally-occurring partial bracketings, including capitalization changes, punctuation and web markup. Connections between traditional syntax and

HTML annotations, for instance, were not previously known, and are one of several discoveries about statistical regularities in text that this thesis contributes to the science linguistics.

Resulting grammar induction pipelines attain state-of-the-art performance not only on a standard English dependency parsing test bed, but also as judged by constituent structure metrics, in addition to a more comprehensive multilingual evaluation that spans disparate language families. This work widens the scope and difficulty of the evaluation methodology for unsupervised parsing, testing against nineteen languages (rather than just English), evaluating on all (not just short) sentence lengths, and using disjoint (blind) training and test data splits. The proposed methods also show that it is possible to eliminate commonly used supervision signals, including biased initializers, manually tuned training subsets, custom termination criteria and knowledge of part-of-speech tags, and still improve performance.

Empirical evidence presented in this dissertation strongly suggests that complex learning tasks like grammar induction can cope with non-convexity and discover more correct syntactic structures by pursuing learning strategies that begin with simple data and basic models and progress to more complex data instances and more expressive model parameterizations. A contribution to artificial intelligence more broadly is thus a collection of search techniques that make expectation-maximization and other optimization algorithms less sensitive to local optima. The proposed tools include multi-objective approaches for avoiding or escaping fixed points, iterative model recombination and "starting small" strategies that gradually improve candidate solutions, and a generic framework for transforming these and other already-found locally optimal models. Such transformations make for informed, intelligent, non-random restarts, enabling the design of comprehensive search networks that are capable of exploring combinatorial parameter spaces more rapidly and more thoroughly than conventional optimization methods.

Dedicated to my loving family.

Acknowledgements

I must thank many people who have contributed to the successful completion of this thesis, starting with the oral examination committee members: my advisor, Daniel S. Jurafsky, for his patience, encouragement and wisdom; Hayden Shaw, who has been a close collaborator at Google Research and a de facto mentor; Christopher D. Manning, from whom I learned NLP and IR; Serafim Batzoglou, with whom I coauthored my earliest academic paper; and Arthur B. Owen, who became my first guide into the world of Statistics. I am also grateful to others — professors, coauthors, coworkers, labmates, other collaborators, anonymous reviewers, recommendation letter writers, graduate program coordinators, friends, well-wishers, as well as the numerous dance, martial arts, and yoga instructors who helped to keep me (relatively) sane through it all. Any attempt to name everyone is doomed to failure.

Here are the results of one such effort: Omri Abend, Eneko Agirre, Lauren Anas, Kelly Ariagno, Michael Bachmann, Prachi Balaji, Edna Barr, Alexei Barski, John Bauer, Steven Bethard, Yonatan Bisk, Anna Botelho, Stephen P. Boyd, Thorsten Brants, Helen Buendicho, Jerry Cain, Luz Castineiras, Daniel Cer, Nathanael Chambers, Cynthia Chan, Angel X. Chang, Helen Chang, Ming Chang, Pi-Chuan Chang, Jean Chao, Wanxiang Che, Johnny Chen, Renate & Ron Chestnut, Rich Chin, Yejin Choi, Daniel Clancy, John Clark, Ralph L. Cohen, Glenn Corteza, Chris Cosner, Luke Dahl, Maria David, Amir Dembo, Persi Diaconis, Kathi DiTommaso, Lynda K. Dunnigan, Jason Eisner, Song Feng, Jenny R. Finkel, Susan Fox, Michael Genesereth, Suvan Gerlach, Kevin Gimpel, Andy Golding, Leslie Gordon, Bill Graham, Spence Green, Sonal Gupta, David L.W. Hall, Krassi Harwell, Cynthia Hayashi, Julia Hockenmaier, John F. Holzrichter, Anna Kazantseva, Carla Murray Kenworthy, Steven P. Kerckhoff, Sadri Khalessi, Jam Kiattinant, Donald E. Knuth, Daphne Koller, Mikhail Kozhevnikov, Linda Kubiak, Polina Kuznetsova, Cristina N. & Homer G.

Ladas, Diego Lanau, Leo Landa, Beth Levin, Eisar Lipkovitz, Claudia Lissette, Ting Liu, Sherman Lo, Gabby Magana, Kim Marinucci, Marie-Catherine de Marneffe, Felipe Martinez, Andrea McBride, David McClosky, Ryan McDonald, R. James Milgram, Elizabeth Morin, Rajeev Motwani, Andrew Y. Ng, Natasha Ng, Barbara Nichols, Peter Norvig, Ingram Olkin, Jennifer Olson, Nick Parlante, Marius Paşca, Fernando Pereira, Lisette Perelle, Daniel Peters, Leslie Miller Peters, Slav Petrov, Ann Marie Pettigrew, Keyko Pintz, Daniel Pipes, Igor Polkovnikov, Elias Ponvert, Zuby Pradhan, Agnieszka Purves, Bala Rajaratnam, Daniel Ramage, Julian Miller Ramil, Marta Recasens, Roi Reichart, David Rogosa, Joseph P. Romano, Mendel Rosenblum, Vicente Rubio, Roy Schwartz, Xiaolin Shi, David O. Siegmund, Noah A. Smith, Richard Socher, Alfred Spector, Yun-Hsuan Sung, Mihai Surdeanu, Julie Tibshirani, Ravi Vakil, Raja Velu, Pier Voulkos, Mengqiu Wang, Tsachy Weissman, Jennifer Widom, Verna L. Wong, Adrianne Kishiyama & Bruce Wonnacott, Lowell Wood, Wei Xu, Eric Yeh, Ayano Yoneda, Alexander Zeyliger, Nancy R. Zhang, and many others of Stanford's Natural Language Processing Group, the Computer Science, Electrical Engineering, Linguistics, Mathematics, and Statistics Departments, Aikido and Argentine Tango Clubs, as well as the Fannie & John Hertz Foundation and Google Inc.

Portions of the work presented in this dissertation were supported, in part, by research grants from the National Science Foundation (award number IIS-0811974) and the Defense Advanced Research Projects Agency's Air Force Research Laboratory (contract numbers FA8750-09-C-0181 and FA8750-13-2-0040), as well as by a graduate Fellowship from the Fannie & John Hertz Foundation. I am grateful to these organizations, and to the organizers of TAC, NAACL, LREC, ICGI, EMNLP, CoNLL and ACL conferences and workshops. Last but certainly not least, I thank Jayanthi Subramanian in the Ph.D. Program Office and Ron L. Racilis at the Office of the University Registrar, who sprung into action when, at the last moment, one of the doctoral dissertation reading committee members left the state before conferring his approval with a physical signature. It's been an exciting ride until the very end. I am grateful to (and for) all those who stood by me when life threw us curves.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Parsing free-form text is a core task in natural language processing. Written sentences and speech-transcribed utterances are usually stored in a computer's memory as character sequences. However, this simple representation belies the rich linguistic structure that permeates language. Correctly identifying hierarchical substructures, from the parts-of-speech of individual words to phrasal and clausal bracketings of multi-word spans (see Figure 1.1), is indispensable for many applications of computational linguistics. Coreference resolution [139, 184], semantic role labeling [116, 328] and relation extraction [135, 221] are just a few of the important problems that depend on the information in syntactic parse trees. Unfortunately high quality parsers are not available for most languages, since manually

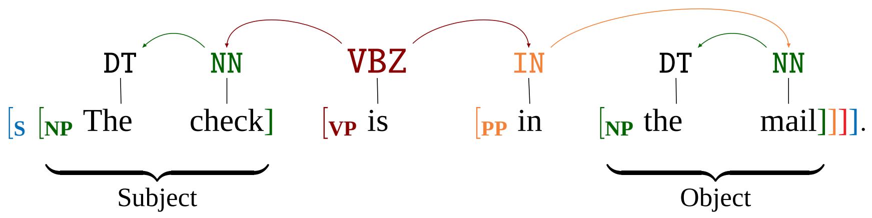


Figure 1.1: A syntactic annotation of the running example sentence, including (i) individual word tokens' parts-of-speech (POS), which can be determiners (`DT`), adjectives (`JJ`), nouns (`NN`), prepositions (`IN`), verbs (`VBZ`), etc.; (ii) a bracketing that shows how words are arranged into coherent chunks, i.e., the noun (`NP`), prepositional (`PP`) and verb phrases (`VP`), which culminate in a simple declarative clause (`S`) that spans the input text in its entirety; and (iii) lexical head words of the constituents, i.e., the main nouns, preposition and verb of the corresponding phrases, as well as the head verb (*is*) that derives the full sentence.

specifying comprehensive parsing rules, or constructing large reference treebanks from which valid grammatical productions could be extracted statistically, is an extremely time, labor and money-intensive process. Even where modern supervised parsers are available they tend not to generalize well out-of-domain, for example from traditional news-style data to biomedical text [210]. Nevertheless the ability to parse understudied and low-resource languages, in addition to non-standard genres like scientific writing, legalese and web text, is a crucial prerequisite to exploiting any higher-level NLP components, which have come to rely on good quality parses for their success, in these important domains. Partly because it may not be feasible to thoroughly annotate structure for most genres of most languages, fully-unsupervised parsing and grammar induction [44, 82, 345] emerged as active research areas, alongside more traditional semi-supervised and domain adaptation methods, but further distinguished by a possible connection to human language acquisition.

Many standard grammatical formalisms and parsing styles have been used as vehicles for inducing syntactic structure, including constituency [245, 82, 171, 34], dependency [44, 345, 244, 172, 133] and combinatory categorial grammars [30, 31], or CCG, for which reference treebanks already exist from the manual annotation efforts in the supervised parsing settings, as well as tree-substitution grammars [33, 68] and other representations [299, 283]. I chose to work within a simple dependency parsing framework, where the task, given a sentence (e.g., *The check is in the mail.*), is to identify its root word (i.e., *is*), along with the parents of all other (non-root) words (i.e., *check* for *The*, *is* for *check*, *is* for *in*, *mail* for *the*, and *in* for *mail* — see Figure 1.1). If we restrict attention only to well-formed parses, the task becomes equivalent to finding a spanning tree, taking the input tokens as vertices of a graph [212]. This representation had become a dominant paradigm for grammar induction following Klein and Manning's publication of the dependency model with valence [172], or DMV, which I describe in the next chapter (see Ch. 2). Resulting unlabeled dependency edges are, arguably, closer to semantics and capturing meaning [10] than the output of many other syntactic formalisms, such as unlabeled constituents. At the same time, dependency grammars present a light-weight framework that, although shallower than CCG, is also easier to induce and faster to parse. This representation is therefore not only relevant to the more general problem of language understanding but also strikes the correct balance for certain important applications that motivate grammar induction in industry. Prime examples include: (i) information retrieval and web search [38, 125], where distances between words in dependency parse trees may work as better indicators of proximity than their nominal sequential positioning in surface text [45, 181]; (ii) question answering [101], where (once again, dependency) parses of natural language questions [248] are transformed to match the structure of corresponding hypothetical sentences that may contain an answer; and (iii) syntax-aware statistical machine translation [340], in which one side of a parallel corpus is sometimes "pre-ordered" to better match the other side of a bitext [164], for example, from the subject-verb-object (SVO) word order of English to subject-object-verb (SOV) in Japanese, simply by pushing main verbs to ends of sentences, or to object-subject-verb (OSV) of "Yoda-speak," by also rearranging the arguments of root words: *In the mail the check is.* In situations where constituent parses are preferred, the weak equivalence between phrase and dependency structures [337] could be exploited to obtain the corresponding unlabeled bracketings, such as

$$[[\textit{The } \underline{\textit{check}}] [\underline{\textit{is}} [\underline{\textit{in}} [\textit{the } \underline{\textit{mail}}]]]].$$

The DMV ushered a breakthrough in unsupervised dependency parsing performance, for the first time beating both left- and right-branching baselines, which simply connect adjacent words. Many of the state-of-the-art results that followed Klein and Manning's seminal publication were also based on their model [295, 65, 133, 66, 117, 33]. For this reason, I began by replicating the core DMV architecture, inheriting many of its simplifying assumptions. These included: (i) using POS tags as word categories [44], in place of actual words; (ii) imposing a projective parsing model to generate these tokens [7, 8, 244],[1] efficiently learnable via inside-outside re-estimation [89, 243]; and (iii) processing all sentences independently. The above simplifications are, of course, mere heuristics and don't always hold. Lexical items will often contain important semantic information that could facilitate parsing in a way that coarse syntactic categories cannot. A minority of correct dependency parse trees will be non-projective, with some dependency arcs crossing, hence unattainable by the DMV. Expectation-maximization (EM) algorithms [83, 14] for grammar induction

[1]In a *projective* parse structure, the yield of any syntactic head is required to be continuous [176]: more specifically, a dependency graph is projective precisely when an edge from ⓐ to ⓩ implies the existence of a directed path from ⓐ also to all of the intervening words that lie between them in a sentence [211, §1.3.1].

will get stuck in local optima, requiring careful initialization and/or restarts. And the syntactic roles played by words in nearby sentences will tend to be correlated [270]. Despite these clear deficiencies, the DMV has stood the test of time as a robust platform for getting grammar inducers off the ground. Taking a cue from this success story, the work presented in this thesis further strengthens independence assumptions, for example splitting sentences on punctuation and processing the resulting pieces separately. Focusing on simple examples, such as short sentences and incomplete text fragments, helps guide unsupervised learning, mirroring the well-known effect that boosting hard examples has in supervised training [108]. And unlike in supervised parsing, where one popular trend has been to introduce more complex models, with specialized priors to prevent overfitting [156], the over-arching theme of this work on grammar induction is to employ extremely simple parsing models, but coupled with strong, hard constraints, to guard against *underfitting*.

The research described in this dissertation followed a two-phase trajectory. In the first phase, I took apart the DMV set-up, trying to understand which pieces worked, which didn't and why. In the second phase, I used the insights obtained from my experience in the first phase to improve the working components and to design more effective grammar induction models and pipelines around them. Some of the known weak points in the DMV set-up include its sensitivity to local optima and choice of initializer [113, §6.2]. Part I of this thesis therefore focuses on optimization strategies that either don't require initialization (Ch. 3) or work well with uninformed, uniform-at-random initializers (Ch. 4), as well as strategies for avoiding and escaping local optima (Ch. 5). The DMV's "ad-hoc harmonic" initializer, whose stated goal was "to point the model in the vague general direction of what linguistic dependency structures should look like," is only one of many kinds of universal knowledge that could be baked into a grammar inducer. In that vein, Smith and Eisner [295] further emphasized structural locality biases; Seginer [283, 284] made use of the facts that humans process most sentences in linear time, that parse trees tend to be skewed, and that words follow a Zipfian distribution; Gillenwater et al. [117] exploited the realized sparsity in the quadratic space of possible word-word interactions; and my early attempt to understand the power laws of harmonic initializers yielded an additional, novel observation: dependency arc lengths are log-normally distributed (Ch. 2). Leveraging such biases can be troublesome, however, since the exact parameters of soft universal properties have typically been optimized for English or fitted to treebanks, rather than learned from text. Part II of this thesis therefore focuses on identifying reliable sources of hard constraints on parse trees that can be mined for naturally-occurring partial bracketings [245], making explicit the connection between linguistic structure and web markup (Ch. 6, the first work to explore such a connection), punctuation (Ch. 7), and capitalization (Ch. 8), to augment the projectivity restrictions that are implicitly enforced by head-outward generative parsing models.

One of the biggest questions that this dissertation aims to answer is the extent to which supervision is truly necessary for grammar induction. To this end, Part III begins by showing how word categories based on gold parts-of-speech, on which the entire dependency grammar induction field had been relying for state-of-the-art performance since 2004,[2] when the lexicalized system of Paskin [244] was revealed to score below chance [172], can be replaced by fully-unsupervised word clusters and still improve results (Ch. 9).[3] This is a key contribution, since assuming knowledge of parts-of-speech is not only unrealistic from the language acquisition perspective but also an inefficient use of the syntactic information that these tags contain: at around the same time, in 2011, McDonald et al. [213] showed how universal part-of-speech categories [249] can be exploited to transfer delexicalized parsers across languages, resulting in a stronger alternative solution to the unsupervised parsing problem than grammar induction from gold tags. The remainder of Part III covers dependency-and-boundary models (DBMs), which heavily exploit any available information about structure that is *not* latent, for example at sentence boundaries — another key contribution of this thesis. DBMs are novel head-outward generative parsing models and can be learned via simple curricula (Ch. 10) that don't require knowing manually tuned training length cut-offs (e.g., "up to length ten" from the DMV set-up). They can also be bootstrapped from inter-punctuation fragments (Ch. 11), which vastly increases the number of visible edges being exploited, as well as the overall amount of simple text made available

[2]A notable exception is the work of Seginer [283, 284] whose incremental "common cover link" (CCL) parser is trained from raw text, without discarding long sentences or punctuation. His contribution was carefully analyzed by Ponvert et al. [253, 254], who determined that the CCL parser is, in fact, an excellent unsupervised chunker, and that better (constituent) parsers could be constructed simply by hooking up the lowest-level bracketings induced by CCL into a linear chain. Their thorough analysis attributed CCL's success at finding word clumps specifically to how punctuation marks are incorporated in its internal representations.

[3]It is important to mention here that, unlike most work that followed the DMV, which does not report performance with unsupervised tags, Klein and Manning's 2004 paper does include results that rely only on word clusters, which are worse than their state-of-the-art results with gold POS tags [172, Table 6: English].

to the earliest phases of learning. Splitting sentences on punctuation is a natural next step in the progression of hard constraints based on punctuation from Part II, which quantifies the strength of correlations between punctuation marks and phrase structure, and also in the exploration of initialization strategies from Part I, which first demonstrates the power of starting from simpler and easier input data. Every chapter in parts I–III, chapters 3–11, corresponds to a peer-reviewed publication. The final part, Part IV, consists of a single additional chapter that integrates the majority of this dissertation's contributions to the field in a modular state-of-the-art grammar induction pipeline (Ch. 12); this tenth article, which can be viewed as a culmination of the entire thesis, received a "best paper" award at the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).

My efforts, throughout the thesis, to minimize the amount of prior knowledge built into grammar inducers boil it down to knowing about punctuation and projectivity.[4] Having eliminated POS, I found that it can be useful to view sentences not only as sequences of word categories [44, 172], which can be crucial in the earliest training phases, but also as actual words [345, 244], which filters out any clustering noise and further allows for simple and precise system combination via mixture models (Ch. 12). But the lexicalized versus unlexicalized distinction is just one dichotomy — a narrow band in the rich spectrum of the grammar induction typology. For instance, though it is common to use chart-based parsing methods, as I had, it is also possible to induce grammars with (left-to-right) incremental parsers [283, 79, 259]. In addition, the underlying models themselves can be not only simple, generative, projective and learned via EM, as in this work, but also feature-rich, discriminative, and non-projective [212, 238, 239], as in supervised settings, learned via sampling methods like MCMC [33, 227, 202] or gradient-based optimizers like L-BFGS [24]. Rather than explore all such alternative possibilities myself, I show (Chs. 5, 12) how the existence of these and other views [32] of a learning problem can be exploited, again and again, to systematically fight the challenges posed by non-convexity of objective functions.

Part IV really introduces a *framework* for designing comprehensive search networks and may therefore be the biggest contribution of this dissertation, as it applies not just to grammar induction but any areas where non-convex optimization and local search problems

[4]In addition to sentence and token boundaries, which could themselves have been induced from raw text, along with the identities of the tokens that represent punctuation marks, as part of *unsupervised tokenization*.

arise. Its primitive modules are the individual local optimizers, system combination and several other entirely generic methods for intelligently finding places to restart local search, informed by already-discovered locally-optimal solutions, such as model ablation, data-set filtering and self-training. The trouble with unsupervised learning in general [97, 219, 189], and grammar induction in particular [245, 82, 119], is frequently having to optimize against a likelihood objective that is not only plagued by local extrema, which is enough to make research frustrating and its replication inconvenient, but also a poor proxy for extrinsic performance, like parsing accuracy. This last fact is both depressing and liberating, for it justifies, on occasion, ignoring the moves proposed by a local optimizer, treating them as mere suggestions, to help a non-convex optimization process make progress. Part I culminates in several “lateen EM” strategies (Ch. 5) that zig-zag around local attractors, for example by switching between ordinary “soft” and “hard” EM algorithms. The basic idea is simple: if one flavor of EM stalls, use the other to dig it out, in a way that doesn’t undo all previous work; a faster and more practical approach, which strives to avoid getting close to local optima in the first place, is to validate proposed moves, switching when improving one EM’s objective would harm another’s. Lateen EM thus leverages the fact that two views of data, as sentence strings (soft EM) or as their most likely parse trees (hard EM), yield different equi-plausible unsupervised objective functions. Part IV formalizes the various ways in which other views of data can be similarly exploited to break out of local optima.

Chapter 2

Background

This thesis continues a line of grammar induction research that was sparked by the famous experiments of Carroll and Charniak [44, Footnote 1], who credit Mark Johnson for sharing with them Martin Kay's suggestion to use a *dependency* schema. The idea was to bound the number of possible valid productions that might participate in the derivation of a sentence by restricting the set of non-terminal symbols to its words. In practice, the space of grammatical rules had to be further reduced to a more manageable size, by replacing words with their parts-of-speech and emphasizing short sentences [44, Footnote 2]. A resulting dependency grammar was then cast as a one-bar-level X-bar [59, 149] constituency grammar, so that its rules' probabilities could be learned efficiently via inside-outside re-estimation [14], an instance of the EM algorithm [83], by locally maximizing the likelihood of a text corpus.

Subsequent research focused on *split-head* dependency grammars (under various names), which also allow for efficient implementations of the inside-outside algorithm, due to Eisner and Satta [91, §8]. These grammars correspond to a special case of the head-outward automata for producing dependency parse trees proposed by Alshawi [7, 6, 9]. Their generative stories begin by selecting a root word, e.g., *is* (see Figure 1.1), with some probability. Each generated word then recursively initiates a new chain of probabilistic state transitions in an automaton that simulates a head word spawning off dependents, i.e., *is* attaching *check* to its left and *in* to its right, away from itself. If the automaton associated to *is* were to spawn off an additional dependent prior to entering a stopping state, that dependent would have to lie either to the left of *check* or to the right of *in*; instead, *is* stops after just two dependents, *check* attaches only *The*, *in* attaches only *mail*, *mail* attaches only *the*, and the two determiners, *The* and *the*, stop without generating any children. Models equivalent to head-outward automata, restricted to the split-head case,[1] in which each head generates left- and right-dependents separately, have been central to many generative parsing systems. One of their earlier manifestations was in supervised "head-driven" constituent parsers [69, 71].

Among unsupervised models, a 2001 system [243, 244] was the first to learn locally-optimal probabilities from naturally occurring monolingual text (and did not rely on POS).[2] Paskin used a rudimentary grammar, in which root words were chosen uniformly at random, and whose equivalent split-head automata could be thought of as having just two states (see Figure 2.1), with an even chance of leaving the (split) starting states for a stopping state. The only learned state transition parameters in this "grammatical bigrams" model are pair-wise word-word probabilities, $\{\gamma_{\overleftarrow{dh}}\}$ and $\{\gamma_{\overrightarrow{hd}}\}$, of spawning a particular dependent d upon taking a self-loop to stay in a generative state, conditioned on identities of the head word h and side (left or right) of the path taken in its associated automaton. Although the machine learning behind the approach is sound, dependency parse trees induced by Paskin's system were less accurate than random guessing [172]. Its major stumbling blocks were, most likely, due to starting from specific words,[3] instead of generalized word categories (see Ch. 9) — and all sentences with soft EM instead of just the short inputs (see Ch. 3) or hard EM (see Ch. 4) — and *not* because of the extremely simple parsing model (see Ch. 11).

The dependency model with valence operates over word *classes*, i.e., $\{c_h\}$, instead of raw lexical items $\{h\}$, and is therefore more compact than "grammatical bigrams," drawing on Carroll and Charniak's [44] work from 1992. In addition to aggregating the lexicalized bigram parameters $\{\gamma\}$ according to POS tags, the DMV can be viewed as using slightly larger, three-state automata (see Figure 2.2). Furthermore, Klein and Manning introduced explicit parameters (which I labeled as $\{\alpha\}$ and $\{\beta\}$ in the automata diagram) to capture the

[1]Unrestricted head-outward automata are strictly more powerful (e.g., they recognize the language $a^n b^n$ in finite state) than the split-head variants, which can be thought of as processing one side before the other.

[2]Though several years prior, Yuret [345] had used mutual information to guide greedy linkage of words; and the head automaton models trained by Alshawi et al. [9, 11] also estimated such probabilities, with two sets of parameters being learned simultaneously, using bitexts, in a fully-unsupervised fashion, from words.

[3]Alshawi et al. [9, 11] could get away with using actual words in their head-outward automata because they were performing *synchronous* grammar induction, with the bitext constraining both learning problems.

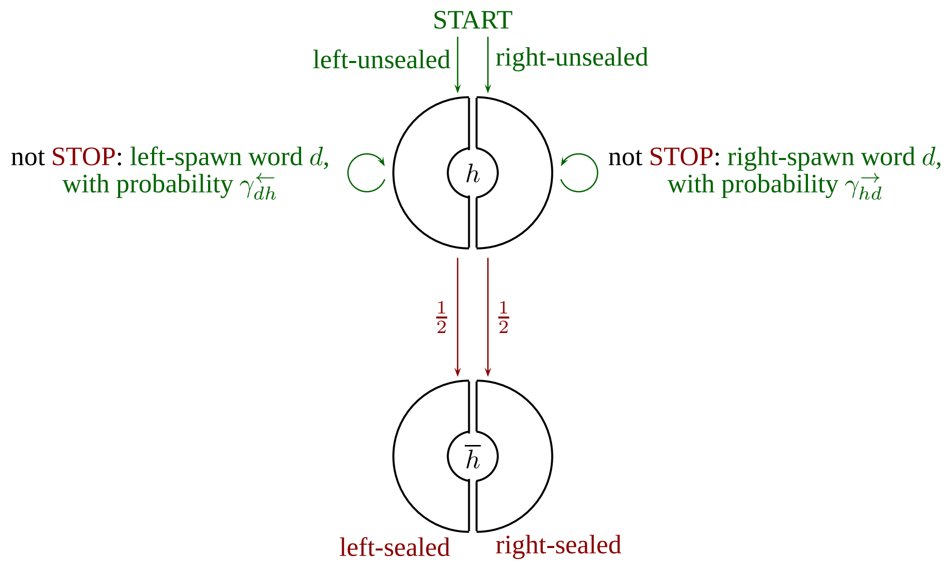

START
left-unsealed
right-unsealed
not STOP: left-spawn word d, with probability $\gamma_{dh}^{\leftarrow}$
h
not STOP: right-spawn word d, with probability $\gamma_{hd}^{\rightarrow}$
$\frac{1}{2}$
$\frac{1}{2}$
$\overline{h}$
left-sealed
right-sealed

Figure 2.1: Paskin's "grammatical bigrams" as head-outward automata (for head words h).

linguistic notion of *valency* [319, 104]: "adjacent" stopping probabilities (binomial parameters $\{\alpha\}$) capture the likelihood that a word will not spawn any children, on a particular side (left or right); and the "non-adjacent" probabilities (geometric parameters $\{\beta\}$) encode the tendency to stop after at least one child has been generated on that side. With the extra state, POS tags and a more fleshed out parameterization of the automata, which I describe in more traditional detail, including the required initializer, in the next section, the DMV could beat baseline performance for an important subcase of grammar induction, motivated by language acquisition in children: text corpora limited to sentences up to length ten.

Klein and Manning experimented with both gold part-of-speech tags and unsupervised English word clusters. Despite their finding that the unsupervised tags performed significantly worse, much of the work that followed chose to adopt the version of the task that assumes knowledge of POS, perhaps expecting that improvements in induction from raw words rather than gold tags would be orthogonal to other advances in unsupervised dependency parsing. Yet several research efforts focused specifically on exploiting syntactic information in the gold tags, e.g., by manually specifying universal parsing rules [228] or statistically tying parameters of grammars across different languages [66], shifting the focus away from grammar induction proper. One of the main proposed contributions of this

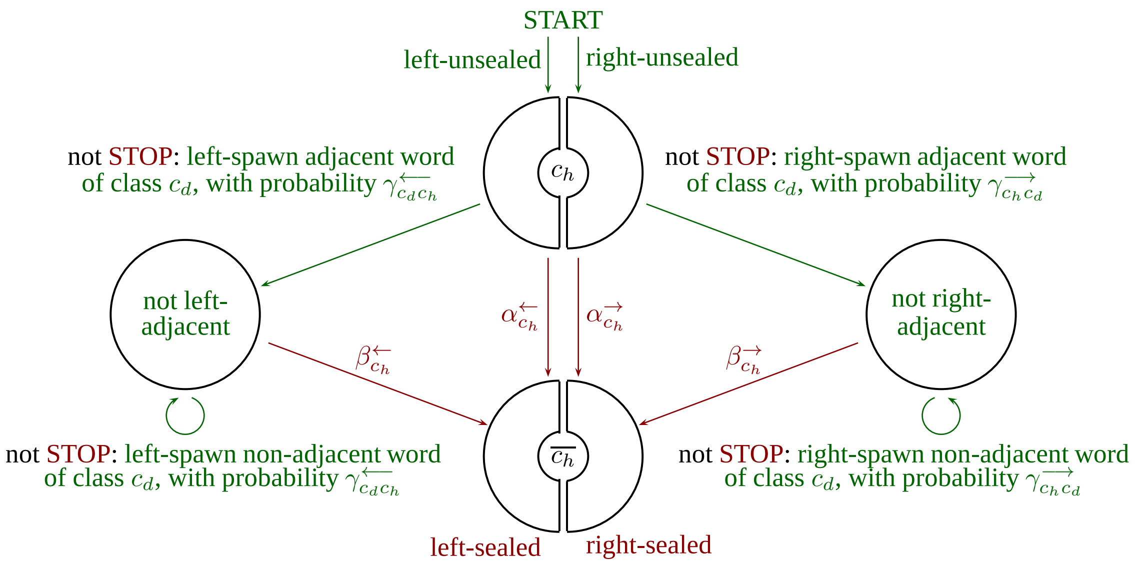
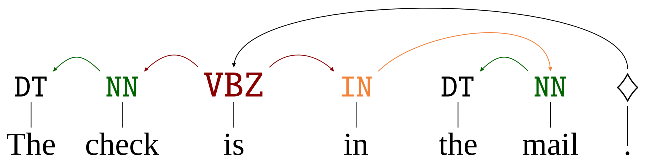

START
left-unsealed
right-unsealed
c_h
not STOP: left-spawn adjacent word of class c_d, with probability $\gamma_{c_d c_h}^{\leftarrow}$
not STOP: right-spawn adjacent word of class c_d, with probability $\gamma_{c_h c_d}^{\rightarrow}$
not left-adjacent
not right-adjacent
$\alpha_{c_h}^{\leftarrow}$
$\alpha_{c_h}^{\rightarrow}$
$\beta_{c_h}^{\leftarrow}$
$\beta_{c_h}^{\rightarrow}$
$\overline{c_h}$
not STOP: left-spawn non-adjacent word of class c_d, with probability $\gamma_{c_d c_h}^{\leftarrow}$
not STOP: right-spawn non-adjacent word of class c_d, with probability $\gamma_{c_h c_d}^{\rightarrow}$
left-sealed
right-sealed

Figure 2.2: Klein and Manning's dependency model with valence (DMV) as head-outward automata (for head words of class c_h). A similar diagram could depict Headden et al.'s [133] extended valence grammar (EVG), by using a separate set of parameters, $\{\delta\}$ instead of $\{\gamma\}$, for the word-word attachment probabilities in the self-loops of non-adjacent states.

dissertation to methodology, as already mentioned in the previous chapter, is to show how state-of-the-art results can be attained using fully unsupervised word clusters. Other important methodological contributions address the evaluation of unsupervised parsing systems.

Since the DMV did not include smoothing, Klein and Manning tested their unsupervised parsers on the training sets, i.e., sentences up to length ten in the input. Most work that followed also evaluated on short data, which can be problematic for many reasons, including overfitting to simple grammatical structures, higher measurement noise due to smaller evaluation sets, and overstated results, since short sentences are easier to parse (left- and right-branching baselines can be much more formidable at higher length cutoffs [127, Figure 1]). Although a child may initially encounter only basic utterances, an important goal of language acquisition is to enable the comprehension of previously unheard and complex speech. The work in this dissertation therefore tests on both short and long sentence lengths and uses held-out evaluation sets, such as the parsed portion of the Brown corpus [106], when training on text from the Wall Street Journal [200]. Furthermore, since children are expected to be able to acquire arbitrary human languages, it is important to make sure that a grammar inducer similarly generalizes, to avoid accidentally over-engineering to a particular language and genre. Consequently, many of the systems in this thesis are also evaluated against all 19 languages of the 2006/7 CoNLL test sets [42, 236], essentially treating English WSJ as development data. Indeed, work presented in this dissertation is some of the earliest to call for this kind of evaluation: *all* languages, *all* sentences and *blind* test sets.

2.1 The Dependency Model with Valence



$$
\begin{aligned}
\mathbb{P} \;=\; & (1-\overbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{L};\ \texttt{T})}^{0}) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{VBZ} \mid \diamond;\ \texttt{L}) \\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T}, \texttt{VBZ})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{NN} \mid \texttt{VBZ};\ \texttt{L}) \\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T}, \texttt{VBZ})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{IN} \mid \texttt{VBZ};\ \texttt{R}) \\
\times\ & \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F}, \texttt{VBZ}) && \times\ \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{F}, \texttt{VBZ}) \\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T}, \texttt{NN}))^2 && \times\ \mathbb{P}^2_{\texttt{ATTACH}}(\texttt{DT} \mid \texttt{NN};\ \texttt{L}) \\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T}, \texttt{IN})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{NN} \mid \texttt{IN};\ \texttt{R}) \\
\times\ & \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T}, \texttt{NN}) && \times\ \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F}, \texttt{NN}) \\
\times\ & \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T}, \texttt{IN}) && \times\ \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{F}, \texttt{IN}) \\
\times\ & \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T}, \texttt{DT}) && \times\ \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T}, \texttt{DT}) \\
\times\ & \underbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{L};\ \texttt{F})}_{1} && \times\ \underbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{R};\ \texttt{T})}_{1}.
\end{aligned}
$$

Figure 2.3: A dependency structure and its probability, as factored by the DMV.

The DMV is a simple head automata model over lexical word classes $\{c_w\}$ — POS tags. Its generative story for a subtree rooted at a head (of class c_h) rests on three types of independent decisions: (i) initial direction $dir \in \{\texttt{L}, \texttt{R}\}$ (left or right) in which to attach children, via probability $\mathbb{P}_{\texttt{ORDER}}(c_h)$; (ii) whether to seal dir, stopping with probability $\mathbb{P}_{\texttt{STOP}}(c_h, dir, adj)$, conditioned on $adj \in \{\texttt{T}, \texttt{F}\}$ (true only when considering dir's first, i.e., *adjacent*, child); and (iii) attachment of a particular dependent (of class c_d), according to $\mathbb{P}_{\texttt{ATTACH}}(c_h, dir, c_d)$. This process produces only projective trees. By convention [93], a root token ♢ generates the head of a sentence as its left (and only) child. Figure 2.3 displays an example that ignores (sums out) $\mathbb{P}_{\texttt{ORDER}}$, for the short running example sentence.

The DMV was trained by re-estimating without smoothing, starting from an "ad-hoc harmonic" completion: aiming for balanced trees, non-root head words attached dependents in inverse proportion to (a constant plus) their distance; ♢ generated heads uniformly at random. This non-distributional heuristic created favorable initial conditions that nudged learners towards typical linguistic dependency structures. In practice, 40 iterations of EM was usually deemed sufficient, as opposed to waiting for optimization to actually converge.

Although the DMV is described as a *head-outward* model [172, §3], the probabilities that it assigns to dependency parse trees are, in fact, invariant to permutations of siblings on the given side of a head word. Naturally, the same is also true of "grammatical bigrams" and the EVG (see Figures 2.1–2.2). Dependency-and-boundary models that I introduce in Part III (Chs. 10–11) will be more sensitive to the ordering of words in input.

2.2 Evaluation Metrics

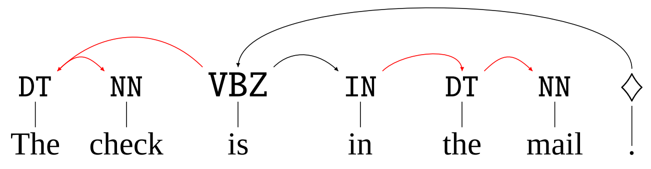


Figure 2.4: A dependency structure that interprets determiners as heads of noun phrases. Four of the six arcs in the parse tree are wrong (in red), resulting in a directed score of 2/6 or 33.3%. But two of the incorrect dependencies connect the right pairs of words, determiners and nouns, in the wrong direction. Undirected scoring grants partial credit: 4/6 or 66.7%.

The standard way to judge a grammar inducer is by the quality of the single "best" parses that it chooses for each sentence: a *directed* score is then simply the fraction of correctly guessed (unlabeled) dependencies; a more flattering *undirected* score is also sometimes used (see Figure 2.4). Ignoring polarity of parent-child relations can partially obscure effects of alternate analyses (systematic choices between modals and main verbs for heads of sentences, determiners for noun phrases, etc.) and facilitated comparisons of the DMV

Corpus	*Sentences*	*POS Tokens*
WSJ1	159	159
WSJ2	499	839
WSJ3	876	1,970
WSJ4	1,394	4,042
WSJ5	2,008	7,112
WSJ6	2,745	11,534
WSJ7	3,623	17,680
WSJ8	4,730	26,536
WSJ9	5,938	37,408
WSJ10	7,422	52,248
WSJ11	8,856	68,022
WSJ12	10,500	87,750
WSJ13	12,270	110,760
WSJ14	14,095	136,310
WSJ15	15,922	163,715
WSJ20	25,523	336,555
WSJ25	34,431	540,895
WSJ30	41,227	730,099
WSJ35	45,191	860,053
WSJ40	47,385	942,801
WSJ45	48,418	986,830
WSJ100	49,206	1,028,054
Section 23	2,353	48,201
Brown100	24,208	391,796

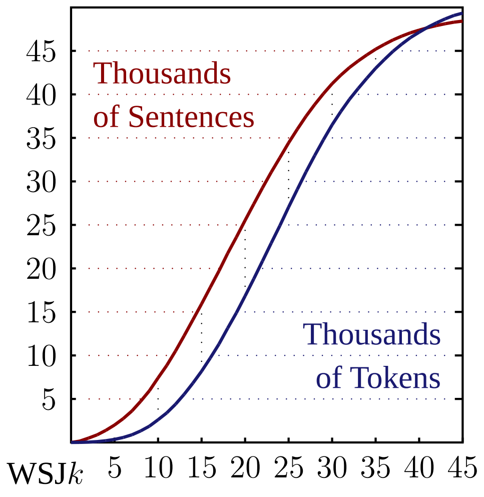


Figure 2.5: Sizes of WSJ$\{1, \dots, 45, 100\}$, Section 23 of WSJ$^\infty$ and Brown100.

with prior work. Stylistic disagreements between valid linguistic theories complicate evaluation of unsupervised grammar inducers to this day, despite several recent efforts to neutralize the effects of differences in annotation [282, 322].[4] Since theory-neutral evaluation of unsupervised dependency parsers is not yet a solved problem [113, §6.2], the primary metric used in this dissertation is simple unlabeled directed dependency accuracies (DDA).

[4]As an additional alternative to intrinsic, supervised parse quality metrics, unsupervised systems could also be evaluated extrinsically, by using features of their induced parse structures in down-stream tasks [81]. Unfortunately, task-based evaluation would make it difficult to compare to previous work: even concurrent evaluation of grammar induction systems, for machine translation, has proved impractical [113, Footnote 16].

2.3 English Data

The DMV was both trained and tested on a customized subset (WSJ10) of Penn English Treebank's Wall Street Journal portion [200]. Its 49,208 annotated parse trees were pruned down to 7,422 sentences of at most ten terminals, spanning 35 unique POS tags, by stripping out all empty subtrees, punctuation and terminals (tagged # and $) not pronounced where they appear. Following standard practice, automatic "head-percolation" rules [70] were used to convert remaining trees into dependencies. The work presented in this thesis makes use of generalizations WSJk, for $k \in \{1, \dots, 45, 100\}$, as well as Section 23 of WSJ$^\infty$ (the entire WSJ) and the Brown100 data set (see Figure 2.5 for all data set sizes), which is similarly derived from the parsed portion of the Brown corpus [106].

2.4 Multilingual Data

In addition to English WSJ, most of the work in this dissertation is also evaluated against all 23 held-out test sets of the 2006/7 CoNLL data [42, 236], spanning 19 languages from several different language families (see Table 2.1 for the sizes of its *disjoint* training and evaluation data, which were furnished by the CoNLL conference organizers). As with Section 23 of WSJ, here too I test on *all* sentence lengths, with the small exception of Arabic '07, from which I discarded the longest sentence (145 tokens). When computing macro-averages of directed dependency accuracies for the multilingual data, I down-weigh the four languages that appear in both years (Arabic, Chinese Czech and Turkish) by 50%.

2.5 A Note on Initialization Strategies

The exact form of Klein and Manning's initializer appears in the next chapter (Ch. 3), but two salient facts are worth mentioning sooner. First, my preliminary attempts to replicate the DMV showed that it is extremely important to start from the highest scoring trees for each training input (i.e., a step of Viterbi EM), instead of a forest of all projective trees

CoNLL Year			*Training*		*Testing*	
& Language			*Sentences*	*Tokens*	*Sentences*	*Tokens*
(ar)	Arabic	2006	1,460	52,752	146	5,215
		’7	2,912	102,375	130	4,537
(eu)	Basque	’7	3,190	41,013	334	4,511
(bg)	Bulgarian	’6	12,823	162,985	398	5,032
(ca)	Catalan	’7	14,958	380,525	167	4,478
(zh)	Chinese	’6	56,957	337,162	867	5,012
		’7	56,957	337,175	690	5,161
(cs)	Czech	’6	72,703	1,063,413	365	5,000
		’7	25,364	368,624	286	4,029
(da)	Danish	’6	5,190	80,743	322	4,978
(nl)	Dutch	’6	13,349	172,958	386	4,989
(en)	English	’7	18,577	395,139	214	4,386
(de)	German	’6	39,216	605,337	357	4,886
(el)	Greek	’7	2,705	58,766	197	4,307
(hu)	Hungarian	’7	6,034	111,464	390	6,090
(it)	Italian	’7	3,110	60,653	249	4,360
(ja)	Japanese	’6	17,044	133,927	709	5,005
(pt)	Portuguese	’6	9,071	177,581	288	5,009
(sl)	Slovenian	’6	1,534	23,779	402	5,004
(es)	Spanish	’6	3,306	78,068	206	4,991
(sv)	Swedish	’6	11,042	163,301	389	4,873
(tr)	Turkish	’6	4,997	48,373	623	6,288
		’7	5,635	54,761	300	3,983

Table 2.1: Sizes of all 2006/7 CoNLL training and evaluation data sets.

weighed proportionally to their ad-hoc harmonic scores. Viterbi training steps are colloquially known to be a worthwhile tool in machine learning,[5] and will be used extensively in the final part of this dissertation (Ch. 12) to transfer information between differently-factored models, initializing retraining. Although the properties of Viterbi EM (Chs. 4–5) are starting to receive theoretical treatment [67, 4], it may be interesting to also zoom in on the effects of the initial Viterbi step. For instance, Cohen and Smith [67] showed the optimality of initializing Viterbi training from uniform distributions. Pilot experiments with the DMV indicate that starting from parse trees chosen uniformly at random not only works

[5]Personal communication with Jenny Finkel and Slav Petrov.

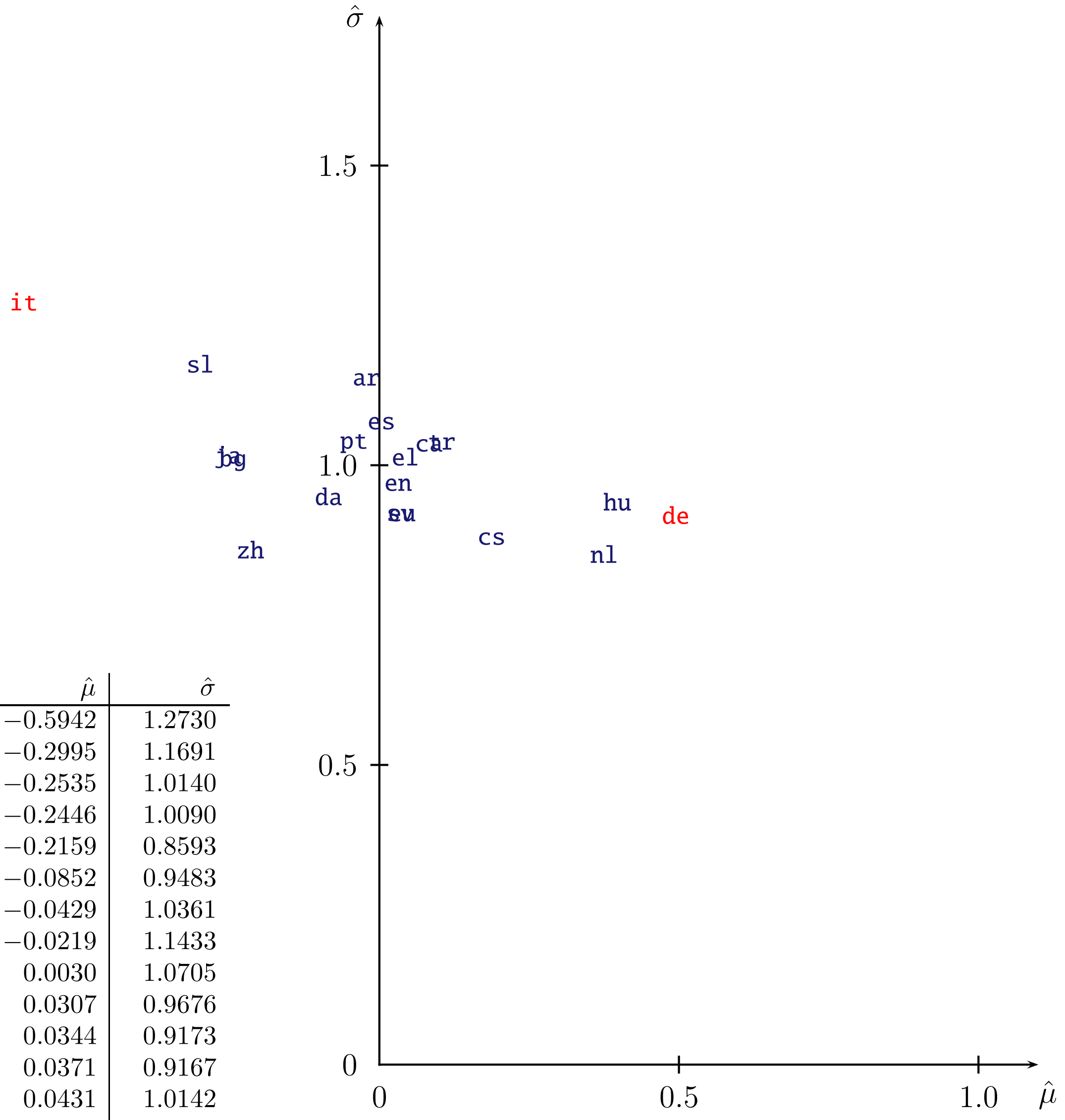
even better with soft EM but also outperforms the ad-hoc harmonic initializer, across many languages, and especially when a favorable maximum length cut-off has not been tuned.

Second, the harmonic part of Klein and Manning's initializer suggests a power law, with probabilities of attachment inversely proportional to distances between heads and dependents. However, a careful statistical analysis of arc lengths in the CoNLL data shows that if they indeed followed a power law, with $\mathbb{P}_{\texttt{ATTACH}}(d) \propto |d|^{-r}$, where d is the difference between connected words' positions in a sentence, then the power r would have to be at least two, and certainly not as low as one. It so happens that many empirical phenomena are easy to mistake for power laws,[6] particularly if the underlying distribution is log-normal [222]. The binned log-normal functional form

$$\mathbb{P}_{\texttt{ATTACH}}(d) \propto \left[\int_{\ln(|d|-1)}^{\ln|d|} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2} dt \right]^{-1}$$

is a better fit for the data (see Figure 2.6 for estimates $\hat{\mu}$ and $\hat{\sigma}$ of the CoNLL languages). But despite most languages clustering around the standard log-normal ($\mu = 0$ and $\sigma = 1$), using this fact to bias selection of initial parse trees, in pilot experiments, also proved worse than starting from a uniform distribution, as with harmonic initializers. Better uses of universal properties, such as functional forms, might incorporate them into parsing models and learn their parameters from data. The bulk of this dissertation thus focuses on the training aspects of grammar induction, aiming to eliminate dependence on tuned initializers.

[6]`http://vserver1.cscs.lsa.umich.edu/~crshalizi/weblog/491.html`



	$\hat{\mu}$	$\hat{\sigma}$
it	-0.5942	1.2730
sl	-0.2995	1.1691
ja	-0.2535	1.0140
bg	-0.2446	1.0090
zh	-0.2159	0.8593
da	-0.0852	0.9483
pt	-0.0429	1.0361
ar	-0.0219	1.1433
es	0.0030	1.0705
en	0.0307	0.9676
sv	0.0344	0.9173
eu	0.0371	0.9167
el	0.0431	1.0142
ca	0.0827	1.0335
tr	0.1039	1.0381
cs	0.1871	0.8779
nl	0.3752	0.8518
hu	0.3972	0.9394
de	0.4944	0.9170

Figure 2.6: Estimates of the binned log-normals' parameters, $\{\hat{\mu}\}$ and $\{\hat{\sigma}\}$, for arc lengths of CoNLL languages cluster around the standard log-normal's $\mu = 0$ and $\sigma = 1$. Outliers (in red) are Italian (`it`) and German (`de`), with very short and very long arcs, respectively.

Part I

Optimization

... the real challenge is to make simple things look beautiful.
— Glenn Corteza

Glenn and Aviv, from *The Tango in Ink Gallery*, by Jordana del Feld.

Chapter 3

Baby Steps

The purpose of this chapter is to get an understanding of how an established unsupervised dependency parsing model responds to the limits on sentence lengths that are conventionally used to filter input data, as well as its sensitivity to different initialization strategies. Supporting peer-reviewed publication is *From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing* in NAACL 2010 [302].

3.1 Introduction

This chapter explores what can be achieved through judicious use of data and simple, scalable techniques. The first approach iterates over a series of training sets that gradually increase in size and complexity, forming an initialization-independent scaffolding for learning a grammar. It works with Klein and Manning's simple model (the DMV) and training algorithm (classic EM) but eliminates their crucial dependence on manually-tuned priors. The second technique is consistent with the intuition that learning is most successful within a band of the size-complexity spectrum. Both could be applied to more intricate models and advanced learning algorithms. They are combined in a third, efficient hybrid method.

3.2 Intuition

Focusing on simple examples helps guide unsupervised learning, as blindly added confusing data can easily mislead training. Unless it is increased gradually, unbridled, complexity can overwhelm a system. How to grade an example's difficulty? The cardinality of its solution space presents a natural proxy. In the case of parsing, the number of possible syntactic trees grows exponentially with sentence length. For longer sentences, the unsupervised optimization problem becomes severely under-constrained, whereas for shorter sentences, learning is tightly reined in by data. In the extreme case of a single-word sentence, there is no choice but to parse it correctly. At two words, a raw 50% chance of telling the head from its dependent is still high, but as length increases, the accuracy of even educated guessing rapidly plummets. In model re-estimation, long sentences amplify ambiguity and pollute fractional counts with noise. At times, batch systems are better off using less data.

Baby Steps: Global non-convex optimization is hard. But a meta-heuristic can take the guesswork out of initializing local search. Beginning with an easy (convex) case, it is possibly to slowly extend it to the fully complex target task by taking tiny steps in the problem space, trying not to stray far from the relevant neighborhoods of the solution space. A series of nested subsets of increasingly longer sentences that culminates in the complete data set offers a natural progression. Its base case — sentences of length one — has a trivial solution that requires neither initialization nor search yet reveals something of sentence heads. The next step — sentences of length one and two — refines initial impressions of heads, introduces dependents, and exposes their identities and relative positions. Although not representative of the full grammar, short sentences capture enough information to paint most of the picture needed by slightly longer sentences. They set up an easier, incremental subsequent learning task. Step $k + 1$ augments training input to include lengths $1, 2, \ldots, k, k + 1$ of the full data set and executes local search starting from the (appropriately smoothed) model estimated by step k. This truly is grammar induction...

Less is More: For standard batch training, just using simple, short sentences is not enough. They are rare and do not reveal the full grammar. Instead, it is possible to find a "sweet spot" — sentence lengths that are neither too long (excluding the truly daunting examples) nor too few (supplying enough accessible information), using Baby Steps' learning curve as a guide. It makes sense to train where that learning curve flattens out, since remaining sentences contribute little (incremental) educational value.[1]

Leapfrog: An alternative to discarding data, and a better use of resources, is to combine the results of batch and iterative training up to the sweet spot data gradation, then iterate with a large step size.

3.3 Related Work

Two types of scaffolding for guiding language learning debuted in Elman's [95] experiments with "starting small": data complexity (restricting input) and model complexity (restricting memory). In both cases, gradually increasing complexity allowed artificial neural networks to master a pseudo-natural grammar that they otherwise failed to learn. Initially-limited capacity resembled maturational changes in working memory and attention span that occur over time in children [163], in line with the "less is more" proposal [230, 231]. Although Rohde and Plaut [267] failed to replicate this[2] result with simple recurrent networks, many machine learning techniques, for a variety of language tasks, reliably benefit from annealed model complexity. Brown et al. [40] used IBM Models 1–4 as "stepping stones" to training word-alignment Model 5. Other prominent examples include "coarse-to-fine" approaches to parsing, translation, speech recognition and unsupervised POS tagging [53, 54, 250, 251, 261]. Initial models tend to be particularly simple,[3] and each refinement towards a full model introduces only limited complexity, supporting incrementality.

Filtering complex data, the focus of this chapter, is unconventional in natural language processing. Such scaffolding qualifies as *shaping* — a method of instruction (routinely

[1]This is akin to McClosky et al.'s [208] "Goldilocks effect."

[2]Worse, they found that limiting input *hindered* language acquisition. And making the grammar more English-like (by introducing and strengthening semantic constraints), *increased* the already significant advantage for "starting large!" With iterative training invoking the optimizer multiple times, creating extra opportunities to converge, Rohde and Plaut suspected that Elman's simulations simply did not allow networks exposed exclusively to complex inputs sufficient training time. Extremely generous, low termination threshold for EM (see §3.4.1) address this concern, and the DMV's purely syntactic POS tag-based approach (see §2.1) is, in a later chapter (see §12.5.2), replaced with Baby Steps iterating over fully-lexicalized models.

[3]Brown et al.'s [40] Model 1 (and, similarly, the first baby step) has a global optimum that can be computed exactly, so that no initial or subsequent parameters depend on initialization.

exploited in animal training) in which the teacher decomposes a complete task into sub-components, providing an easier path to learning. When Skinner [289] coined the term, he described it as a "method of successive approximations." Ideas that gradually make a task more difficult have been explored in robotics (typically, for navigation), with reinforcement learning [288, 275, 271, 86, 279, 280]. More recently, Krueger and Dayan [175] showed that shaping speeds up language acquisition and leads to better generalization in abstract neural networks. Bengio et al. [22] confirmed this for deep deterministic and stochastic networks, using simple multi-stage *curriculum* strategies. They conjectured that a well-chosen sequence of training criteria — different sets of weights on the examples — could act as a continuation method [5], helping find better local optima for non-convex objectives. Elman's learners constrained the peaky solution space by focusing on just the right data (simple sentences that introduced basic representational categories) at just the right time (early on, when their plasticity was greatest). Self-shaping, they simplified tasks through deliberate omission (or misunderstanding). Analogously, Baby Steps induces an early structural locality bias [295], then relaxes it, as if annealing [292]. Its curriculum of binary weights initially discards complex examples responsible for "high-frequency noise," with earlier, "smoothed" objectives revealing more of the global picture.

There are important differences between the work in this chapter and prior research. In contrast to Elman, it relies on a large data set (WSJ) of real English. Unlike Bengio et al. and Krueger and Dayan, it shapes a parser, not a language model. Baby Steps is similar, in spirit, to Smith and Eisner's methods. Deterministic annealing (DA) shares nice properties with Baby Steps, but performs worse than EM for (constituent) parsing; Baby Steps handedly defeats standard training. Structural annealing works well, but requires a hand-tuned annealing schedule and direct manipulation of the objective function; Baby Steps works "out of the box," its locality biases a natural consequence of a complexity/data-guided tour of optimization problems. Skewed DA incorporates a good initializer by interpolating between two probability distributions, whereas the Leapfrog hybrid admits multiple initializers by mixing structures instead. "Less is More" is novel and confirms the tacit consensus implicit in training on small data sets (e.g., WSJ10).

3.4 New Algorithms for the Classic Model

Many seemingly small implementation details can have profound effects on the final output of a training procedure tasked with optimizing a non-convex objective. Contributing to the chaos are handling of ties (e.g., in decoding), the choice of random number generator and seed (e.g., if tie-breaking is randomized), whether probabilities are represented in log-space, their numerical precision, and also the order in which these floating point numbers are added or multiplied, to say nothing of initialization (and termination) conditions. For these reasons, even the correct choices of tuned parameters in the next section might not result in a training run that would match Klein and Manning's actual execution of the DMV.

3.4.1 Algorithm #0: Ad-Hoc* — A Variation on Original Ad-Hoc Initialization

Below are the ad-hoc harmonic scores (for all tokens other than $\diamondsuit$):

$$\tilde{\mathbb{P}}_{\texttt{ORDER}} \equiv 1/2;$$

$$\tilde{\mathbb{P}}_{\texttt{STOP}} \equiv (d_s + \delta_s)^{-1} = (d_s + 3)^{-1}, \;\; d_s \geq 0;$$

$$\tilde{\mathbb{P}}_{\texttt{ATTACH}} \equiv (d_a + \delta_a)^{-1} = (d_a + 2)^{-1}, \;\; d_a \geq 1.$$

Integers $d_{\{s,a\}}$ are distances from heads to stopping boundaries and dependents.[4] Training is initialized by producing best-scoring parses of all input sentences and converting them into proper probability distributions $\mathbb{P}_{\texttt{STOP}}$ and $\mathbb{P}_{\texttt{ATTACH}}$ via maximum-likelihood estimation (a single step of Viterbi training [40]). Since left and right children are independent, $\mathbb{P}_{\texttt{ORDER}}$ is dropped altogether, making "headedness" deterministic. The parser carefully randomizes tie-breaking, so that all structures having the same score get an equal shot at being selected (both during initialization and evaluation). EM is terminated when a successive change in overall per-token cross-entropy drops below 2^{-20} bits.

[4]Constants $\delta_{\{s,a\}}$ come from personal communication. Note that δ_s is one higher than is strictly necessary to avoid both division by zero and determinism; δ_a could have been safely zeroed out, since the quantity $1 - \mathbb{P}_{\texttt{ATTACH}}$ is never computed (see Figure 2.3).

3.4.2 Algorithm #1: Baby Steps — An Initialization-Independent Scaffolding

The need for initialization is eliminated by first training on a trivial subset of the data — WSJ1; this works, since there is only one (the correct) way to parse a single-token sentence. A resulting model is plugged into training on WSJ2 (sentences up to two tokens), and so forth, building up to WSJ45.[5] This algorithm is otherwise identical to Ad-Hoc*, with the exception that it re-estimates each model using Laplace smoothing, so that earlier solutions could be passed to next levels, which sometimes contain previously unseen POS tags.

3.4.3 Algorithm #2: Less is More — Ad-Hoc* where Baby Steps Flatlines

Long, complex sentences are dropped, deploying Ad-Hoc*'s initializer for batch training at WSJ$\hat{k}^*$, an estimate of the sweet spot data gradation. To find it, Baby Steps' successive models' cross-entropies on the complete data set, WSJ45, are tracked. An initial segment of rapid improvement is separated from the final region of convergence by a *knee* (points of maximum curvature, see Figure 3.1). An improved[6] L method [272] automatically locates this area of diminishing returns: the end-points $[k_0, k^*]$ are determined by minimizing squared error, estimating $\hat{k}_0 = 7$ and $\hat{k}^* = 15$. Training at WSJ15 just misses the plateau.

3.4.4 Algorithm #3: Leapfrog — A Practical and Efficient Hybrid Mixture

Cherry-picking the best features of "Less is More" and Baby Steps, the hybrid begins by combining their models at WSJ$\hat{k}^*$. Using one best parse from each, for every sentence in

[5]Its 48,418 sentences (see Figure 3.1) cover 94.4% of all sentences in WSJ; the longest of the missing 790 has length 171.

[6]Instead of iteratively fitting a two-segment form and adaptively discarding its tail, we use *three* line segments, applying ordinary least squares to the first two, but requiring the third to be horizontal and tangent to a minimum. The result is a *batch* optimization routine that returns an *interval* for the knee, rather than a point estimate (see Figure 3.1 for details).

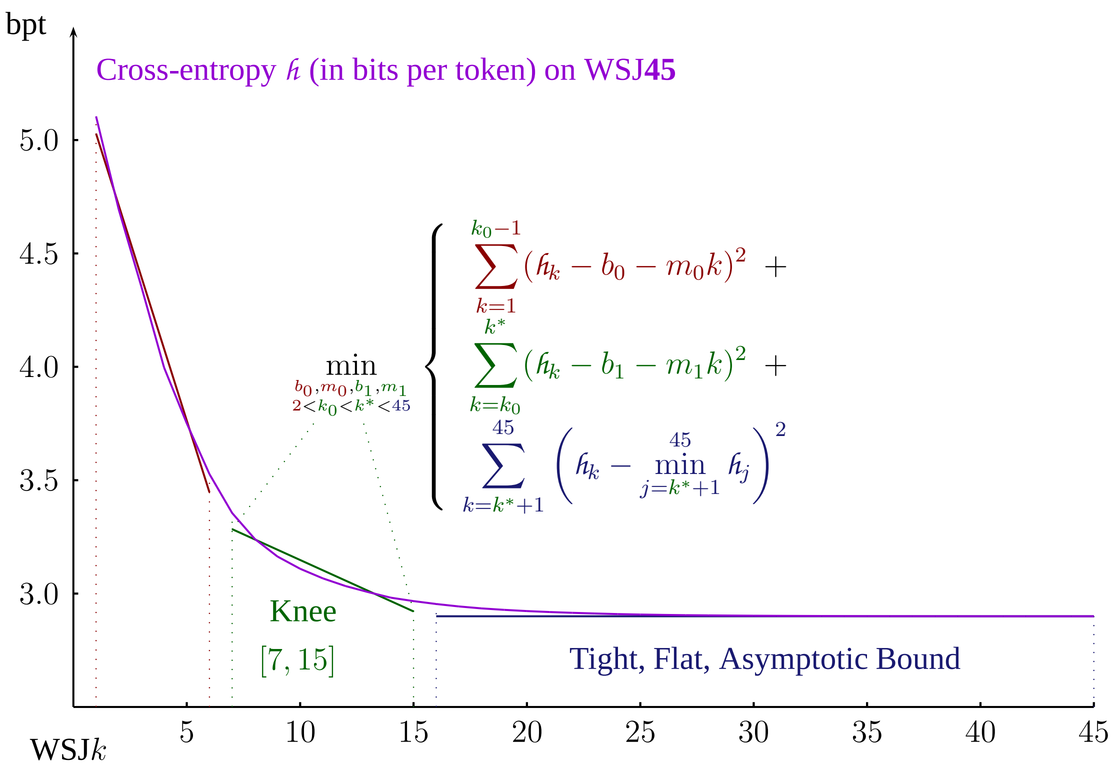


Figure 3.1: Cross-entropy on WSJ45 after each baby step, a piece-wise linear fit, and an estimated region for the knee.

WSJ$\hat{k}^*$, the base case re-estimates a new model from a *mixture* of twice the normal number of trees; inductive steps leap over $\hat{k}^*$ lengths, conveniently ending at WSJ45, and estimate their initial models by applying a previous solution to a new input set. Both follow up the single step of Viterbi training with at most five iterations of EM.

This hybrid makes use of two good (conditionally) independent initialization strategies and executes many iterations of EM where that is cheap — at shorter sentences (WSJ15 and below). It then increases the step size, training just three more times (at WSJ$\{15, 30, 45\}$) and allowing only a few (more expensive) iterations of EM there. Early termination improves efficiency and regularizes these final models.

3.4.5 Reference Algorithms — Baselines, a Skyline and Published Art

The working performance space can be carved out using two extreme initialization strategies: (i) the uninformed uniform prior, which serves as a fair "zero-knowledge" baseline for comparing uninitialized models; and (ii) the maximum-likelihood "oracle" prior, computed from reference parses, which yields a *skyline* (a reverse baseline) — how well any algorithm that stumbled on the true solution would fare at EM's convergence.

Accuracies on Section 23 of WSJ$^\infty$ are compared to two state-of-the-art systems and past baselines (see Table 3.2), in addition to Klein and Manning's results. Headden et al.'s [133] lexicalized EVG had the best previous results on short sentences, but its performance is unreported for longer sentences, for which Cohen and Smith's [66] seem to be the highest published scores; intermediate results that preceded parameter-tying — Bayesian models with Dirichlet and log-normal priors, coupled with both Viterbi and minimum Bayes-risk (MBR) decoding [65] — are also included.

3.5 Experimental Results

Thousands of empirical outcomes are packed into the space of several graphs (Figures 3.2, 3.3 and 3.4). The colors (also in Tables 3.1 and 3.2) correspond to different initialization strategies — to a first approximation, the learning algorithm was held constant (see §2.1).

Figures 3.2 and 3.3 tell one part of our story. As data sets increase in size, training algorithms gain access to more information; however, since in this unsupervised setting training and test sets are the same, additional longer sentences make for substantially more challenging evaluation. To control for these dynamics, it is possible to apply Laplace smoothing to all (otherwise unsmoothed) models and replot their performance, holding several test sets fixed (see Figure 3.4). (Undirected accuracies are reported parenthetically.)

3.5.1 Result #1: Baby Steps

Figure 3.2 traces out performance on the training set. Klein and Manning's published scores appear as dots (Ad-Hoc) at WSJ10: 43.2% (63.7%). Baby Steps achieves 53.0% (65.7%)

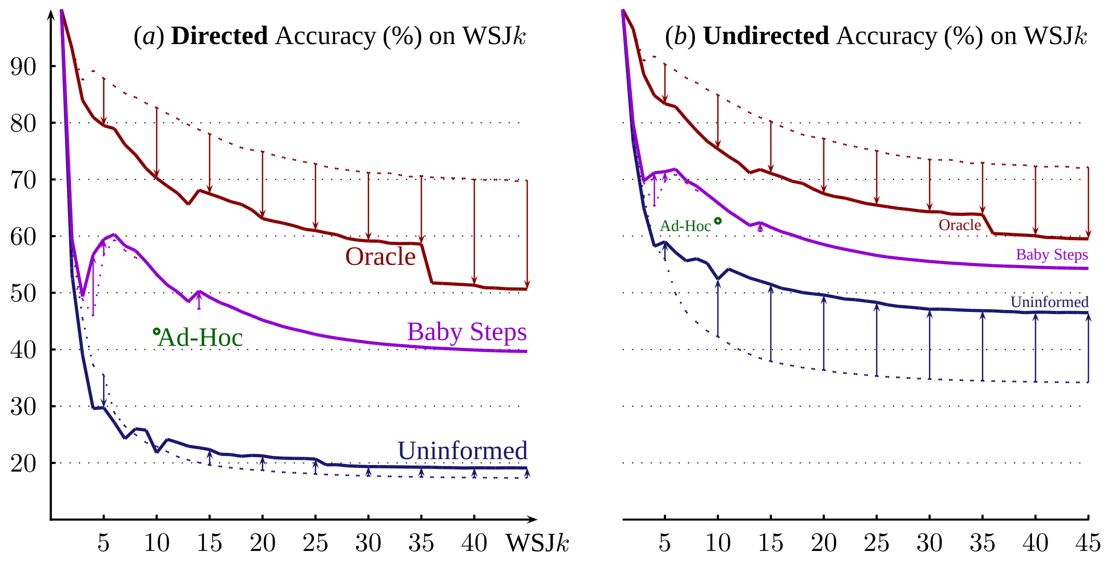


Figure 3.2: Directed and undirected accuracy scores attained by the DMV, when trained and tested on the same gradation of WSJ, for several different initialization strategies. Green circles mark Klein and Manning's published scores; red, violet and blue curves represent the supervised (maximum-likelihood oracle) initialization, Baby Steps, and the uninformed uniform prior. Dotted curves reflect starting performance, solid curves register performance at EM's convergence, and the arrows connecting them emphasize the impact of learning.

by WSJ10; trained and tested on WSJ45, it gets 39.7% (54.3%). Uninformed, classic EM learns little about directed dependencies: it improves only slightly, e.g., from 17.3% (34.2%) to 19.1% (46.5%) on WSJ45 (learning some of the structure, as evidenced by its undirected scores), but degrades with shorter sentences, where its initial guessing rate is high. In the case of oracle training, EM is expected to walk away from supervised solutions [97, 219, 189], but the extent of its drops is alarming, e.g., from the supervised 69.8% (72.2%) to the skyline's 50.6% (59.5%) on WSJ45. By contrast, Baby Steps' scores usually do not change much from one step to the next, and where its impact of learning is big (at WSJ$\{4, 5, 14\}$), it is invariably positive.

3.5.2 Result #2: Less is More

Ad-Hoc*'s curve (see Figure 3.3) suggests how Klein and Manning's Ad-Hoc initializer may have scaled with different gradations of WSJ. Strangely, the implementation in this

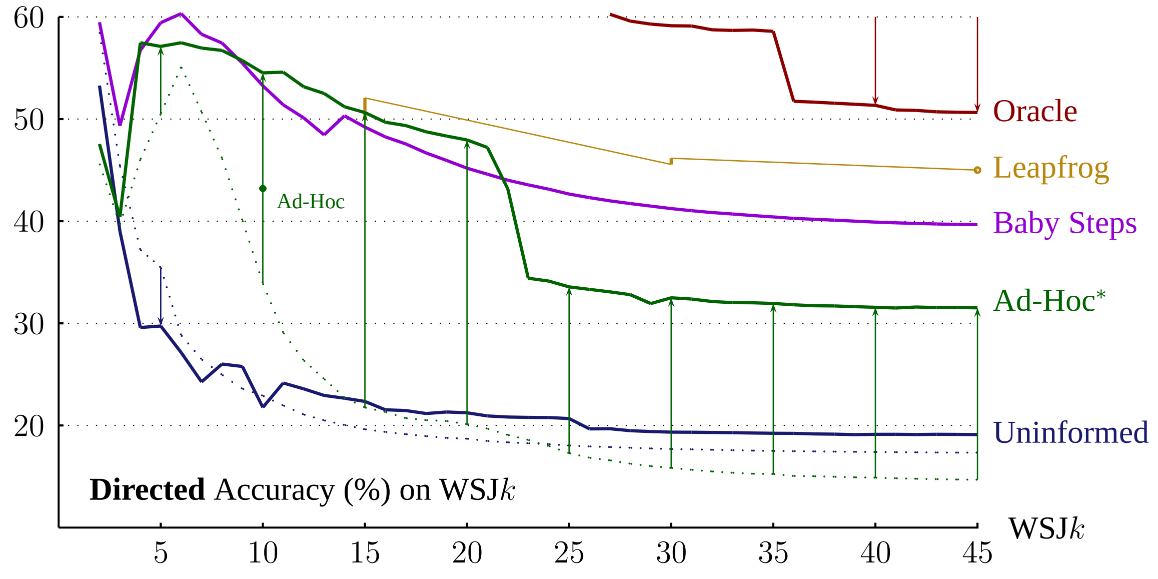


Figure 3.3: Directed accuracies for Ad-Hoc* (shown in green) and Leapfrog (in gold); all else as in Figure 3.2(*a*).

chapter performs significantly above their reported numbers at WSJ10: 54.5% (68.3%) is even slightly higher than Baby Steps; nevertheless, given enough data (from WSJ22 onwards), Baby Steps overtakes Ad-Hoc*, whose ability to learn takes a serious dive once the inputs become sufficiently complex (at WSJ23), and never recovers. Note that Ad-Hoc*'s biased prior peaks early (at WSJ6), eventually falls below the guessing rate (by WSJ24), yet still remains well-positioned to climb, outperforming uninformed learning.

Figure 3.4 shows that Baby Steps scales better with more (complex) data — its curves do not trend downwards. However, a good initializer induces a sweet spot at WSJ15, where the DMV is learned best using Ad-Hoc*. This mode *is* "Less is More," scoring 44.1% (58.9%) on WSJ45. Curiously, even oracle training exhibits a bump at WSJ15: once sentences get long enough (at WSJ36), its performance degrades below that of oracle training with virtually no supervision (at the hardly representative WSJ3).

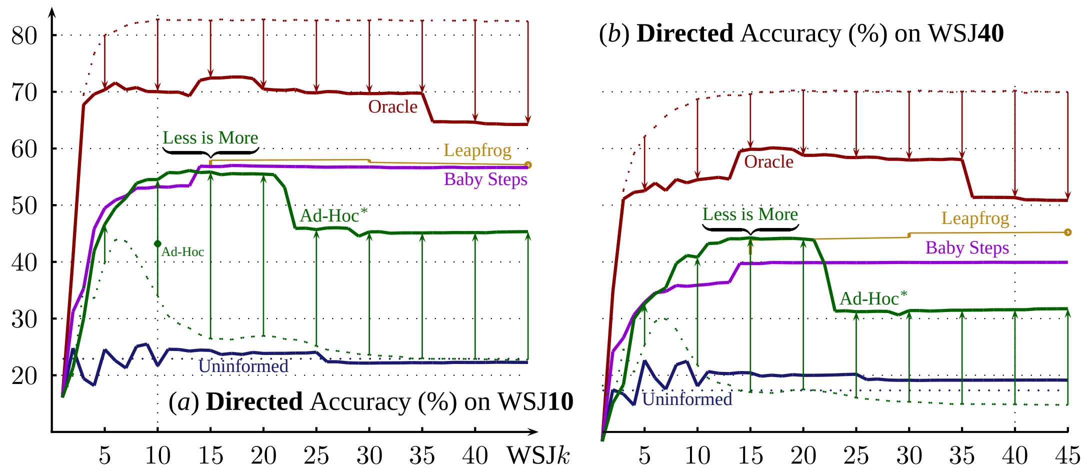


Figure 3.4: Directed accuracies attained by the DMV, when trained at various gradations of WSJ, smoothed, then tested against fixed evaluation sets — WSJ$\{10, 40\}$; graphs for WSJ$\{20, 30\}$, not shown, are qualitatively similar to WSJ40.

3.5.3 Result #3: Leapfrog

Mixing Ad-Hoc* with Baby Steps at WSJ15 yields a model whose performance initially falls between its two parents but surpasses both with a little training (see Figure 3.3). Leaping to WSJ45, via WSJ30, results in the strongest model: 45.0% (58.4%) accuracy bridges half of the gap between Baby Steps and the skyline, and at a tiny fraction of the cost.

	*Ad-Hoc**	*Baby Steps*	*Leapfrog*	*Ad-Hoc**	*Baby Steps*	*Leapfrog*
Section 23	44.1 (58.8)	39.2 (53.8)	43.3 (55.7)	31.5 (51.6)	39.4 (54.0)	**45.0** (58.4)
WSJ100	43.8 (58.6)	39.2 (53.8)	43.3 (55.6)	31.3 (51.5)	39.4 (54.1)	**44.7** (58.1)
Brown100	43.3 (59.2)	42.3 (55.1)	42.8 (56.5)	32.0 (52.4)	42.5 (55.5)	**43.6** (59.1)
	@15			@45		

Table 3.1: Directed (and undirected) accuracies on Section 23 of WSJ$^\infty$, WSJ100 and Brown100 for Ad-Hoc*, Baby Steps and Leapfrog, trained at WSJ15 (left) and WSJ45.

3.5.4 Result #4: Generalization

These models carry over to the larger WSJ100, Section 23 of WSJ$^\infty$, and the independent Brown100 (see Table 3.1). Baby Steps improves out of domain, confirming that shaping

			Decoding	WSJ10	WSJ20	WSJ∞
	Attach-Right	[172]	—	38.4	33.4	31.7
DMV	Ad-Hoc	[172]	Viterbi	45.8	39.1	34.2
	Dirichlet	[65]	Viterbi	45.9	39.4	34.9
	Ad-Hoc	[65]	MBR	46.1	39.9	35.9
	Dirichlet	[65]	MBR	46.1	40.6	36.9
	Log-Normal Families	[65]	Viterbi	59.3	45.1	39.0
	Baby Steps (@15)		*Viterbi*	*55.5*	*44.3*	*39.2*
	Baby Steps (@45)		*Viterbi*	*55.1*	*44.4*	*39.4*
	Log-Normal Families	[65]	MBR	59.4	45.9	40.5
	Shared Log-Normals (tie-verb-noun)	[66]	MBR	61.3	47.4	41.4
	Bilingual Log-Normals (tie-verb-noun)	[66]	MBR	62.0	48.0	42.2
	Less is More (Ad-Hoc* @15)		*Viterbi*	*56.2*	*48.2*	*44.1*
	Leapfrog (Hybrid @45)		*Viterbi*	*57.1*	**48.7**	**45.0**
EVG	Smoothed (skip-val)	[133]	Viterbi	62.1		
	Smoothed (skip-head)	[133]	Viterbi	65.0		
	Smoothed (skip-head), Lexicalized	[133]	Viterbi	**68.8**		

Table 3.2: Directed accuracies on Section 23 of WSJ$\{10, 20, ^\infty\}$ for several baselines and previous state-of-the-art systems.

generalizes well [175, 22]. Leapfrog does best across the board but dips on Brown100, despite its safe-guards against overfitting.

Section 23 (see Table 3.2) reveals, unexpectedly, that Baby Steps would have been state-of-the-art in 2008, whereas "Less is More" outperforms all prior work on longer sentences. Baby Steps is competitive with log-normal families [65], scoring slightly better on longer sentences against Viterbi decoding, though worse against MBR. "Less is More" beats state-of-the-art on longer sentences by close to 2%; Leapfrog gains another 1%.

3.6 Conclusion

This chapter explored three simple ideas for unsupervised dependency parsing. Pace Halevy et al. [130], it suggests, "Less is More" — the paradoxical result that better performance can be attained by training with less data, even when removing samples from the true (test) distribution. Small tweaks to Klein and Manning's approach of 2004 break through the 2009 state-of-the-art on longer sentences, when trained at WSJ15 (the auto-detected sweet spot gradation). Second, Baby Steps, is an elegant meta-heuristic for optimizing non-convex training criteria. It eliminates the need for linguistically-biased manually-tuned initializers, particularly if the location of the sweet spot is not known. This technique scales gracefully with more (complex) data and should easily carry over to more powerful parsing models and learning algorithms. Finally, Leapfrog forgoes the elegance and meticulousness of Baby Steps in favor of pragmatism. Employing both good initialization strategies at its disposal, and spending CPU cycles wisely, it achieves better performance than both "Less is More" and Baby Steps.

Later chapters will explore unifying these techniques with other state-of-the-art approaches, which will scaffold on both data and model complexity. There are many opportunities for improvement, considering the poor performance of oracle training relative to the supervised state-of-the-art, and in turn the poor performance of unsupervised state-of-the-art relative to the oracle models.

Chapter 4

Viterbi Training

The purpose of this chapter is to explore, compare and contrast the implications of using Viterbi training (hard EM) versus traditional inside-outside re-estimation (soft EM) for grammar induction with the DMV, as well as to clarify that the unsupervised objectives used by both algorithms can be "wrong," from perspectives of would-be supervised objectives. Supporting peer-reviewed publication is *Viterbi Training Improves Unsupervised Dependency Parsing* in CoNLL 2010 [309].

4.1 Introduction

Unsupervised learning is hard, often involving difficult objective functions. A typical approach is to attempt maximizing the likelihood of unlabeled data, in accordance with a probabilistic model. Sadly, such functions are riddled with local optima [49, Ch. 7, *inter alia*], since their number of peaks grows exponentially with instances of hidden variables. Furthermore, higher likelihood does not always translate into superior task-specific accuracy [97, 219]. Both complications are real, but this chapter will discuss perhaps more significant shortcomings.

This chapter proves that learning can be error-prone even in cases when likelihood *is* an appropriate measure of extrinsic performance *and* where global optimization is feasible. This is because a key challenge in unsupervised learning is that the *desired* likelihood is unknown. Its absence renders tasks like structure discovery inherently under-constrained.

Search-based algorithms adopt surrogate metrics, gambling on convergence to the "right" regularities in data. Wrong objectives create opportunities to improve *both* efficiency *and* performance by replacing expensive exact learning techniques with cheap approximations.

This chapter proposes using Viterbi training [40, §6.2], instead of the more standard inside-outside re-estimation [14], to induce hierarchical syntactic structure from natural language text. Since the objective functions being used in unsupervised grammar induction are provably wrong, advantages of exact inference may not apply. It makes sense to try the Viterbi approximation — it is also wrong, only simpler and cheaper than classic EM. As it turns out, Viterbi EM is not only faster but also more accurate, consistent with hypotheses of de Marcken [82] and with the suggestions from the previous chapter. After reporting the experimental results and relating its contributions to prior work, this chapter delves into proofs by construction, using the DMV.

4.2 Viterbi Training and Evaluation with the DMV

Viterbi training [40] re-estimates each next model as if supervised by the previous best parse trees. And supervised learning from reference parse trees is straight-forward, since maximum-likelihood estimation reduces to counting: $\hat{\mathbb{P}}_{\texttt{ATTACH}}(c_h, dir, c_d)$ is the fraction of dependents — those of class c_d — attached on the dir side of a head of class c_h; $\hat{\mathbb{P}}_{\texttt{STOP}}(c_h, dir, adj = \texttt{T})$, the fraction of words of class c_h with no children on the dir side; and $\hat{\mathbb{P}}_{\texttt{STOP}}(c_h, dir, adj = \texttt{F})$, the ratio[1] of the number of words of class c_h having a child on the dir side to their total number of such children.

Proposed parse trees are judged on accuracy: a *directed score* is simply the overall fraction of correctly guessed dependencies. Let S be a set of sentences, with $|s|$ the number of terminals (tokens) for each $s \in S$. Denote by $T(s)$ the set of all dependency parse trees of s, and let $t_i(s)$ stand for the parent of token i, $1 \leq i \leq |s|$, in $t(s) \in T(s)$. Call the gold reference $t^*(s) \in T(s)$. For a given model of grammar, parameterized by θ, let

[1]The expected number of trials needed to get one Bernoulli(p) success is $n \sim$ Geometric(p), with $n \in \mathbb{Z}^+$, $\mathbb{P}(n) = (1-p)^{n-1}p$ and $\mathbb{E}(n) = p^{-1}$; MoM and MLE agree, in this case, $\hat{p} = (\#\text{ of successes})/(\#\text{ of trials})$.

$\hat{t}^{\theta}(s) \in T(s)$ be a (not necessarily unique) likeliest (also known as Viterbi) parse of s:

$$\hat{t}^{\theta}(s) \in \left\{ \arg\max_{t \in T(s)} \mathbb{P}_{\theta}(t) \right\};$$

then θ's directed accuracy on a reference set R is

$$100\% \cdot \frac{\sum_{s \in R} \sum_{i=1}^{|s|} 1_{\{\hat{t}_i^{\theta}(s) = t_i^*(s)\}}}{\sum_{s \in R} |s|}.$$

4.3 Experimental Setup and Results

As in the previous chapter, the DMV was trained on data sets WSJ$\{1, \ldots, 45\}$ using three initialization strategies: (i) the uninformed uniform prior; (ii) a linguistically-biased initializer, Ad-Hoc*; and (iii) an oracle — the supervised MLE solution. Previously, training was without smoothing, iterating each run until successive changes in overall per-token cross-entropy drop below 2^{-20} bits. In this chapter all models are re-trained using Viterbi EM instead of inside-outside re-estimation, and also explore Laplace (add-one) smoothing during training and experiment with hybrid initialization strategies.

4.3.1 Result #1: Viterbi-Trained Models

The results of the previous chapter, tested against WSJ40, are re-printed in Figure 4.1(a); and the corresponding Viterbi runs appear in Figure 4.1(b). There are crucial differences between the two training modes for each of the three initialization strategies. Both algorithms walk away from the supervised maximum-likelihood solution; however, Viterbi EM loses at most a few points of accuracy (3.7% at WSJ40), whereas classic EM drops nearly twenty points (19.1% at WSJ45). In both cases, the single best unsupervised result is with good initialization, although Viterbi peaks earlier (45.9% at WSJ8) — and in a narrower range (WSJ8-9) — than classic EM (44.3% at WSJ15; WSJ13-20). The uniform prior

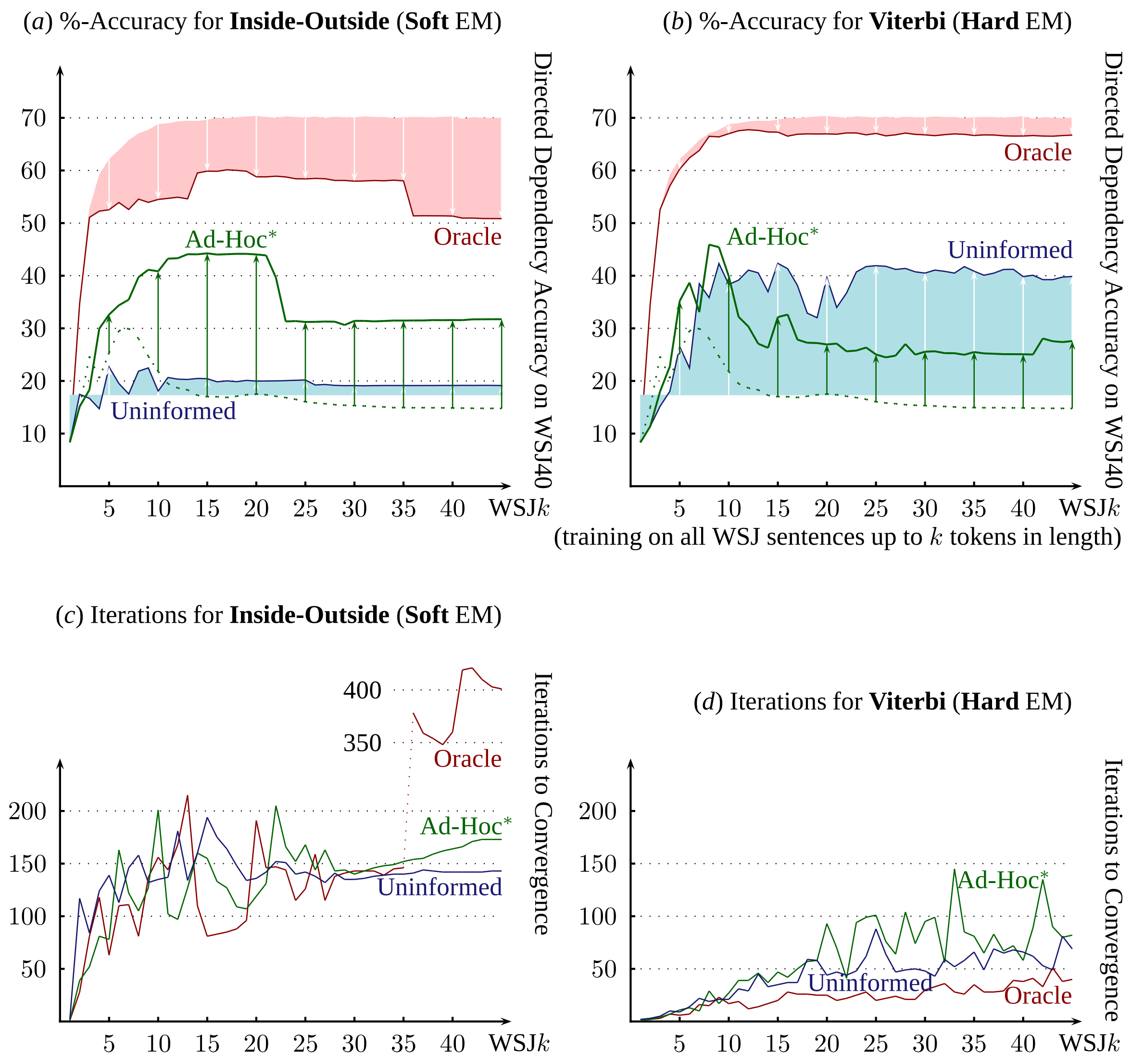


Figure 4.1: Directed dependency accuracies attained by the DMV, when trained on WSJk, smoothed, then tested against a fixed evaluation set, WSJ40, for three different initialization strategies. Red, green and blue graphs represent the supervised (maximum-likelihood oracle) initialization, a linguistically-biased initializer (Ad-Hoc*) and the uninformed (uniform) prior. Panel (b) shows results obtained with Viterbi training instead of classic EM — Panel (a), but is otherwise identical (in both, each of the 45 vertical slices captures five new experimental results and arrows connect starting performance with final accuracy, emphasizing the impact of learning). Panels (c) and (d) show the corresponding numbers of iterations until EM's convergence.

never quite gets off the ground with classic EM but manages quite well under Viterbi training,[2] given sufficient data — it even beats the "clever" initializer everywhere past WSJ10. The "sweet spot" at WSJ15 — a neighborhood where both Ad-Hoc* and the oracle excel under classic EM — disappears with Viterbi. Furthermore, Viterbi does not degrade with more (complex) data, except with a biased initializer.

More than a simple efficiency hack, Viterbi EM actually improves performance. And its benefits to running times are also non-trivial: it not only skips computing the outside charts in every iteration but also converges (sometimes an order of magnitude) faster than classic EM (see Figure 4.1(c,d)).[3]

4.3.2 Result #2: Smoothed Models

Smoothing rarely helps classic EM and hurts in the case of oracle training (see Figure 4.2(a)). With Viterbi, supervised initialization suffers much less, the biased initializer is a wash, and the uninformed uniform prior generally gains a few points of accuracy, e.g., up 2.9% (from 42.4% to 45.2%, evaluated against WSJ40) at WSJ15 (see Figure 4.2(b)).

Baby Steps (Ch. 3) — iterative re-training with increasingly more complex data sets, WSJ$1, \dots,$WSJ45 — using smoothed Viterbi training fails miserably (see Figure 4.2(b)), due to Viterbi's poor initial performance at short sentences (possibly because of data sparsity and sensitivity to non-sentences — see examples in §4.6.3).

4.3.3 Result #3: State-of-the-Art Models

Simply training up smoothed Viterbi at WSJ15, using the uninformed uniform prior, yields 44.8% accuracy on Section 23 of WSJ$^\infty$, which already surpasses the previous state-of-the-art by 0.7% (see Table 4.1(A)). Since both classic EM and Ad-Hoc* initializers work

[2]In a concurrently published related work, Cohen and Smith [67] prove that the uniform-at-random initializer is a competitive starting M-step for Viterbi EM; the uninformed prior from the last chapter consists of uniform multinomials, seeding the E-step, which also yields equally-likely parse trees for models like DMV.

[3]For classic EM, the number of iterations to convergence appears sometimes inversely related to performance, giving still more credence to the notion of early termination as a regularizer.

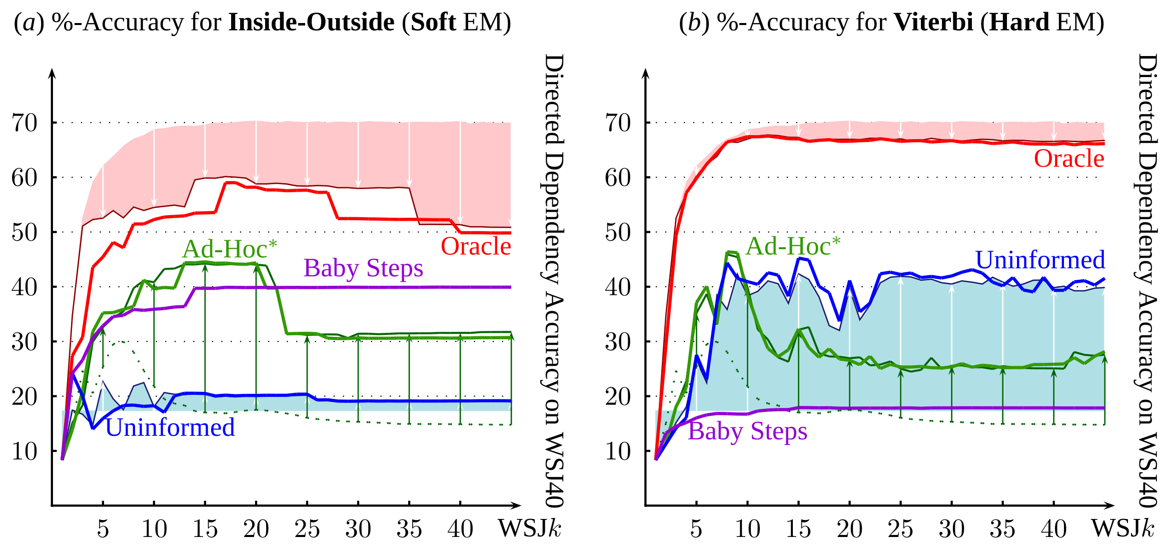


Figure 4.2: Directed accuracies for DMV models trained *with* Laplace smoothing (brightly-colored curves), superimposed over Figure 4.1(a,b); violet curves represent Baby Steps.

well with short sentences (see Figure 4.1(a)), it makes sense to use their pre-trained models to initialize Viterbi training, mixing the two strategies. Judging all Ad-Hoc* initializers against WSJ15, it turns out that the one for WSJ8 minimizes sentence-level cross-entropy (see Figure 4.3). This approach does not involve reference parse trees and is therefore still unsupervised. Using the Ad-Hoc* initializer based on WSJ8 to seed classic training at WSJ15 yields a further 1.4% gain in accuracy, scoring 46.2% on WSJ$^\infty$ (see Table 4.1(*B*)). This good initializer boosts accuracy attained by smoothed Viterbi at WSJ15 to 47.8% (see Table 4.1(*C*)). Using its solution to re-initialize training at WSJ45 gives a tiny further improvement (0.1%) on Section 23 of WSJ$^\infty$ but bigger gains on WSJ10 (0.9%) and WSJ20 (see Table 4.1(*D*)). These results generalize. Gains due to smoothed Viterbi training and favorable initialization carry over to Brown100 — accuracy improves by 7.5% over previous published numbers (see Table 4.1).

Model	*Incarnation*		WSJ10	WSJ20	WSJ$^\infty$	Brown100
DMV	Bilingual Log-Normals (tie-verb-noun)	[66]	62.0	48.0	42.2	
	Less is More (Ad-Hoc* @15)	(Ch. 3)	56.2	48.2	44.1	43.3
A.	Smoothed Viterbi Training (@15), Initialized with the Uniform Prior		59.9	50.0	44.8	48.1
B.	A Good Initializer (Ad-Hoc*'s @8), Classically Pre-Trained (@15)		63.8	52.3	46.2	49.3
C.	Smoothed Viterbi Training (@15), Initialized with *B*		64.4	53.5	47.8	50.5
D.	Smoothed Viterbi Training (@45), Initialized with *C*		65.3	**53.8**	**47.9**	**50.8**
EVG	Smoothed (skip-head), Lexicalized	[133]	**68.8**			

Table 4.1: Accuracies on Section 23 of WSJ$\{10, 20, ^\infty\}$ and Brown100 for three previous state-of-the-art systems, this chapter's initializer, and smoothed Viterbi-trained runs that employ different initialization strategies.

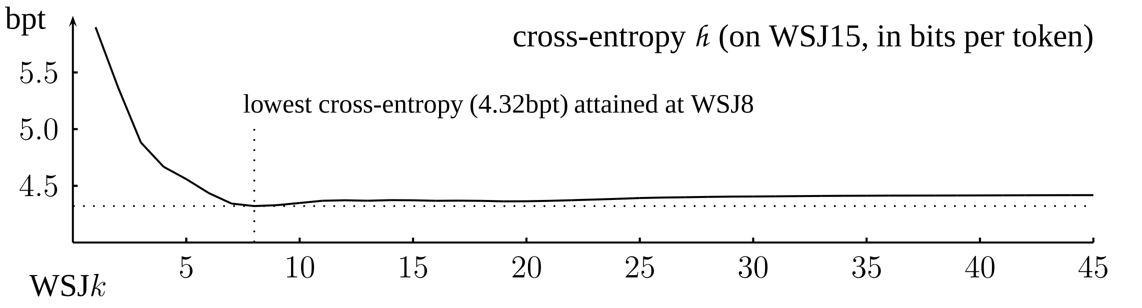


Figure 4.3: Sentence-level cross-entropy for Ad-Hoc* initializers of WSJ$\{1, \ldots, 45\}$.

4.4 Discussion of Experimental Results

The DMV has no parameters to capture syntactic relationships beyond local trees, e.g., agreement. Results from the previous chapter suggest that classic EM breaks down as sentences get longer precisely because the model makes unwarranted independence assumptions: the DMV reserves too much probability mass for what should be unlikely productions. Since EM faithfully allocates such re-distributions across the possible parse trees, once sentences grow sufficiently long, this process begins to deplete what began as likelier structures. But medium lengths avoid a flood of exponentially-confusing longer sentences, as well as the sparseness of unrepresentative shorter ones. The experiments in this chapter corroborate that hypothesis.

First of all, Viterbi manages to hang on to supervised solutions much better than classic EM. Second, Viterbi does not universally degrade with more (complex) training sets, except with a biased initializer. And third, Viterbi learns poorly from small data sets of short sentences (WSJk, $k < 5$). But although Viterbi may be better suited to unsupervised grammar induction compared with classic EM, neither is sufficient, by itself. Both algorithms abandon good solutions and make no guarantees with respect to extrinsic performance. Unfortunately, these two approaches share a deep flaw.

4.5 Related Work on Improper Objectives

It is well-known that maximizing likelihood may, in fact, degrade accuracy [245, 97, 219]. De Marcken [82] showed that classic EM suffers from a fatal attraction towards deterministic grammars and suggested a Viterbi training scheme as a remedy. Liang and Klein's [189] analysis of errors in unsupervised learning began with the inappropriateness of the likelihood objective (approximation), explored problems of data sparsity (estimation) and focused on EM-specific issues related to non-convexity (identifiability and optimization).

Previous literature primarily relied on experimental evidence; de Marcken's analytical result is an exception but pertains only to EM-specific local attractors. The analysis in this chapter confirms his intuitions and moreover shows that there can be *global* preferences for deterministic grammars — problems that would persist with tractable optimization. It proves that there is a fundamental disconnect between objective functions even when likelihood is a reasonable metric and training data are infinite.

4.6 Proofs (by Construction)

There is a subtle distinction between *three* different probability distributions that arise in parsing, each of which can be legitimately termed "likelihood" — the mass that a particular model assigns to (i) highest-scoring (Viterbi) parse trees; (ii) the correct (gold) reference trees; and (iii) the sentence strings (sums over all derivations). A classic unsupervised parser trains to optimize the third, makes actual parsing decisions according to the first, and is evaluated against the second. There are several potential disconnects here. First of all, the true generative model θ^* may not yield the largest margin separations for discriminating between gold parse trees and next best alternatives; and second, θ^* may assign sub-optimal mass to string probabilities. There is no reason why an optimal estimate $\hat{\theta}$ should make the best parser or coincide with a peak of an unsupervised objective.

4.6.1 The Three Likelihood Objectives

A supervised parser finds the "best" parameters $\hat{\theta}$ by maximizing the likelihood of all reference *structures* $t^*(s)$ — the product, over all sentences, of the probabilities that it assigns to each such tree:

$$\hat{\theta}_{\text{SUP}} = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \prod_{s} \mathbb{P}_{\theta}(t^*(s)).$$

For the DMV, this objective function is convex — its unique peak is easy to find and should match the true distribution θ^* given enough data, barring practical problems caused by numerical instability and inappropriate independence assumptions. It is often easier to work in log-probability space:

$$\begin{aligned} \hat{\theta}_{\text{SUP}} &= \arg\max\nolimits_{\theta} \ \log \mathcal{L}(\theta) \\ &= \arg\max\nolimits_{\theta} \textstyle\sum_{s} \log \mathbb{P}_{\theta}(t^*(s)). \end{aligned}$$

Cross-entropy, measured in bits per token (bpt), offers an interpretable proxy for a model's quality:

$$\hbar(\theta) = -\frac{\sum_{s} \lg \mathbb{P}_{\theta}(t^*(s))}{\sum_{s} |s|}.$$

Clearly, $\arg\max_{\theta} \mathcal{L}(\theta) = \hat{\theta}_{\text{SUP}} = \arg\min_{\theta} \hbar(\theta)$.

Unsupervised parsers cannot rely on references and attempt to jointly maximize the probability of each *sentence* instead, summing over the probabilities of all possible trees, according to a model θ:

$$\hat{\theta}_{\text{UNS}} = \arg\max_{\theta} \sum_{s} \log \underbrace{\sum_{t \in T(s)} \mathbb{P}_{\theta}(t)}_{\mathbb{P}_{\theta}(s)}.$$

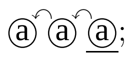
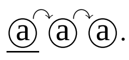
This objective function is not convex and in general does not have a unique peak, so in practice one usually settles for $\tilde{\theta}_{\text{UNS}}$ — a fixed point. There is no reason why $\hat{\theta}_{\text{SUP}}$ should agree with $\hat{\theta}_{\text{UNS}}$, which is in turn (often badly) approximated by $\tilde{\theta}_{\text{UNS}}$, e.g., using EM. A logical alternative to maximizing the probability of sentences is to maximize the probability of the most likely parse trees instead:[4]

$$\hat{\theta}_{\text{VIT}} = \arg\max_{\theta} \sum_{s} \log \mathbb{P}_{\theta}(\hat{t}^{\theta}(s)).$$

This 1-best approximation similarly arrives at $\tilde{\theta}_{\text{VIT}}$, with no claims of optimality. Each next model is re-estimated as if supervised by reference parses.

4.6.2 A Warm-Up Case: Accuracy vs. $\hat{\theta}_{\text{SUP}} \neq \theta^*$

A simple way to derail accuracy is to maximize the likelihood of an incorrect model, e.g., one that makes false independence assumptions. Consider fitting the DMV to a contrived distribution — two equiprobable structures over identical three-token sentences from a unary vocabulary $\{ⓐ\}$:

(i) ; (ii) .

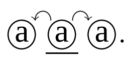
There are six tokens and only two have children on any given side, so adjacent stopping MLEs are:

$$\hat{\mathbb{P}}_{\texttt{STOP}}(ⓐ, \texttt{L}, \texttt{T}) = \hat{\mathbb{P}}_{\texttt{STOP}}(ⓐ, \texttt{R}, \texttt{T}) = 1 - \frac{2}{6} = \frac{2}{3}.$$

The rest of the estimated model is deterministic:

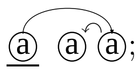
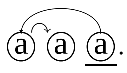
$$\hat{\mathbb{P}}_{\texttt{ATTACH}}(\diamondsuit, \texttt{L}, ⓐ) = \hat{\mathbb{P}}_{\texttt{ATTACH}}(ⓐ, *, ⓐ) = 1 \text{ and } \hat{\mathbb{P}}_{\texttt{STOP}}(ⓐ, *, \texttt{F}) = 1,$$

since all dependents are ⓐ and every one is an only child. But the DMV generates left- and right-attachments independently, allowing a third parse:

(iii) .

[4]It is also possible to use k-best Viterbi, with $k > 1$ (as was later done by Bisk and Hockenmaier [30]).

It also cannot capture the fact that all structures are local (or that all dependency arcs point in the same direction), admitting two additional parse trees:

(iv) ; (v) .

Each possible structure must make four (out of six) adjacent stops, incurring identical probabilities:

$$\hat{\mathbb{P}}_{\texttt{STOP}}(ⓐ, *, \texttt{T})^4 \times (1 - \hat{\mathbb{P}}_{\texttt{STOP}}(ⓐ, *, \texttt{T}))^2 = \frac{2^4}{3^6}.$$

Thus, the MLE model does not break symmetry and rates each of the five parse trees as equally likely. Therefore, its expected per-token accuracy is 40%. Average overlaps between structures (i-v) and answers (i,ii) are (i) 100% or 0; (ii) 0 or 100%; and (iii,iv,v) $33.\overline{3}$%:

$$\frac{3+3}{5 \times 3} = \frac{2}{5} = 0.4.$$

A decoy model without left- or right-branching, i.e.,

$$\tilde{\mathbb{P}}_{\texttt{STOP}}(ⓐ, \texttt{L}, \texttt{T}) = 1 \text{ or } \tilde{\mathbb{P}}_{\texttt{STOP}}(ⓐ, \texttt{R}, \texttt{T}) = 1,$$

would assign zero probability to some of the training data. It would be forced to parse every instance of ⓐⓐⓐ either as (i) or as (ii), deterministically. Nevertheless, it would attain a higher per-token accuracy of 50%. (Judged on exact matches, at the granularity of whole trees, the decoy's guaranteed 50% accuracy clobbers the MLE's expected 20%.)

The toy data set could be replicated n-fold without changing the analysis. This confirms that, even in the absence of estimation errors or data sparsity, there can be a fundamental disconnect between likelihood and accuracy, if the model is wrong.[5]

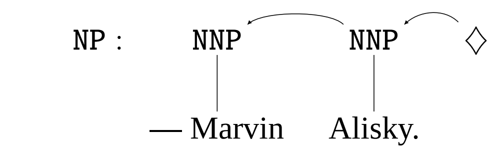
4.6.3 A Subtler Case: $\theta^* = \hat{\theta}_{\text{SUP}}$ vs. $\hat{\theta}_{\text{UNS}}$ vs. $\hat{\theta}_{\text{VIT}}$

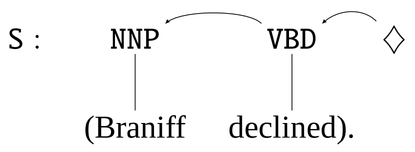
Let's now prove that, even with the *right* model, mismatches between the different objective likelihoods can also handicap the truth. The calculations are again exact, so there are no issues with numerical stability. The set of parameters θ^* is already factored by the DMV,

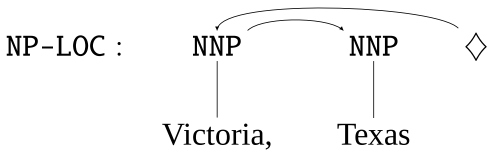
[5]And as George Box quipped, "Essentially, all models are wrong, but some are useful" [35, p. 424].

so that its problems could not be blamed on invalid independence assumptions. Yet it is possible to find another impostor distribution $\tilde{\theta}$ that outshines $\hat{\theta}_{\text{SUP}} = \theta^*$ on both unsupervised metrics, which proves that the true models $\hat{\theta}_{\text{SUP}}$ and θ^* are not globally optimal, as judged by the two surrogate objective functions.

This next example is organic. Starting with WSJ10 confirms that classic EM abandons the supervised solution. Large portions of this data set can then be iteratively discarded, so long as the remainder maintains the (un)desired effect — EM walking away from its $\hat{\theta}_{\text{SUP}}$. This procedure isolates such behavior, arriving at a minimal set:







The above kernel is tiny, but, as before, the analysis is invariant to n-fold replication: the problem cannot be explained away by a small training size — it persists even in infinitely large data sets. And so, consider three reference parse trees for two-token sentences over a binary vocabulary $\{ⓐ, ⓩ\}$:

$$\text{(i) } \overset{\curvearrowleft}{ⓐ\;\underline{ⓐ}}; \quad \text{(ii) } \overset{\curvearrowleft}{ⓐ\;\underline{ⓩ}}; \quad \text{(iii) } \overset{\curvearrowright}{\underline{ⓐ}\;ⓐ}.$$

One third of the time, $\textcircled{z}$ is the head; only $\textcircled{a}$ can be a child; and only $\textcircled{a}$ has right-dependents. Trees (i)-(iii) are the only two-terminal parses generated by the model and are equiprobable. Thus, these sentences are representative of a length-two restriction of everything generated by the true θ^*:

$$\mathbb{P}_{\texttt{ATTACH}}(\diamondsuit, \texttt{L}, \textcircled{a}) = \frac{2}{3} \text{ and } \mathbb{P}_{\texttt{STOP}}(\textcircled{a}, *, \texttt{T}) = \frac{4}{5},$$

since $\textcircled{a}$ is the head two out of three times, and since only one out of five $\textcircled{a}$'s attaches a child on either side. Elsewhere, the model is deterministic:

$$\mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{L}, \texttt{T}) = 0;$$

$$\mathbb{P}_{\texttt{STOP}}(*, *, \texttt{F}) = \mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{R}, \texttt{T}) = 1;$$

$$\mathbb{P}_{\texttt{ATTACH}}(\textcircled{a}, *, \textcircled{a}) = \mathbb{P}_{\texttt{ATTACH}}(\textcircled{z}, \texttt{L}, \textcircled{a}) = 1.$$

Contrast the optimal estimate $\hat{\theta}_{\text{SUP}} = \theta^*$ with the decoy *fixed point*[6] $\tilde{\theta}$ that is identical to θ^*, except

$$\tilde{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{L}, \texttt{T}) = \frac{3}{5} \text{ and } \tilde{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{R}, \texttt{T}) = 1.$$

The probability of stopping is now 3/5 on the left and 1 on the right, instead of 4/5 on both sides — $\tilde{\theta}$ disallows $\textcircled{a}$'s right-dependents but preserves its overall fertility. The probabilities of leaves $\textcircled{a}$ (no children), under the models $\hat{\theta}_{\text{SUP}}$ and $\tilde{\theta}$, are:

$$\hat{\mathbb{P}}(\textcircled{a}) = \hat{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{L}, \texttt{T}) \times \hat{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{R}, \texttt{T}) = \left(\frac{4}{5}\right)^2$$

$$\text{and } \tilde{\mathbb{P}}(\textcircled{a}) = \tilde{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{L}, \texttt{T}) \times \tilde{\mathbb{P}}_{\texttt{STOP}}(\textcircled{a}, \texttt{R}, \texttt{T}) = \frac{3}{5}.$$

And the probabilities of, e.g., structure $\overset{\curvearrowleft}{\textcircled{a}\,\textcircled{z}}$, are:

$$\hat{\mathbb{P}}_{\texttt{ATTACH}}(\diamondsuit, \texttt{L}, \textcircled{z}) \times \hat{\mathbb{P}}_{\texttt{STOP}}(\textcircled{z}, \texttt{R}, \texttt{T}) \times \ (1 - \hat{\mathbb{P}}_{\texttt{STOP}}(\textcircled{z}, \texttt{L}, \texttt{T})) \times \hat{\mathbb{P}}_{\texttt{STOP}}(\textcircled{z}, \texttt{L}, \texttt{F})$$

[6]Models estimated from trees induced by $\tilde{\theta}$ over these sentences are again $\tilde{\theta}$, for both soft and hard EM.

$$\begin{aligned}
&\times\ \hat{\mathbb{P}}_{\texttt{ATTACH}}(ⓩ, \texttt{L}, ⓐ) \times \hat{\mathbb{P}}(ⓐ)\\
&= \hat{\mathbb{P}}_{\texttt{ATTACH}}(\diamondsuit, \texttt{L}, ⓩ) \times \hat{\mathbb{P}}(ⓐ) = \frac{1}{3} \cdot \frac{16}{25}\\
\text{and }\ &\tilde{\mathbb{P}}_{\texttt{ATTACH}}(\diamondsuit, \texttt{L}, ⓩ) \times \tilde{\mathbb{P}}(ⓐ) = \frac{1}{3} \cdot \frac{3}{5}.
\end{aligned}$$

Similarly, the probabilities of all four possible parse trees for the two distinct sentences, ⓐⓐ and ⓐⓩ, under the two models, $\hat{\theta}_{\text{SUP}} = \theta^*$ and $\tilde{\theta}$, are:

	$\hat{\theta}_{\text{SUP}} = \theta^*$	$\tilde{\theta}$
ⓐ $\curvearrowleft$ $\underline{ⓩ}$	$\frac{1}{3}\left(\frac{16}{25}\right) =$	$\frac{1}{3}\left(\frac{3}{5}\right) =$
	$\frac{16}{75} = \mathbf{0.21\overline{3}}$	$\frac{1}{5} = \mathbf{0.2}$
$\underline{ⓐ}$ $\curvearrowright$ ⓩ	$\mathbf{0}$	$\mathbf{0}$
ⓐ $\curvearrowleft$ $\underline{ⓐ}$	$\frac{2}{3}\left(\frac{4}{5}\right)\left(1-\frac{4}{5}\right)\left(\frac{16}{25}\right) =$	$\frac{2}{3}\left(1-\frac{3}{5}\right)\left(\frac{3}{5}\right) =$
	$\frac{128}{1875} = \mathbf{0.0682\overline{6}}$	$\frac{4}{25} = \mathbf{0.16}$
$\underline{ⓐ}$ $\curvearrowright$ ⓐ	$\mathbf{0.0682\overline{6}}$	$\mathbf{0}$

To the three *true parses*, $\hat{\theta}_{\text{SUP}}$ assigns probability $\left(\frac{16}{75}\right)\left(\frac{128}{1875}\right)^2 \approx 0.0009942$ — about 1.66bpt; $\tilde{\theta}$ leaves zero mass for (iii), corresponding to a larger (infinite) cross-entropy, consistent with theory. So far so good, but if asked for *best* (Viterbi) *parses*, $\hat{\theta}_{\text{SUP}}$ could still produce the actual trees, whereas $\tilde{\theta}$ would happily parse sentences of (iii) and (i) the same, perceiving a joint probability of $(0.2)(0.16)^2 = 0.00512$ — just 1.27bpt, appearing to outperform $\hat{\theta}_{\text{SUP}} = \theta^*$! Asked for *sentence probabilities*, $\tilde{\theta}$ would remain unchanged (it parses each sentence unambiguously), but $\hat{\theta}_{\text{SUP}}$ would aggregate to $\left(\frac{16}{75}\right)\left(2 \cdot \frac{128}{1875}\right)^2 \approx 0.003977$, improving to 1.33bpt, but still noticeably "worse" than $\tilde{\theta}$.

Despite leaving zero probability to the truth, $\tilde{\theta}$ beats θ^* on both surrogate metrics, globally. This seems like an egregious error. Judged by (extrinsic) accuracy, $\tilde{\theta}$ still holds its own: it gets four directed edges from predicting parse trees (i) and (ii) completely right, but none of (iii) — a solid 66.7%. Subject to tie-breaking, θ^* is equally likely to get (i) and/or (iii) entirely right or totally wrong (they are indistinguishable): it could earn a perfect 100%, tie $\tilde{\theta}$, or score a low 33.3%, at 1:2:1 odds, respectively — same as $\tilde{\theta}$'s deterministic 66.7% accuracy, in expectation, but with higher variance.

4.7 Discussion of Theoretical Results

Daumé et al. [80] questioned the benefits of using exact models in approximate inference. In the case of grammar induction, the model already makes strong simplifying assumptions *and* the objective is also incorrect. It makes sense that Viterbi EM sometimes works, since an approximate wrong "solution" *could*, by chance, be better than one that is exactly wrong.

One reason why Viterbi EM may work well is that *its* score is used in selecting actual output parse trees. Wainwright [330] provided strong theoretical and empirical arguments for using the same approximate inference method in training as in performing predictions for a learned model. He showed that if inference involves an approximation, then using the same approximate method to train the model gives even better performance guarantees than exact training methods. If the task were not parsing but language modeling, where the relevant score is the sum of the probabilities over individual derivations, perhaps classic EM would not be doing as badly, compared to Viterbi.

Viterbi training is not only faster and more accurate but also free of inside-outside's recursion constraints. It therefore invites more flexible modeling techniques, including discriminative, feature-rich approaches that target *conditional* likelihoods, essentially via (unsupervised) self-training [63, 232, 208, 209, *inter alia*]. Such "learning by doing" approaches may be relevant to understanding human language acquisition, as children frequently find themselves forced to interpret a sentence in order to interact with the world. Since most models of *human* probabilistic parsing are massively pruned [158, 55, 186, *inter alia*], the serial nature of Viterbi EM, or the very limited parallelism of k-best Viterbi, may be more appropriate in modeling this task than fully-integrated inside-outside solutions.[7]

4.8 Conclusion

Without a known objective, as in unsupervised learning, correct exact optimization becomes impossible. In such cases, approximations, although liable to pass over a true optimum, may achieve faster convergence and still *improve* performance. This chapter showed

[7]Following the work in this chapter, k-best Viterbi training [30] and other blends of EM have been applied to both grammar induction [324, 325] (see also next chapter) and other natural language learning tasks [273].

that this is the case with Viterbi training, a cheap alternative to inside-outside re-estimation, for unsupervised dependency parsing.

This chapter explained why Viterbi EM may be particularly well-suited to learning from longer sentences, in addition to any general benefits to synchronizing approximation methods across learning and inference. Its best algorithm is simpler and an order of magnitude faster than classic EM and achieves state-of-the-art performance: 3.8% higher accuracy than previous published best results on Section 23 (all sentences) of the Wall Street Journal corpus. This improvement generalizes to the Brown corpus, the held-out evaluation set, where the same model registers a 7.5% gain.

Unfortunately, approximations alone do not bridge the real gap between objective functions. This deeper issue will be addressed by drawing parsing constraints [245] from specific applications. One example of such an approach, tied to machine translation, is synchronous grammars [11]. An alternative — observing constraints induced by hyper-text markup harvested from the web, punctuation and capitalization — is explored in the second part of this dissertation.

Chapter 5

Lateen EM

This chapter proposes a suite of algorithms that make non-convex optimization with EM less sensitive to local optima, by exploiting the availability of multiple plausible unsupervised objectives, covered in the previous two chapters. Supporting peer-reviewed publication is *Lateen EM: Unsupervised Training with Multiple Objectives, Applied to Dependency Grammar Induction* in EMNLP 2011 [303].

5.1 Introduction

Expectation maximization (EM) algorithms [83] play important roles in learning latent linguistic structure. Unsupervised techniques from this family excel at core natural language processing (NLP) tasks, including segmentation, alignment, tagging and parsing. Typical implementations specify a probabilistic framework, pick an initial model instance, and iteratively improve parameters using EM. A key guarantee is that subsequent model instances are no worse than the previous, according to training data likelihood in the given framework. Another attractive feature that helped make EM instrumental [218] is its initial efficiency: Training tends to begin with large steps in a parameter space, sometimes bypassing many local optima at once. After a modest number of such iterations, however, EM lands close to an attractor. Next, its convergence rate necessarily suffers: Disproportionately many (and ever-smaller) steps are needed to finally approach this fixed point, which is

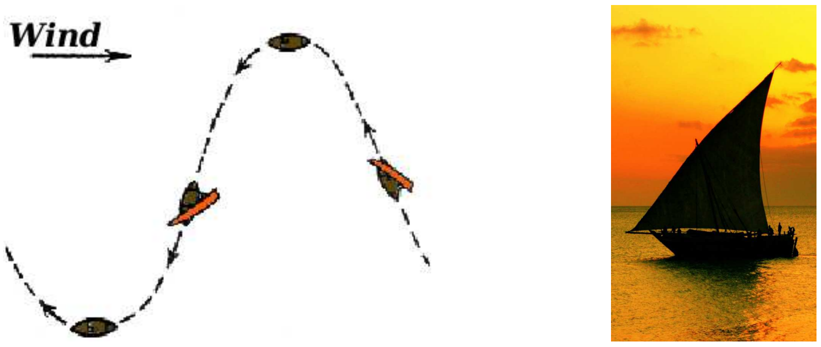


Figure 5.1: A triangular sail atop a traditional Arab sailing vessel, the *dhow* (right). Older square sails permitted sailing only before the wind. But the efficient *lateen* sail worked like a wing (with high pressure on one side and low pressure on the other), allowing a ship to go almost directly into a headwind. By *tacking*, in a zig-zag pattern, it became possible to sail in any direction, provided there was some wind at all (left). For centuries seafarers expertly combined both sails to traverse extensive distances, greatly increasing the reach of medieval navigation. (Partially adapted from `http://www.britannica.com/EBchecked/topic/331395`, `http://allitera.tive.org/archives/004922.html` and `http://landscapedvd.com/desktops/images/ship1280x1024.jpg`.)

almost invariably a local optimum. Deciding when to terminate EM often involves guesswork; and finding ways out of local optima requires trial and error. This chapter proposes several strategies that address both limitations.

Unsupervised objectives are, at best, loosely correlated with extrinsic performance [245, 219, 189, *inter alia*]. This fact justifies (occasionally) deviating from a prescribed training course. For example, since *multiple* equi-plausible objectives are usually available, a learner could cycle through them, optimizing alternatives when the primary objective function gets stuck; or, instead of trying to escape, it could aim to avoid local optima in the first place, by halting search early if an improvement to one objective would come at the expense of harming another. This chapter tests these general ideas by focusing on non-convex likelihood optimization using EM. This setting is standard and has natural and well-understood objectives: the classic, “soft” EM; and Viterbi, or “hard” EM [166]. The name “lateen” comes from the sea — triangular *lateen* sails can take wind on either side, enabling sailing vessels to *tack* (see Figure 5.1). As a captain can’t count on favorable winds, so an unsupervised learner can’t rely on co-operative gradients: soft EM maximizes likelihoods of observed data across assignments to hidden variables, whereas hard EM focuses on most likely completions. These objectives are plausible, yet both can be provably "wrong," as demonstrated in the previous chapter. Thus, it is permissible for lateen EM to maneuver between their gradients, for example by tacking around local attractors, in a zig-zag fashion.

5.2 The Lateen Family of Algorithms

This chapter proposes several strategies that use a secondary objective to improve over standard EM training. For hard EM, the secondary objective is that of soft EM; and vice versa if soft EM is the primary algorithm.

5.2.1 Algorithm #1: Simple Lateen EM

Simple lateen EM begins by running standard EM to convergence, using a user-supplied initial model, primary objective and definition of convergence. Next, the algorithm alternates. A single lateen alternation involves two phases: (i) retraining using the secondary objective, starting from the previous converged solution (once again iterating until convergence, but now of the secondary objective); and (ii) retraining using the primary objective again, starting from the latest converged solution (once more to convergence of the primary objective). The algorithm stops upon failing to sufficiently improve the primary objective across alternations (applying the standard convergence criterion end-to-end) and returns the best of all models re-estimated during training (as judged by the primary objective).

5.2.2 Algorithm #2: Shallow Lateen EM

Same as algorithm #1, but switches back to optimizing the primary objective after a *single* step with the secondary, during phase (i) of all lateen alternations. Thus, the algorithm alternates between optimizing a primary objective to convergence, then stepping away, using one iteration of the secondary optimizer.

5.2.3 Algorithm #3: Early-Stopping Lateen EM

This variant runs standard EM but quits early if the secondary objective suffers. Convergence is redefined by "or"-ing the user-supplied termination criterion (i.e., a "small-enough" change in the primary objective) with *any* adverse change of the secondary (i.e., an increase in its cross-entropy). Early-stopping lateen EM does *not* alternate objectives.

5.2.4 Algorithm #4: Early-Switching Lateen EM

Same as algorithm #1, but with the new definition of convergence, as in algorithm #3. Early-switching lateen EM halts primary optimizers as soon as they hurt the secondary objective and stops secondary optimizers once they harm the primary objective. It terminates upon failing to sufficiently improve the primary objective across a full alternation.

5.2.5 Algorithm #5: Partly-Switching Lateen EM

Same as algorithm #4, but again iterating primary objectives to convergence, as in algorithm #1; secondary optimizers still continue to terminate early.

5.3 The Task and Study #1

This chapter tests the impact of the five lateen algorithms on unsupervised dependency parsing — a task in which EM plays an important role [244, 172, 117, *inter alia*]. It entails two sets of experiments: Study #1 tests whether single alternations of simple lateen EM (as defined in §5.2.1, Algorithm #1) improve a publicly-available system for English dependency grammar induction (from Ch. 6).[1] Study #2 introduces a more sophisticated methodology that uses factorial designs and regressions to evaluate lateen strategies with unsupervised dependency parsing in many languages, after also controlling for other important sources of variation.

For study #1, the base system is an instance of the DMV, trained using hard EM on WSJ45. To confirm that the base model had indeed converged, 10 steps of hard EM on

[1]`http://nlp.stanford.edu/pubs/markup-data.tar.bz2`: `dp.model.dmv`

System	*DDA (%)*
Tree Substitution Grammars [33]	**55.7**
Posterior Sparsity [117]	53.3
Web Markup (Ch. 6)	50.4
+ soft EM + hard EM	52.8 (+**2.4**)
lexicalized, using hard EM	54.3 (+1.5)
+ soft EM + hard EM	**55.6** (+**1.3**)

Table 5.1: Directed dependency accuracies (DDA) on Section 23 of WSJ (all sentences) for contemporary state-of-the-art systems and two experiments (one unlexicalized and one lexicalized) with a single alternation of lateen EM.

WSJ45 were run, verifying that its objective did not change much. Next, a single alternation of simple lateen EM was applied: first running soft EM (this took 101 steps, using the same termination criterion, 2^{-20} bpt), followed by hard EM (again to convergence — another 23 iterations). The result was a decrease in hard EM's cross-entropy, from 3.69 to 3.59 bits per token (bpt), accompanied by a 2.4% jump in accuracy, from 50.4 to 52.8%, on Section 23 of WSJ (see Table 5.1).

The first experiment showed that lateen EM holds promise for simple models. The next test is a more realistic setting: re-estimating *lexicalized* models,[2] starting from the unlexicalized model's parses; this took 24 steps with hard EM. For the second lateen alternation, soft EM ran for 37 steps, hard EM took another 14, and the new model again improved, by 1.3%, from 54.3 to 55.6% (see Table 5.1); the corresponding drop in (lexicalized) cross-entropy was from 6.10 to 6.09 bpt. This last model is competitive with the contemporary state-of-the-art; moreover, gains from single applications of simple lateen alternations (2.4 and 1.3%) are on par with the increase due to lexicalization alone (1.5%).

5.4 Methodology for Study #2

Study #1 suggests that lateen EM can improve grammar induction in English. To establish statistical significance, however, it is important to test a hypothesis in many settings [148].

[2]Using Headden et al.'s [133] method (also the approach of the two stronger state-of-the-art systems): for words seen at least 100 times in the training corpus, gold POS tags are augmented with their lexical items.

Therefore, a factorial experimental design and regression analyses were used, with a variety of lateen strategies. Two regressions — one predicting accuracy, the other, the number of iterations — capture the effects that lateen algorithms have on performance and efficiency, relative to standard EM training. They controlled for important dimensions of variation, such as the underlying language: to make sure that results are not English-specific, grammars were induced for 19 languages. Also explored were the impact from the quality of an initial model (using both uniform and ad hoc initializers), the choice of a primary objective (i.e., soft or hard EM), and the quantity and complexity of training data (shorter versus both short and long sentences). Appendix gives the full details.

5.5 Experiments

All 23 train/test splits from the 2006/7 CoNLL shared tasks are used [42, 236]. These disjoint splits require smoothing (in the WSJ setting, training and test sets overlapped). All punctuation labeled in the data is spliced out, as is standard practice [244, 172], introducing new arcs from grand-mothers to grand-daughters where necessary, both in train- and test-sets. Thus, punctuation does not affect scoring. An optimizer is always halted once a change in its objective's consecutive cross-entropy values falls below 2^{-20} bpt, at which point it is considered "stuck." All unsmoothed models are smoothed immediately prior to evaluation; some of the baseline models are also smoothed during training. In both cases, the "add-one" (a.k.a. Laplace) smoothing algorithm is used.

5.5.1 Baseline Models

This chapter tests a total of six baseline models, experimenting with two types of alternatives: (i) strategies that perturb stuck models directly, by *smoothing*, ignoring secondary objectives; and (ii) *shallow* applications of a single EM step, ignoring convergence.

Baseline $B1$ alternates running standard EM to convergence and smoothing. A second baseline, $B2$, smooths after every step of EM instead. Another shallow baseline, $B3$, alternates single steps of soft and hard EM.[3] Three such baselines begin with hard EM (marked

[3]It approximates a mixture (the average of soft and hard objectives) — a natural comparison, computable

with the subscript h); and three more start with soft EM (marked with the subscript s).

5.5.2 Lateen Models

Ten models, $A\{1,2,3,4,5\}_{\{h,s\}}$, correspond to the lateen algorithms #1–5 (§5.2), starting with either hard or soft EM's objective, to be used as the primary.

5.6 Results

		Soft EM		Hard EM	
	Model	Δa	Δi	Δa	Δi
Baselines	$B3$	**-2.7**	$\times$**0.2**	**-2.0**	$\times$**0.3**
	$B2$	+0.6	$\times$**0.7**	+0.6	$\times$1.2
	$B1$	0.0	$\times$**2.0**	+0.8	$\times$**3.7**
Algorithms	$A1$	0.0	$\times$**1.3**	**+5.5**	$\times$**6.5**
	$A2$	-0.0	$\times$**1.3**	+1.5	$\times$**3.6**
	$A3$	0.0	$\times$**0.7**	-0.1	$\times$**0.7**
	$A4$	0.0	$\times$**0.8**	**+3.0**	$\times$**2.1**
	$A5$	0.0	$\times$**1.2**	**+2.9**	$\times$**3.8**

Table 5.2: Estimated additive changes in directed dependency accuracy (Δa) and multiplicative changes in the number of iterations before terminating (Δi) for all baseline models and lateen algorithms, relative to standard training: soft EM (left) and hard EM (right). Bold entries are statistically different ($p < 0.01$) from zero, for Δa, and one, for Δi (details in Table 5.4 and Appendix).

No baseline attained a statistically significant performance improvement. Shallow models $B3_{\{h,s\}}$, in fact, significantly lowered accuracy: by 2.0%, on average ($p \approx 7.8 \times 10^{-4}$), for $B3_h$, which began with hard EM; and down 2.7% on average ($p \approx 6.4 \times 10^{-7}$), for $B3_s$, started with soft EM. They were, however, 3–5x faster than standard training, on average (see Table 5.4 for all estimates and associated p-values; above, Table 5.2 shows a preview of the full results).

via gradients and standard optimization algorithms, such as L-BFGS [192]. Exact interpolations are not explored because replacing EM is itself a significant confounder, even with unchanged objectives [24].

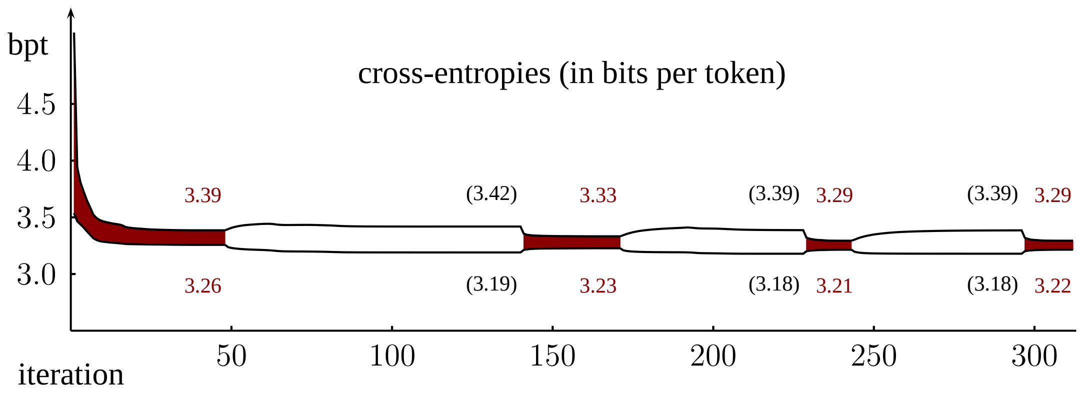


Figure 5.2: Cross-entropies for Italian '07, initialized uniformly and trained on sentences up to length 45. The two curves are primary and secondary objectives (soft EM's lies below, as sentence yields are at least as likely as parse trees): shaded regions indicate iterations of hard EM (primary); and annotated values are measurements upon each optimizer's convergence (soft EM's are parenthesized).

5.6.1 $A1_{\{h,s\}}$ — Simple Lateen EM

$A1_h$ runs 6.5x slower, but scores 5.5% higher, on average, compared to standard Viterbi training; $A1_s$ is only 30% slower than standard soft EM, but does not impact its accuracy at all, on average. Figure 5.2 depicts a sample training run: Italian '07 with $A1_h$. Viterbi EM converges after 47 iterations, reducing the primary objective to 3.39 bpt (the secondary is then at 3.26); accuracy on the held-out set is 41.8%. Three alternations of lateen EM (totaling 265 iterations) further decrease the primary objective to 3.29 bpt (the secondary also declines, to 3.22) and accuracy increases to 56.2% (14.4% higher).

5.6.2 $A2_{\{h,s\}}$ — Shallow Lateen EM

$A2_h$ runs 3.6x slower, but scores only 1.5% higher, on average, compared to standard Viterbi training; $A2_s$ is again 30% slower than standard soft EM and also has no measurable impact on parsing accuracy.

5.6.3 $A3_{\{h,s\}}$ — Early-Stopping Lateen EM

Both $A3_h$ and $A3_s$ run 30% faster, on average, than standard training with hard or soft EM; and neither heuristic causes a statistical change to accuracy. Table 5.3 shows accuracies and iteration counts for 10 (of 23) train/test splits that terminate early with $A3_s$ (in one particular, example setting). These runs are nearly twice as fast, and only two score (slightly) lower, compared to standard training using soft EM.

5.6.4 $A4_{\{h,s\}}$ — Early-Switching Lateen EM

$A4_h$ runs only 2.1x slower, but scores only 3.0% higher, on average, compared to standard Viterbi training; $A4_s$ is, in fact, 20% faster than standard soft EM, but still has no measurable impact on accuracy.

5.6.5 $A5_{\{h,s\}}$ — Partly-Switching Lateen EM

$A5_h$ runs 3.8x slower, scoring 2.9% higher, on average, compared to standard Viterbi training; $A5_s$ is 20% slower than soft EM, but, again, no more accurate. Indeed, $A4$ strictly dominates both $A5$ variants.

5.7 Discussion

Lateen strategies improve dependency grammar induction in several ways. Early stopping offers a clear benefit: 30% higher efficiency yet same performance as standard training. This technique could be used to (more) fairly compare learners with radically different objectives (e.g., lexicalized and unlexicalized), requiring quite different numbers of steps — or magnitude changes in cross-entropy — to converge.

The second benefit is improved performance, but only starting with hard EM. Initial local optima discovered by soft EM are such that the impact on accuracy of all subsequent heuristics is indistinguishable from noise (it's not even negative). But for hard EM, lateen strategies consistently improve accuracy — by 1.5, 3.0 or 5.5% — as an algorithm follows the secondary objective longer (a single step, until the primary objective gets worse, or to

CoNLL Year & Language		Soft EM *DDA*	Soft EM *iters*	$A3_s$ *DDA*	$A3_s$ *iters*
Arabic	2006	28.4	180	28.4	118
Bulgarian	’06	39.1	253	**39.6**	131
Chinese	’06	49.4	268	49.4	204
Dutch	’06	21.3	246	**27.8**	35
Hungarian	’07	17.1	366	**17.4**	213
Italian	’07	39.6	194	39.6	164
Japanese	’06	56.6	113	56.6	93
Portuguese	’06	37.9	180	*37.5*	102
Slovenian	’06	30.8	234	**31.1**	118
Spanish	’06	33.3	125	*33.1*	73
	Average:	35.4	216	36.1	125

Table 5.3: Directed dependency accuracies (DDA) and iteration counts for the 10 (of 23) train/test splits affected by early termination (setting: soft EM’s primary objective, trained using shorter sentences and ad-hoc initialization).

convergence). These results suggest that soft EM should use early termination to improve efficiency. Hard EM, by contrast, could use any lateen strategy to improve either efficiency or performance, or to strike a balance.

5.8 Related Work

5.8.1 Avoiding and/or Escaping Local Attractors

Simple lateen EM is similar to Dhillon et al.’s [84] refinement algorithm for text clustering with spherical k-means. Their “ping-pong” strategy alternates batch and incremental EM, exploits the strong points of each, and improves a *shared* objective at every step. Unlike generalized (GEM) variants [229], lateen EM uses multiple objectives: it sacrifices the primary in the short run, to escape local optima; in the long run, it also does no harm, by construction (as it returns the best model seen). Of the meta-heuristics that use more than a standard, scalar objective, deterministic annealing (DA) [268] is closest to lateen EM. DA perturbs objective functions, instead of manipulating solutions directly. As other continuation methods [5], it optimizes an easy (e.g., convex) function first, then “rides” that optimum by gradually morphing functions towards the difficult objective; each step reoptimizes from the previous approximate solution. Smith and Eisner [292] employed DA to improve part-of-speech disambiguation, but found that objectives had to be further "skewed," using domain knowledge, before it helped (constituent) grammar induction. (For this reason, this chapter does not experiment with DA, despite its strong similarities to lateen EM.) Smith and Eisner used a "temperature" β to anneal a flat uniform distribution ($\beta = 0$) into soft EM's non-convex objective ($\beta = 1$). In their framework, hard EM corresponds to $\beta \longrightarrow \infty$, so the algorithms differ only in their β-schedule: DA's is continuous, from 0 to 1; lateen EM's is a discrete alternation, of 1 and $+\infty$ (a kind of "beam search" [194], with soft EM expanding and hard EM pruning a frontier).

5.8.2 Terminating Early, Before Convergence

EM is rarely run to (even numerical) convergence. Fixing a modest number of iterations a priori [170, §5.3.4], running until successive likelihood ratios become small [301, §4.1] or using a combination of the two [261, §4, Footnote 5] is standard practice in NLP. Elworthy's [97, §5, Figure 1] analysis of part-of-speech tagging showed that, in most cases, a small number of iterations is actually preferable to convergence, in terms of final accuracies: "regularization by early termination" had been suggested for image deblurring algorithms in statistical astronomy [197, §2]; and validation against held-out data — a strategy proposed much earlier, in psychology [179], has also been used as a halting criterion in NLP [344, §4.2, 5.2]. Early-stopping lateen EM tethers termination to a *sign* change in the direction of a secondary objective, similarly to (cross-)validation [314, 112, 12], but without splitting data — it trains using all examples, at all times.[4,5]

[4]It can be viewed as a milder contrastive estimation [293, 294], agnostic to implicit negative evidence, but caring *whence* learners push probability mass towards training examples: when most likely parse trees begin to benefit at the expense of their sentence yields (or vice versa), optimizers halt.

[5]For a recently proposed instance of EM that uses cross-validation (CV) to optimize *smoothed* data likelihoods (in learning synchronous PCFGs, for phrase-based machine translation), see Mylonakis and Sima'an's [226, §3.1] CV-EM algorithm.

5.8.3 Training with Multiple Views

Lateen strategies may seem conceptually related to co-training [32]. However, bootstrapping methods generally begin with some labeled data and gradually label the rest (discriminatively) as they grow more confident, but do not optimize an explicit objective function; EM, on the other hand, can be fully unsupervised, relabels all examples on each iteration (generatively), and guarantees not to hurt a well-defined objective, at every step.[6] Co-training classically relies on two views of the data — redundant feature sets that allow different algorithms to label examples for each other, yielding "probably approximately correct" (PAC)-style guarantees under certain (strong) assumptions. In contrast, lateen EM uses the same data, features, model and essentially the same algorithms, changing only their objective functions: it makes no assumptions, but guarantees not to harm the primary objective. Some of these distinctions have become blurred with time: Collins and Singer [72] introduced an objective function (also based on agreement) into co-training; Goldman and Zhou [126], Ng and Cardie [232] and Chan et al. [46] made do without redundant views; Balcan et al. [15] relaxed other strong assumptions; and Zhou and Goldman [347] generalized co-training to accommodate three and more algorithms. Several such methods have been applied to dependency parsing [298], constituent parsing [277] and parser reranking [76]. Fundamentally, co-training exploits redundancies in unlabeled data and/or learning algorithms. Lateen strategies *also* exploit redundancies: in noisy objectives. Both approaches use a second vantage point to improve their perception of difficult training terrains.

5.9 Conclusions and Future Work

Lateen strategies can improve performance and efficiency for dependency grammar induction with the DMV. Early-stopping lateen EM is 30% faster than standard training, without affecting accuracy — it reduces guesswork in terminating EM. At the other extreme, simple lateen EM is slower, but significantly improves accuracy — by 5.5%, on average — for hard EM, escaping some of its local optima. Future work could explore other NLP tasks

[6]Some authors [234, 232, 293] draw a hard line between bootstrapping algorithms, such as self- and co-training, and probabilistic modeling using EM; others [78, 47] tend to lump them together.

— such as clustering, sequence labeling, segmentation and alignment — that often employ EM. The new meta-heuristics are multi-faceted, featuring aspects of iterated local search, deterministic annealing, cross-validation, contrastive estimation and co-training. They may be generally useful in machine learning and non-convex optimization.

5.10 Appendix on Experimental Design

Statistical techniques are vital to many aspects of computational linguistics [155, 50, 3, *inter alia*]. This chapter used factorial designs,[7] which are standard throughout the natural and social sciences, to assist with experimental design and statistical analyses. Combined with ordinary regressions, these methods provide succinct and interpretable summaries that explain which settings meaningfully contribute to changes in dependent variables, such as running time and accuracy.

5.10.1 Dependent Variables

Two regressions were constructed, for two types of dependent variables: to summarize performance, accuracies were predicted; and to summarize efficiency, (logarithms of) iterations before termination.

In the performance regression, four different scores were used for the dependent variable. These include both directed accuracies and *undirected* accuracies, each computed in two ways: (i) using a best parse tree; and (ii) using all parse trees. These four types of scores provide different kinds of information. Undirected scores ignore polarity of parent-child relations [244, 172, 282], partially correcting for some effects of alternate analyses (e.g., systematic choices between modals and main verbs for heads of sentences, determiners for noun phrases, etc.). And *integrated* scoring, using the inside-outside algorithm [14] to compute expected accuracy across all — not just best — parse trees, has the advantage of incorporating probabilities assigned to individual arcs: This metric is more sensitive to the margins that separate best from next-best parse trees, and is not affected by tie-breaking.

[7]It used *full* factorial designs for clarity of exposition. But many fewer experiments would suffice, especially in regression models without interaction terms: for the more efficient *fractional* factorial designs, as well as for randomized block designs and full factorial designs, see Montgomery [223, Ch. 4–9].

		Regression for **Accuracies**		Regression for $\ln($**Iterations**$)$		
	Goodness-of-Fit:	($R^2_{adj} \approx 76.2\%$)		($R^2_{adj} \approx 82.4\%$)		
	Indicator Factors	*coeff.* $\hat{\beta}$	*adj.* p*-value*	*coeff.* $\hat{\beta}$	*mult.* $e^{\hat{\beta}}$	*adj.* p*-value*
	undirected	**18.1**	$< 2.0 \times 10^{-16}$			
	integrated	**-0.9**	$\approx 7.0 \times 10^{-7}$			
	(intercept)	**30.9**	$< 2.0 \times 10^{-16}$	5.5	**255.8**	$< 2.0 \times 10^{-16}$
	adhoc	**1.2**	$\approx 3.1 \times 10^{-13}$	-0.0	1.0	≈ 1.0
Model	*sweet*	**1.0**	$\approx 3.1 \times 10^{-9}$	-0.2	**0.8**	$< 2.0 \times 10^{-16}$
$B3_s$	*shallow (soft-first)*	**-2.7**	$\approx 6.4 \times 10^{-7}$	-1.5	**0.2**	$< 2.0 \times 10^{-16}$
$B3_h$	*shallow (hard-first)*	**-2.0**	$\approx 7.8 \times 10^{-4}$	-1.2	**0.3**	$< 2.0 \times 10^{-16}$
$B2_s$	*shallow smooth*	0.6	≈ 1.0	-0.4	**0.7**	$\approx 1.4 \times 10^{-12}$
$B1_s$	*smooth*	0.0	≈ 1.0	0.7	**2.0**	$< 2.0 \times 10^{-16}$
$A1_s$	*simple lateen*	0.0	≈ 1.0	0.2	**1.3**	$\approx 4.1 \times 10^{-4}$
$A2_s$	*shallow lateen*	-0.0	≈ 1.0	0.2	**1.3**	$\approx 5.8 \times 10^{-4}$
$A3_s$	*early-stopping lateen*	0.0	≈ 1.0	-0.3	**0.7**	$\approx 2.6 \times 10^{-7}$
$A4_s$	*early-switching lateen*	0.0	≈ 1.0	-0.3	**0.8**	$\approx 2.6 \times 10^{-7}$
$A5_s$	*partly-switching lateen*	0.0	≈ 1.0	0.2	**1.2**	$\approx 4.2 \times 10^{-3}$
	viterbi	**-4.0**	$\approx 5.7 \times 10^{-16}$	-1.7	**0.2**	$< 2.0 \times 10^{-16}$
$B2_h$	*shallow smooth*	0.6	≈ 1.0	0.2	1.2	$\approx 5.6 \times 10^{-2}$
$B1_h$	*smooth*	0.8	≈ 1.0	1.3	**3.7**	$< 2.0 \times 10^{-16}$
$A1_h$	*simple lateen*	**5.5**	$< 2.0 \times 10^{-16}$	1.9	**6.5**	$< 2.0 \times 10^{-16}$
$A2_h$	*shallow lateen*	1.5	$\approx 5.0 \times 10^{-2}$	1.3	**3.6**	$< 2.0 \times 10^{-16}$
$A3_h$	*early-stopping lateen*	-0.1	≈ 1.0	-0.4	**0.7**	$\approx 1.7 \times 10^{-11}$
$A4_h$	*early-switching lateen*	**3.0**	$\approx 1.0 \times 10^{-8}$	0.7	**2.1**	$< 2.0 \times 10^{-16}$
$A5_h$	*partly-switching lateen*	**2.9**	$\approx 7.6 \times 10^{-8}$	1.3	**3.8**	$< 2.0 \times 10^{-16}$

Table 5.4: Regressions for accuracies and natural-log-iterations, using 86 binary predictors (all p-values jointly adjusted for simultaneous hypothesis testing; $\{$*langyear*$\}$ indicators not shown). Accuracies' estimated coefficients $\hat{\beta}$ that are statistically different from 0 — and iteration counts' multipliers $e^{\hat{\beta}}$ significantly different from 1 — are shown in bold.

Scores were tagged using two binary predictors in a simple (first order, multi-linear) regression, where having multiple relevant quality assessments improves goodness-of-fit.

In the efficiency regression, dependent variables were logarithms of the numbers of iterations. Wrapping EM in an inner loop of a heuristic has a multiplicative effect on the total number of models re-estimated prior to termination. Consequently, logarithms of the final counts better fit the observed data (however, since the logarithm is concave, the price of this better fit is a slight bias towards overestimating the coefficients).

5.10.2 Independent Predictors

All of the predictors are binary indicators (a.k.a. "dummy" variables). The *undirected* and *integrated* factors only affect the regression for accuracies (see Table 5.4, left); remaining factors participate also in the running times regression (see Table 5.4, right). In a default run, all factors are zero, corresponding to the intercept estimated by a regression; other estimates reflect changes in the dependent variable associated with having that factor "on" instead of "off."

- *adhoc* — This setting controls initialization. By default, the uninformed uniform initializer is used; when it is on, Ad-Hoc*, bootstrapped using sentences up to length 10, from the training set, is used.

- *sweet* — This setting controls the length cutoff. By default, training is with all sentences containing up to 45 tokens; when it is on, the "sweet spot" cutoff of 15 tokens (recommended for English, WSJ) is used.

- *viterbi* — This setting controls the primary objective of the learning algorithm. By default, soft EM is run; when it is on, hard EM is run.

- $\{\textit{langyear}_i\}_{i=1}^{22}$ — This is a set of 22 mutually-exclusive selectors for the language/year of a train/test split; default (all zeros) is English '07.

Due to space limitations, *langyear* predictors are excluded from Table 5.4. Further, interactions between predictors are not explored. This approach may miss some interesting facts, e.g., that the *adhoc* initializer is exceptionally good for English, with soft EM. Instead it yields coarse summaries of regularities supported by overwhelming evidence across data and training regimes.

5.10.3 Statistical Significance

All statistical analyses relied on the R package [258], which does not, by default, adjust statistical significance (p-values) for multiple hypotheses testing.[8] This was corrected using the Holm-Bonferroni method [141], which is uniformly more powerful than the older (Dunn-)Bonferroni procedure; since many fewer hypotheses ($44 + 42$ — one per intercept/coefficient $\hat{\beta}$) than settings combinations were tested, its adjustments to the p-values

[8]Since one would *expect* $p\%$ of randomly chosen hypotheses to appear significant at the $p\%$ level simply by *chance*, one must take precautions against these and other "data-snooping" biases.

CoNLL Year		$A3_s$		Soft EM		$A3_h$		Hard EM		$A1_h$	
& Language		*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*
Arabic	2006	28.4	118	28.4	162	21.6	19	21.6	21	**32.1**	200
	’7	–	–	**26.9**	171	24.7	17	24.8	24	22.0	239
Basque	’7	–	–	39.9	180	32.0	16	32.2	20	**43.6**	128
Bulgarian	’6	39.6	131	39.1	253	41.6	22	41.5	25	**44.3**	140
Catalan	’7	–	–	58.5	135	50.1	48	50.1	54	**63.8**	279
Chinese	’6	**49.4**	204	49.4	268	31.3	24	31.6	55	37.9	378
	’7	–	–	**46.0**	262	30.0	25	30.2	64	34.5	307
Czech	’6	–	–	**50.5**	294	27.8	27	27.7	33	35.2	445
	’7	–	–	**49.8**	263	29.0	37	29.0	41	31.4	307
Danish	’6	–	–	43.5	116	43.8	31	43.9	45	**44.0**	289
Dutch	’6	27.8	35	21.3	246	24.9	44	24.9	49	**32.5**	241
English	’7	–	–	**38.1**	180	34.0	32	33.9	42	34.9	186
German	’6	–	–	33.3	136	25.4	20	25.4	39	**33.5**	155
Greek	’7	–	–	17.5	230	18.3	18	18.3	21	**21.4**	117
Hungarian	’7	17.4	213	17.1	366	12.3	26	12.4	36	**23.0**	246
Italian	’7	**39.6**	164	39.6	194	32.6	25	32.6	27	37.6	273
Japanese	’6	**56.6**	93	56.6	113	49.6	20	49.7	23	53.5	91
Portuguese	’6	37.5	102	**37.9**	180	28.6	27	28.9	41	34.4	134
Slovenian	’6	31.1	118	30.8	234	–	–	23.4	22	**33.6**	255
Spanish	’6	33.1	73	**33.3**	125	18.2	29	18.4	36	33.3	235
Swedish	’6	–	–	41.8	242	36.0	24	36.1	29	**42.5**	296
Turkish	’6	–	–	29.8	303	17.8	19	22.2	38	**31.9**	134
	’7	–	–	28.3	227	14.0	9	10.7	31	**33.4**	242
	Average:	**37.4**	162	37.0	206	30.0	26	30.0	35	37.1	221

Table 5.5: Performance (directed dependency accuracies measured against all sentences in the evaluation sets) and efficiency (numbers of iterations) for standard training (soft and hard EM), early-stopping lateen EM ($A3$) and simple lateen EM with hard EM’s primary objective ($A1_h$), for all 23 train/test splits, with *adhoc* and *sweet* settings on.

are small (see Table 5.4).[9]

5.10.4 Interpretation

Table 5.4 shows the estimated coefficients and their (adjusted) p-values for both intercepts and most predictors (excluding the language/year of the data sets) for all 1,840 experiments. The default (English) system uses soft EM, trains with both short and long sentences, and starts from an uninformed uniform initializer. It is estimated to score 30.9%, converging after approximately 256 iterations (both intercepts are statistically different from zero: $p < 2.0 \times 10^{-16}$). As had to be the case, a gain is detected from *undirected* scoring; *integrated* scoring is slightly (but significantly: $p \approx 7.0 \times 10^{-7}$) negative, which is reassuring: best

[9]The p-values for all 86 hypotheses were adjusted jointly, using `http://rss.acs.unt.edu/Rdoc/library/multtest/html/mt.rawp2adjp.html`.

CoNLL Year		$A3_s$		Soft EM		$A3_h$		Hard EM		$A1_h$	
& Language		*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*	*DDA*	*iters*
Arabic	2006	–	–	**33.4**	317	20.8	8	20.2	32	16.6	269
	'7	18.6	60	8.7	252	26.5	9	26.4	14	**49.5**	171
Basque	'7	–	–	18.3	245	23.2	16	23.0	23	**24.0**	162
Bulgarian	'6	27.0	242	27.1	293	40.6	33	40.5	34	**43.9**	276
Catalan	'7	15.0	74	13.8	159	53.2	30	53.1	31	**59.8**	176
Chinese	'6	63.5	131	**63.6**	261	36.8	45	36.8	47	44.5	213
	'7	**58.5**	130	58.5	258	35.2	20	35.0	48	43.2	372
Czech	'6	29.5	125	**29.7**	224	23.6	18	23.8	41	27.7	179
	'7	–	–	25.9	215	27.1	37	27.2	64	**28.4**	767
Danish	'6	–	–	16.6	155	28.7	30	28.7	30	**38.3**	241
Dutch	'6	20.4	51	21.2	174	25.5	30	25.6	38	**27.8**	243
English	'7	–	–	18.0	162	–	–	38.7	35	**45.2**	366
German	'6	–	–	24.4	148	30.1	39	30.1	44	**30.4**	185
Greek	'7	**25.5**	133	25.3	156	–	–	13.2	27	13.2	252
Hungarian	'7	–	–	18.9	310	28.9	34	28.9	44	**34.7**	414
Italian	'7	25.4	127	25.3	165	–	–	**52.3**	36	52.3	81
Japanese	'6	–	–	39.3	143	42.2	38	42.4	48	**50.2**	199
Portuguese	'6	35.2	48	35.6	224	–	–	34.5	21	**36.7**	143
Slovenian	'6	24.8	182	25.3	397	28.8	17	28.8	20	**32.2**	121
Spanish	'6	–	–	27.7	252	–	–	28.3	31	**50.6**	130
Swedish	'6	27.9	49	32.6	287	45.2	22	45.6	52	**50.0**	314
Turkish	'6	–	–	30.5	239	30.2	16	**30.6**	24	29.0	138
	'7	–	–	**48.8**	254	34.3	24	33.1	34	35.9	269
	Average:	27.3	161	27.3	225	33.2	28	33.2	35	**38.2**	236

Table 5.6: Performance (directed dependency accuracies measured against all sentences in the evaluation sets) and efficiency (numbers of iterations) for standard training (soft and hard EM), early-stopping lateen EM ($A3$) and simple lateen EM with hard EM's primary objective ($A1_h$), for all 23 train/test splits, with setting *adhoc* off and *sweet* on.

parses are scoring higher than the rest and may be standing out by large margins. The *adhoc* initializer boosts accuracy by 1.2%, overall (also significant: $p \approx 3.1 \times 10^{-13}$), without a measurable impact on running time ($p \approx 1.0$). Training with fewer, shorter sentences, at the *sweet* spot gradation, adds 1.0% and shaves 20% off the total number of iterations, on average (both estimates are significant).

The *viterbi* objective is found harmful — by 4.0%, on average ($p \approx 5.7 \times 10^{-16}$) — for the CoNLL sets. Half of these experiments are with shorter sentences, and half use ad hoc initializers (i.e., three quarters of settings are not ideal for Viterbi EM), which may have contributed to this negative result; still, the estimates do confirm that hard EM is significantly (80%, $p < 2.0 \times 10^{-16}$) faster than soft EM.

5.10.5 More on Viterbi Training

The overall negative impact of Viterbi objectives is a cause for concern: On average, $A1_h$'s estimated gain of 5.5% should more than offset the expected 4.0% loss from starting with hard EM. But it is, nevertheless, important to make sure that simple lateen EM with hard EM's primary objective is in fact an improvement over *both* standard EM algorithms.

Table 5.5 shows performance and efficiency numbers for $A1_h$, $A3_{\{h,s\}}$, as well as standard soft and hard EM, using settings that are least favorable for Viterbi training: *adhoc* and *sweet* on. Although $A1_h$ scores 7.1% higher than hard EM, on average, it is only slightly better than soft EM — up 0.1% (and worse than $A3_s$). Without *adhoc* (i.e., using uniform initializers — see Table 5.6), however, hard EM still improves, by 3.2%, on average, whereas soft EM drops nearly 10%; here, $A1_h$ further improves over hard EM, scoring 38.2% (up 5.0), higher than soft EM's accuracies from *both* settings (27.3 and 37.0).

This suggests that $A1_h$ is indeed better than both standard EM algorithms. This chapter's experimental set-up may be disadvantageous for Viterbi training, since half the settings use ad hoc initializers, and because CoNLL sets are small. Viterbi EM works best with more data and longer sentences.

Part II

Constraints

Chapter 6

Markup

The purpose of this chapter is to explore ways of constraining a grammar induction process, to make up for the deficiencies of unsupervised objectives, and to quantify the extent to which naturally-occurring annotations by laymen that might be used as a guide, such as web markup, agree with syntactic analyses rooted in linguistic theories or could be of help. Supporting peer-reviewed publication is *Profiting from Mark-Up: Hyper-Text Annotations for Guided Parsing* in ACL 2010 [311].

6.1 Introduction

Pereira and Schabes [245] outlined three major problems with classic EM, applied to the related problem of constituent parsing. They extended classic inside-outside re-estimation [14] to respect any bracketing constraints included with a training corpus. This conditioning on partial parses addressed several problems, leading to: (i) linguistically reasonable constituent boundaries and induced grammars more likely to agree with qualitative judgments of sentence structure, which is underdetermined by unannotated text; (ii) fewer iterations needed to reach a good grammar, countering convergence properties that sharply deteriorate with the number of non-terminal symbols, due to a proliferation of local maxima; and (iii) better (in the best case, linear) time complexity per iteration, versus running time that is ordinarily cubic in both sentence length *and* the total number of non-terminals, rendering sufficiently large grammars computationally impractical. Their algorithm sometimes found good solutions from bracketed corpora but not from raw text, supporting the view that purely unsupervised, self-organizing inference methods can miss the trees for the forest of distributional regularities. This was a promising break-through, but the problem of whence to get partial bracketings was left open.

This chapter suggests mining partial bracketings from a cheap and abundant natural language resource: the hyper-text markup that annotates web-pages. For example, consider that anchor text can match linguistic constituents, such as verb phrases, exactly:

..., whereas McCain is secure on the topic,
Obama `<a>`[$_{\mathbf{VP}}$ worries about winning the pro-Israel vote]`</a>`.

Validating this idea involved the creation of a new data set, novel in combining a real blog's raw HTML with tree-bank-like constituent structure parses, generated automatically. A linguistic analysis of the most prevalent tags (anchors, bold, italics and underlines) over its 1M$^+$ words reveals a strong connection between syntax and markup (all of this chapter's examples draw from this corpus), inspiring several simple techniques for automatically deriving parsing constraints. Experiments with both hard and more flexible constraints, as well as with different styles and quantities of annotated training data — the blog, web news and the web itself, confirm that markup-induced constraints consistently improve (otherwise unsupervised) dependency parsing.

6.2 Intuition and Motivating Examples

It is natural to expect hidden structure to seep through when a person annotates a sentence. As it happens, a non-trivial fraction of the world's population routinely annotates text diligently, if only partially and informally.[1] They inject hyper-links, vary font sizes, and toggle colors and styles, using markup technologies such as HTML and XML.

As noted, web annotations can be indicative of phrase boundaries, e.g., in a complicated sentence:

In 1998, however, as I `<a>`[$_{\mathbf{VP}}$ established in `<i>`[$_{\mathbf{NP}}$ *The New Republic*]`</i>`]`</a>` and Bill
Clinton just `<a>`[$_{\mathbf{VP}}$ confirmed in his memoirs]`</a>`, Netanyahu changed his mind and ...

[1]Even when (American) grammar schools lived up to their name, they only taught dependencies. This was back in the days before constituent grammars were invented.

In doing so, markup sometimes offers useful cues even for low-level tokenization decisions:

[$_{\mathbf{NP}}$ [$_{\mathbf{NP}}$ Libyan ruler]
<a>[$_{\mathbf{NP}}$ Mu‘ammar al-Qaddafi]</a>] referred to ...

```
(NP (ADJP (NP (JJ Libyan) (NN ruler))
          (JJ Mu))
    (‘‘ ‘) (NN ammar) (NNS al-Qaddafi))
```

At one point in time, a backward quote in an Arabic name confused some parsers (see above).[2] Yet markup lines up with the broken noun phrase, signals cohesion, and moreover sheds light on the internal structure of a compound. As Vadas and Curran [327] point out, such details are frequently omitted even from manually compiled tree-banks that err on the side of flat annotations of base-NPs. Admittedly, not all boundaries between HTML tags and syntactic constituents match up nicely:

..., but [$_{\mathbf{S}}$ [$_{\mathbf{NP}}$ the <a><i>*Toronto Star*</i>]
[$_{\mathbf{VP}}$ reports [$_{\mathbf{NP}}$ this] [$_{\mathbf{PP}}$ in the softest possible way]</a>,[$_{\mathbf{S}}$ stating only that ...]]]

Combining parsing with markup may not be straight-forward, but there is hope: even above, one of each nested tag's boundaries aligns; and *Toronto Star*'s neglected determiner could be forgiven, certainly within a dependency formulation.

6.3 High-Level Outline of the Approach

Instead of learning the DMV from an unannotated test set, the idea here is to train with text that contains web markup, using various ways of converting HTML into parsing constraints. These constraints come from a blog — a new corpus created for this chapter, the web and news (see Table 6.1 for corpora's sentence and token counts). To facilitate future work, the manually-constructed blog data was made publicly available.[3] Although it is not practical to share larger-scale resources, the main results should be reproducible, as both linguistic analysis and the best model rely exclusively on the blog.

[2]For example, the Stanford parser (circa 2010): `http://nlp.stanford.edu:8080/parser`
[3]`http://cs.stanford.edu/~valentin/`

Corpus	*Sentences*	*POS Tokens*
WSJ$^\infty$	49,208	1,028,347
Section 23	2,353	48,201
WSJ45	48,418	986,830
WSJ15	15,922	163,715
Brown100	24,208	391,796
BLOG$_p$	57,809	1,136,659
BLOG$_t$45	56,191	1,048,404
BLOG$_t$15	23,214	212,872
NEWS45	2,263,563,078	32,119,123,561
NEWS15	1,433,779,438	11,786,164,503
WEB45	8,903,458,234	87,269,385,640
WEB15	7,488,669,239	55,014,582,024

Table 6.1: Sizes of corpora derived from WSJ and Brown and those collected from the web.

6.4 Data Sets for Evaluation and Training

The appeal of unsupervised parsing lies in its ability to learn from surface text alone; but (intrinsic) evaluation still requires parsed sentences. Thus, primary reference sets are still derived from the Penn English Treebank's Wall Street Journal portion — WSJ45 (sentences with fewer than 46 tokens) and Section 23 of WSJ$^\infty$ (all sentence lengths), in addition to Brown100, similarly derived from the parsed portion of the Brown corpus. WSJ$\{15, 45\}$ are also used to train baseline models, but the bulk of the experiments is with web data.

6.4.1 A News-Style Blog: Daniel Pipes

Since there was no corpus overlaying syntactic structure with markup, a new one was constructed by downloading articles[4] from a news-style blog. Although limited to a single genre — political opinion, `danielpipes.org` is clean, consistently formatted, carefully edited and larger than WSJ (see Table 6.1). Spanning decades, Pipes' editorials are mostly in-domain for POS taggers and tree-bank-trained parsers; his recent (internet-era) entries are thoroughly cross-referenced, conveniently providing just the markup one might hope to

[4]`http://danielpipes.org/art/year/all`

Length Cutoff	*Marked Sentences*	*POS Tokens*	*Bracketings All*	*Bracketings Multi-Token*
0	6,047	1,136,659	7,731	6,015
1	*of 57,809*	149,483	7,731	6,015
2	4,934	124,527	6,482	6,015
3	3,295	85,423	4,476	4,212
4	2,103	56,390	2,952	2,789
5	1,402	38,265	1,988	1,874
6	960	27,285	1,365	1,302
7	692	19,894	992	952
8	485	14,528	710	684
9	333	10,484	499	479
10	245	7,887	365	352
15	42	1,519	65	63
20	13	466	20	20
25	6	235	10	10
30	3	136	6	6
40	0	0	0	0

Table 6.2: Counts of sentences, tokens and (unique) bracketings for BLOG$_p$, restricted to only those sentences having at least one bracketing no shorter than the length cutoff (but shorter than the sentence).

study via uncluttered (printer-friendly) HTML.[5]

After extracting moderately clean text and markup locations, MxTerminator [264] was used to detect sentence boundaries. This initial automated pass begot multiple rounds of various semi-automated clean-ups that involved fixing sentence breaking, modifying parser-unfriendly tokens, converting HTML entities and non-ASCII text, correcting typos, and so on. After throwing away annotations of fractional words (e.g., `<i>`*basmachi*`</i>`s) and tokens (e.g., `<i>`*Sesame Street*`</i>`-like), all markup that crossed sentence boundaries was broken up (i.e., loosely speaking, replacing constructs like `<u>`...][s...`</u>` with `<u>`...`</u>`][s `<u>`...`</u>`) and discarding any tags left covering entire sentences.

Two versions of the data were finalized: BLOG$_t$, tagged with the Stanford tagger [321,

[5]`http://danielpipes.org/article_print.php?id=`...

320],[6] and BLOG$_p$, parsed with Charniak's parser [52, 53].[7] The reason for this dichotomy was to use state-of-the-art parses to analyze the relationship between syntax and markup, yet to prevent jointly tagged (and non-standard `AUX[G]`) POS sequences from interfering with the (otherwise unsupervised) training.[8]

6.4.2 Scaled up *Quantity*: The (English) Web

A large (see Table 6.1) but messy data set, WEB, was built — English-looking web-pages, pre-crawled by a search engine. To avoid machine-generated spam, low quality sites flagged by the indexing system were excluded. Only sentence-like runs of words (satisfying punctuation and capitalization constraints), were kept, POS-tagged with TnT [36].

6.4.3 Scaled up *Quality*: (English) Web News

In an effort to trade quantity for quality, a smaller, potentially cleaner data set, NEWS, we also constructed. Editorialized content could lead to fewer extracted non-sentences. Perhaps surprisingly, NEWS is less than an order of magnitude smaller than WEB (see Table 6.1); in part, this is due to less aggressive filtering, because of the trust in sites approved by the human editors at Google News.[9] In all other respects, pre-processing of NEWS pages was identical to the handling of WEB data.

6.5 Linguistic Analysis of Markup

Is there a connection between markup and syntactic structure? Previous work [18] has only examined search engine queries, showing that they consist predominantly of short noun phrases. If web markup shared a similar characteristic, it might not provide sufficiently disambiguating cues to syntactic structure: HTML tags could be too short (e.g., singletons

[6]`http://nlp.stanford.edu/software/stanford-postagger-2008-09-28.tar.gz`
[7]`ftp://ftp.cs.brown.edu/pub/nlparser/parser05Aug16.tar.gz`
[8]However, since many taggers are themselves trained on manually parsed corpora, such as WSJ, no parser that relies on external POS tags could be considered truly unsupervised; for a *fully* unsupervised example, see Seginer's [283] CCL parser, available at `http://www.seggu.net/ccl/`
[9]`http://news.google.com/`

	Count	*POS Sequence*	*Frac*	*Sum*
1	1,242	`NNP NNP`	16.1%	
2	643	`NNP`	8.3	24.4
3	419	`NNP NNP NNP`	5.4	29.8
4	414	`NN`	5.4	35.2
5	201	`JJ NN`	2.6	37.8
6	138	`DT NNP NNP`	1.8	39.5
7	138	`NNS`	1.8	41.3
8	112	`JJ`	1.5	42.8
9	102	`VBD`	1.3	44.1
10	92	`DT NNP NNP NNP`	1.2	45.3
11	85	`JJ NNS`	1.1	46.4
12	79	`NNP NN`	1.0	47.4
13	76	`NN NN`	1.0	48.4
14	61	`VBN`	0.8	49.2
15	60	`NNP NNP NNP NNP`	0.8	50.0
BLOG$_p$	+3,869	*more with Count* ≤ 49	50.0%	

Table 6.3: Top 50% of marked POS tag sequences.

	Count	*Non-Terminal*	*Frac*	*Sum*
1	5,759	`NP`	74.5%	
2	997	`VP`	12.9	87.4
3	524	`S`	6.8	94.2
4	120	`PP`	1.6	95.7
5	72	`ADJP`	0.9	96.7
6	61	`FRAG`	0.8	97.4
7	41	`ADVP`	0.5	98.0
8	39	`SBAR`	0.5	98.5
9	19	`PRN`	0.2	98.7
10	18	`NX`	0.2	99.0
BLOG$_p$	+81	*more with Count* ≤ 16	1.0%	

Table 6.4: Top 99% of dominating non-terminals.

like “click `<a>`here`</a>`”) or otherwise unhelpful in resolving truly difficult ambiguities (such as PP-attachment). Let’s begin simply by counting various basic events in BLOG$_p$.

	Count	Constituent Production	Frac	Sum
1	746	NP → <u>NNP NNP</u>	9.6%	
2	357	NP → <u>NNP</u>	4.6	14.3
3	266	NP → <u>NP PP</u>	3.4	17.7
4	183	NP → <u>NNP NNP NNP</u>	2.4	20.1
5	165	NP → DT <u>NNP NNP</u>	2.1	22.2
6	140	NP → <u>NN</u>	1.8	24.0
7	131	NP → DT <u>NNP NNP NNP</u>	1.7	25.7
8	130	NP → DT <u>NN</u>	1.7	27.4
9	127	NP → <u>DT NNP NNP</u>	1.6	29.0
10	109	S → <u>NP VP</u>	1.4	30.4
11	91	NP → <u>DT NNP NNP NNP</u>	1.2	31.6
12	82	NP → DT <u>JJ NN</u>	1.1	32.7
13	79	NP → <u>NNS</u>	1.0	33.7
14	65	NP → <u>JJ NN</u>	0.8	34.5
15	60	NP → <u>NP NP</u>	0.8	35.3
BLOG$_p$	+5,000	*more with Count* ≤ 60	64.7%	

Table 6.5: Top 15 marked productions, viewed as constituents (bracketings are underlined).

6.5.1 Surface Text Statistics

Out of 57,809 sentences, 6,047 (10.5%) are annotated (see Table 6.2); and 4,934 (8.5%) have multi-token bracketings. Without distinguishing HTML tags, i.e., tracking only unique bracketing end-points within a sentence, 6,015 are multi-token — an average per-sentence yield of 10.4%.[10] As expected, many of the annotated words are nouns, but there are adjectives, verbs and other parts of speech too (see Table 6.3). Markup is short, typically under five words, yet (by far) the most frequently marked sequence of POS tags is a pair.

6.5.2 Common Syntactic Subtrees

For three-quarters of all markup, the lowest dominating non-terminal is a noun phrase (see Table 6.4); there are also non-trace quantities of verb phrases (12.9%) and other phrases, clauses and fragments. Of the top fifteen — *35.2%* of all — annotated productions, only

[10] A non-trivial fraction of the corpus is older (pre-internet) unannotated articles, so this estimate may be conservative.

	Count	*Head-Outward Spawn*	*Frac*	*Sum*
1	1,889	NNP	24.4%	
2	623	NN	8.1	32.5
3	470	DT NNP	6.1	38.6
4	458	DT NN	5.9	44.5
5	345	NNS	4.5	49.0
6	109	NNPS	1.4	50.4
7	98	VBG	1.3	51.6
8	96	NNP NNP NN	1.2	52.9
9	80	VBD	1.0	53.9
10	77	IN	1.0	54.9
11	74	VBN	1.0	55.9
12	73	DT JJ NN	0.9	56.8
13	71	VBZ	0.9	57.7
14	69	POS NNP	0.9	58.6
15	63	JJ	0.8	59.4
BLOG$_p$	+3,136	*more with Count* ≤ 62	40.6%	

Table 6.6: Top 15 marked productions, viewed as dependencies, after recursively expanding any internal nodes that did not align with bracketings (underlined). Tabulated dependencies were collapsed, dropping any dependents that fell entirely in the same region as their parent (i.e., both inside the bracketing, both to its left or both to its right), keeping only crossing attachments.

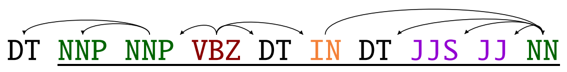
one is *not* a noun phrase (see Table 6.5). Three of the fifteen lowest dominating non-terminals do *not* match the entire bracketing — all three miss the leading determiner, as earlier. In such cases, internal nodes were recursively split until the bracketing aligned, as follows:

[$_{\textbf{S}}$ [$_{\textbf{NP}}$ the <a>*Toronto Star*][$_{\textbf{VP}}$ reports [$_{\textbf{NP}}$ this] [$_{\textbf{PP}}$ in the softest possible way]</a>,[$_{\textbf{S}}$ stating ...]]]

$$\mathtt{S} \rightarrow \mathtt{NP\ VP} \rightarrow \mathtt{DT}\ \underline{\mathtt{NNP\ NNP\ VBZ\ NP\ PP}}\ \mathtt{S}$$

Productions can be summarized more compactly by using a dependency framework and clipping off any dependents whose subtrees do not cross a bracketing boundary, relative to the parent.

Thus,



becomes DT NNP VBZ, “the <a>*Star* reports</a>.” Viewed this way, the top fifteen (now collapsed) productions cover *59.4%* of all cases and include four verb heads, in addition to a preposition and an adjective (see Table 6.6). This exposes five cases of inexact matches, three of which involve neglected determiners or adjectives to the left of the head. In fact, the only case that cannot be explained by dropped dependents is #8, where the daughters are marked but the parent is left out. Most instances contributing to this pattern are flat NPs that end with a noun, incorrectly assumed to be the head of *all* other words in the phrase, e.g.,

... [$_{\mathbf{NP}}$ a 1994 <i>*New Yorker*</i> article] ...

As this example shows, disagreements (as well as agreements) between markup and machine-generated parse trees with automatically percolated heads should be taken with a grain of salt.[11]

6.5.3 Proposed Parsing Constraints

The straight-forward approach — forcing markup to correspond to constituents — agrees with Charniak’s parse trees only **48.0**% of the time, e.g.,

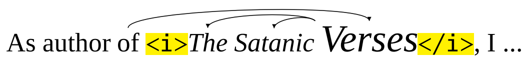
... in [$_{\mathbf{NP}}$<a>[$_{\mathbf{NP}}$ an analysis]</a>[$_{\mathbf{PP}}$ of perhaps the
most astonishing PC item I have yet stumbled upon]].

This number should be higher, as the vast majority of disagreements are due to tree-bank idiosyncrasies (e.g., bare NPs). Earlier examples of incomplete constituents (e.g., legitimately missing determiners) would also be fine in many linguistic theories (e.g., as N-bars). A dependency formulation is less sensitive to such stylistic differences.

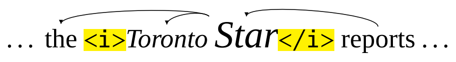
[11]Ravi et al. [262] report that Charniak’s re-ranking parser [53] — `reranking-parserAug06.tar.gz`, also available from `ftp://ftp.cs.brown.edu/pub/nlparser/` — attains 86.3% accuracy when trained on WSJ and tested against Brown; this nearly 5% performance loss out-of-domain is consistent with the numbers originally reported by Gildea [115].

Let's start with the hardest possible constraint on dependencies, then slowly relax it. Every example used to demonstrate a softer constraint doubles as a counter-example against all previous versions.

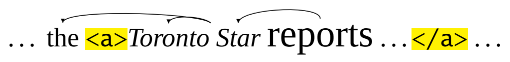
- *strict* — seals markup into attachments, i.e., inside a bracketing, enforces exactly one external arc — into the overall head. This agrees with head-percolated trees just **35.6**% of the time, e.g.,



- *loose* — same as *strict*, but allows the bracketing's head word to have external dependents. This relaxation already agrees with head-percolated dependencies **87.5**% of the time, catching many (though far from all) dropped dependents, e.g.,

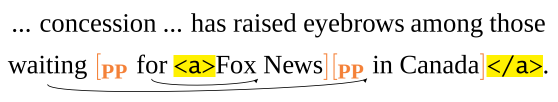


- *sprawl* — same as *loose*, but now allows *all* words inside a bracketing to attach external dependents.[12] This boosts agreement with head-percolated trees to **95.1**%, handling new cases, e.g., where "*Toronto Star*" is embedded in longer markup that includes its own parent — a verb:



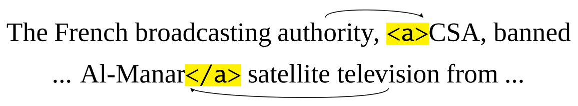
- *tear* — allows markup to fracture after all, requiring only that the external heads attaching the pieces lie to the same side of the bracketing. This propels agreement with percolated dependencies to **98.9**%, fixing previously broken PP-attachment ambiguities, e.g., a fused phrase like "Fox News in Canada" that detached a preposition from its verb:

[12]This view evokes the trapezoids of the $O(n^3)$ recognizer for split head automaton grammars [91].



... concession ... has raised eyebrows among those
waiting [PP for <a>Fox News] [PP in Canada]</a>.

Most of the remaining 1.1% of disagreements are due to parser errors. Nevertheless, it *is* possible for markup to be torn apart by external heads from *both* sides. Below is a (very rare) true negative example: "CSA" modifies "authority" (to its left), appositively, while "Al-Manar" modifies "television" (to its right):[13]



The French broadcasting authority, <a>CSA, banned
... Al-Manar</a> satellite television from ...

6.6 Experimental Methods and Metrics

Viterbi training admits a trivial implementation of most proposed dependency constraints. Six settings parameterized each run:

- `INIT`: `0` — default, uniform initialization; or `1` — a high quality initializer, pre-trained using Ad-Hoc*, with Laplace smoothing, trained at WSJ15 (the "sweet spot" data gradation) but initialized off WSJ8, since that initializer has the best cross-entropy on WSJ15 (see Figure 4.3).

- `GENRE`: `0` — default, baseline training on WSJ; else, uses `1` — BLOG$_t$; `2` — NEWS; or `3` — WEB.

- `SCOPE`: `0` — default, uses all sentences up to length 45; if `1`, trains using sentences up to length 15; if `2`, re-trains on sentences up to length 45, starting from the solution to sentences up to the "sweet spot" length, 15.

- `CONSTR`: if `4`, *strict*; if `3`, *loose*; and if `2`, *sprawl* (evel `1`, *tear*, was not implemented). Over-constrained sentences are re-attempted at successively lower levels until they become possible to parse, if necessary at the lowest (default) level `0`.[14]

[13]A stretch, since the comma after "CSA" renders the marked phrase ungrammatical even *out* of context.
[14]At level `4`, `<b>` **X**`<u>` **Y**`</b>` Z`</u>` is over-constrained.

- `TRIM`: if `1`, discards any sentence without a single multi-token markup (shorter than its length).
- `ADAPT`: if `1`, upon convergence, initializes re-training on WSJ45 using the solution to <`GENRE`>, attempting domain adaptation [183].

These make for 294 meaningful combinations. Each one was judged by its accuracy on WSJ45, using standard directed scoring — the fraction of correct dependencies over randomized "best" parse trees.

6.7 Discussion of Experimental Results

Evaluation on Section 23 of WSJ and Brown reveals that blog-training beats all previously published state-of-the-art numbers in every traditionally-reported length cutoff category, with news-training not far behind. Here is a mini-preview of these results, for Section 23 of WSJ10 and WSJ$^\infty$ (from Table 6.9):

Model			WSJ10	WSJ$^\infty$
DMV	Bilingual Log-Normals (tie-verb-noun)	[66]	62.0	42.2
	Leapfrog	(Ch. 3)	57.1	45.0
	NEWS-best		67.3	*50.1*
	BLOG$_t$-best		**69.3**	**50.4**
EVG	Smoothed (skip-head), Lexicalized	[133]	*68.8*	

Table 6.7: Directed accuracies on Section 23 of WSJ$\{10,^\infty\}$ for previous state-of-the-art systems and the best new runs (as judged against WSJ45) for NEWS and BLOG$_t$ (more details in Table 6.9).

Since the experimental setup involved testing nearly three hundred models simultaneously, extreme care must be taken in analyzing and interpreting these results, to avoid falling prey to any looming "data-snooping" biases, as in the previous chapter. In a sufficiently large pool of models, where each is trained using a randomized and/or chaotic procedure (such as here), the best may look good due to pure chance. An appeal will be made to three separate diagnostics, to conclude that the best results are *not* noise.

The most radical approach would be to write off WSJ as a development set and to focus only on the results from the held-out Brown corpus. It was initially intended as a test of out-of-domain generalization, but since Brown was in no way involved in selecting the best models, it also qualifies as a blind evaluation set. The best models perform even better (and gain more — see Table 6.9) on Brown than on WSJ — a strong indication that the selection process has not overfitted.

The second diagnostic is a closer look at WSJ. Since it would be hard to graph the full (six-dimensional) set of results, a simple linear regression will suffice, using accuracy on WSJ45 as the dependent variable. As in the previous chapter, this full factorial design is preferable to the more traditional ablation studies because it allows one to account for and to incorporate every single experimental data point incurred along the way. Its output is a coarse, high-level summary of our runs, showing which factors significantly contribute to changes in error rate on WSJ45:

Parameter	*(Indicator)*	*Setting*	$\hat{\beta}$	p*-value*
`INIT`	1	ad-hoc @WSJ8,15	11.8	***
`GENRE`	1	BLOG$_t$	-3.7	0.06
	2	NEWS	-5.3	**
	3	WEB	-7.7	***
`SCOPE`	1	@15	-0.5	0.40
	2	@15→45	-0.4	0.53
`CONSTR`	2	*sprawl*	0.9	0.23
	3	*loose*	1.0	0.15
	4	*strict*	1.8	*
`TRIM`	1	drop unmarked	-7.4	***
`ADAPT`	1	WSJ re-training	1.5	**
Intercept		($R^2_{\text{Adjusted}} = 73.6\%$)	39.9	***

Convention: *** for $p < 0.001$; ** for $p < 0.01$ (very significant); and * for $p < 0.05$ (significant).

The default training mode (all parameters zero) is estimated to score 39.9%. A good initializer gives the biggest (double-digit) gain; both domain adaptation and constraints also

Corpus	*Marked Sentences*	*All Sentences*	*POS Tokens*	*All Bracketings*	*Multi-Token Bracketings*
BLOG$_t$45	5,641	56,191	1,048,404	7,021	5,346
BLOG$'_t$45	4,516	4,516	104,267	5,771	5,346
BLOG$_t$15	1,562	23,214	212,872	1,714	1,240
BLOG$'_t$15	1,171	1,171	11,954	1,288	1,240
NEWS45	304,129,910	2,263,563,078	32,119,123,561	611,644,606	477,362,150
NEWS$'$45	205,671,761	205,671,761	2,740,258,972	453,781,081	392,600,070
NEWS15	211,659,549	1,433,779,438	11,786,164,503	365,145,549	274,791,675
NEWS$'$15	147,848,358	147,848,358	1,397,562,474	272,223,918	231,029,921
WEB45	1,577,208,680	8,903,458,234	87,269,385,640	3,309,897,461	2,459,337,571
WEB$'$45	933,115,032	933,115,032	11,552,983,379	2,084,359,555	1,793,238,913
WEB15	1,181,696,194	7,488,669,239	55,014,582,024	2,071,743,595	1,494,675,520
WEB$'$15	681,087,020	681,087,020	5,813,555,341	1,200,980,738	1,072,910,682

Table 6.8: Counts of sentences, tokens and (unique) bracketings for web-based data sets; trimmed versions, restricted to only those sentences having at least one multi-token bracketing, are indicated by a prime ($'$).

make a positive impact. Throwing away unannotated data hurts, as does training out of domain (the blog is least bad; the web is worst). Of course, this overview should not be taken too seriously. Overly simplistic, a first order model ignores interactions between parameters. Furthermore, a least squares fit aims to capture central tendencies, whereas the interesting information is captured by outliers — the best-performing runs.

A major imperfection of the simple regression model is that helpful factors that require an interaction to “kick in” may not, on their own, appear statistically significant. The third diagnostic examines parameter settings that give rise to the best-performing models, looking out for combinations that consistently deliver superior results.

6.7.1 WSJ Baselines

Just two parameters apply to learning from WSJ. Five of their six combinations are state-of-the-art, demonstrating the power of Viterbi training; only the default run scores worse than 45.0%, attained by Leapfrog, on WSJ45:

Settings	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	*41.3*	45.0	45.2
1	46.6	47.5	**47.6**
	@45	@15	@15→45

6.7.2 Blog

Simply training on BLOG$_t$ instead of WSJ hurts:

GENRE=1	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	39.6	36.9	36.9
1	46.5	46.3	46.4
	@45	@15	@15→45

The best runs use a good initializer, discard unannotated sentences, enforce the *loose* constraint on the rest, follow up with domain adaptation and benefit from re-training — GENRE=TRIM=ADAPT=1:

INIT=1	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	45.8	48.3	49.6
(*sprawl*) 2	46.3	49.2	49.2
(*loose*) 3	41.3	*50.2*	**50.4**
(*strict*) 4	40.7	49.9	48.7
	@45	@15	@15→45

The contrast between unconstrained learning and annotation-guided parsing is higher for the default initializer, still using trimmed data sets (just over a thousand sentences for BLOG$'_t$15 — see Table 6.8):

INIT=0	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	25.6	19.4	19.3
(*sprawl*) 2	25.2	22.7	22.5
(*loose*) 3	32.4	26.3	27.3
(*strict*) 4	36.2	38.7	40.1
	@45	@15	@15→45

Above, a clearer benefit to the constraints can be seen.

6.7.3 News

Training on WSJ is also better than using NEWS:

GENRE=2	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	40.2	38.8	38.7
1	43.4	44.0	43.8
	@45	@15	@15→45

As with the blog, the best runs use the good initializer, discard unannotated sentences, enforce the *loose* constraint and follow up with domain adaptation — GENRE=2; INIT=TRIM=ADAPT=1:

Settings	SCOPE=0	SCOPE=1	SCOPE=2
CONSTR=0	46.6	45.4	45.2
(*sprawl*) 2	46.1	44.9	44.9
(*loose*) 3	**49.5**	48.1	48.3
(*strict*) 4	37.7	36.8	37.6
	@45	@15	@15→45

With all the extra training data, the best new score is just 49.5%. On the one hand, the lack of dividends to orders of magnitude more data is disappointing. On the other, the fact that the system arrives within 1% of its best result — 50.4%, obtained with a manually cleaned up corpus — now using an auto-generated data set, is comforting.

6.7.4 Web

The WEB-side story is more discouraging:

GENRE=3	SCOPE=0	SCOPE=1	SCOPE=2
INIT=0	38.3	35.1	35.2
1	42.8	43.6	43.4
	@45	@15	@15→45

The best run again uses a good initializer, keeps *all* sentences, still enforces the *loose* constraint and follows up with domain adaptation, but performs worse than all well-initialized WSJ baselines, scoring only 45.9% (trained at WEB15).

The web seems to be too messy for this chapter's methods. On top of the challenges of language identification and sentence-breaking, there is a lot of boiler-plate; furthermore, web text can be difficult for news-trained POS taggers. For example, the verb "sign" is twice mistagged as a noun and "YouTube" is classified as a verb, in the top four POS sequences of web sentences:[15]

	POS Sequence	WEB *Count*
	Sample web sentence, chosen uniformly at random.	
1	`DT NNS VBN`	82,858,487
	All rights reserved.	
2	`NNP NNP NNP`	65,889,181
	Yuasa et al.	
3	`NN IN TO VB RB`	31,007,783
	Sign in to YouTube now!	
4	`NN IN IN PRP$ JJ NN`	31,007,471
	Sign in with your Google Account!	

6.7.5 The State of the Art

The best model gains more than 5% over previously published state-of-the-art accuracy across all sentences of WSJ's Section 23, more than 8% on WSJ20 and rivals the oracle skyline (70.2% — see Figure 3.2a) on WSJ10; these gains generalize to Brown100, where it improves by nearly 10% (see Table 6.9). The best models agree in using *loose* constraints. Of these, the models trained with less data perform better, with the best two using trimmed data sets, echoing that "less is more," pace Halevy et al. [130]. Orders of magnitude more data did not improve parsing performance further, though a different outcome might be expected from lexicalized models: The primary benefit of additional lower-quality data is

[15]Further evidence: TnT tags the ubiquitous but ambiguous fragments "click here" and "print post" as noun phrases.

Model	*Incarnation*		WSJ10	WSJ20	WSJ∞	Brown100
DMV	Bilingual Log-Normals (tie-verb-noun)	[66]	62.0	48.0	42.2	
	Leapfrog	(Ch. 3)	57.1	48.7	45.0	43.6
	default `INIT=0,GENRE=0,SCOPE=0,CONSTR=0,TRIM=0,ADAPT=0`		55.9	45.8	41.6	40.5
	WSJ-best `INIT=1,GENRE=0,SCOPE=2,CONSTR=0,TRIM=0,ADAPT=0`		65.3	53.8	47.9	50.8
	BLOG$_t$-best `INIT=1,GENRE=1,SCOPE=2,CONSTR=3,TRIM=1,ADAPT=1`		**69.3**	**56.8**	**50.4**	**53.3**
	NEWS-best `INIT=1,GENRE=2,SCOPE=0,CONSTR=3,TRIM=1,ADAPT=1`		67.3	*56.2*	*50.1*	*51.6*
	WEB-best `INIT=1,GENRE=3,SCOPE=1,CONSTR=3,TRIM=0,ADAPT=1`		64.1	52.7	46.3	46.9
EVG	Smoothed (skip-head), Lexicalized	[133]	*68.8*			

Table 6.9: Accuracies on Section 23 of WSJ$\{10, 20, ^\infty\}$ and Brown100 for three recent state-of-the-art systems, our default run, and our best runs (judged by accuracy on WSJ45) for each of four training sets.

in improved coverage. But with only 35 unique POS tags, data sparsity is hardly an issue. Extra examples of lexical items help little and hurt when they are mistagged.

6.8 Related Work

The wealth of new annotations produced in many languages every day already fuels a number of NLP applications. Following their early and wide-spread use by search engines, in service of spam-fighting and retrieval, anchor text and link data enhanced a variety of traditional NLP techniques: crosslingual information retrieval [233], translation [196], both named-entity recognition [220] and categorization [335], query segmentation [318], plus semantic relatedness and word-sense disambiguation [109, 342]. Yet several, seemingly natural, candidate core NLP tasks — tokenization, CJK segmentation, noun-phrase chunking, and (until now) parsing — remained conspicuously uninvolved.[16]

Approaches related to ones covered by this chapter arise in applications that combine parsing with named-entity recognition (NER). For example, constraining a parser to respect the boundaries of known entities is standard practice not only in joint modeling of

[16]Following the work in this chapter, this omission has been partially rectified for Chinese [316, 146, 151, 346], as well as in the form of a linguistic inquiry into the constituency of hyperlinks [99].

(constituent) parsing and NER [103], but also in higher-level NLP tasks, such as relation extraction [221], that couple chunking with (dependency) parsing. Although restricted to proper noun phrases, dates, times and quantities, we suspect that constituents identified by trained (supervised) NER systems would also be helpful in constraining grammar induction.

Following Pereira and Schabes' [245] success with partial annotations in training a model of (English) constituents generatively, their idea has been extended to discriminative estimation [265] and also proved useful in modeling (Japanese) dependencies [278]. There was demand for partially bracketed corpora. Chen and Lee [57] constructed one such corpus by learning to partition (English) POS sequences into chunks [2]; Inui and Kotani [147] used n-gram statistics to split (Japanese) clauses.[17] This chapter combined the two intuitions, using the web to build a partially parsed corpus. Such an approach could be called *lightly* supervised, since it does not require manual annotation of a single complete parse tree. In contrast, traditional semi-supervised methods rely on fully-annotated seed corpora.[18]

6.9 Conclusion

This chapter explored novel ways of training dependency parsing models. The linguistic analysis of a blog reveals that web annotations can be converted into accurate parsing constraints (*loose*: 88%; *sprawl*: 95%; *tear*: 99%) that could also be helpful to supervised methods, e.g., by boosting an initial parser via self-training [208] on sentences with markup. Similar techniques may apply to standard word-processing annotations, such as font changes, and to certain (balanced) punctuation [39].

The blog data set, overlaying markup and syntax, has been made publicly available. Its annotations are 75% noun phrases, 13% verb phrases, 7% simple declarative clauses and 2% prepositional phrases, with traces of other phrases, clauses and fragments. The type

[17]Earlier, Magerman and Marcus [198] used mutual information, rather than a grammar, to recognize phrase-structure. But simple entropy-minimizing techniques tend to clash with human notions of syntax [82]. A classic example is "edby" — a common English character sequence (as in "caus*ed by*" or "walk*ed by*") proposed as a word by Olivier's [242] segmenter.

[18]A significant effort expended in building a tree-bank comes with the first batch of sentences [87].

of markup, combined with POS tags, could make for valuable features in discriminative models of parsing [260].

A logical next step would be to explore the connection between syntax and markup for genres other than a news-style blog and for languages other than English. If the strength of the connection between web markup and syntactic structure is universal across languages and genres, this fact could have broad implications for NLP, with applications extending well beyond parsing.

Chapter 7

Punctuation

The purpose of this chapter is to explore whether constraints developed for English web markup might also be generally useful for punctuation, which is a traditional signal for text boundaries in many languages. Supporting peer-reviewed publication is *Punctuation: Making a Point in Unsupervised Dependency Parsing* in CoNLL 2011 [304].

7.1 Introduction

Uncovering hidden relations between head words and their dependents in free-form text poses a challenge in part because sentence structure is underdetermined by only raw, unannotated words. Structure can be clearer in *formatted* text, which typically includes proper capitalization and punctuation [129]. Raw word streams, such as utterances transcribed by speech recognizers, are often difficult even for humans [167]. Therefore, one would expect grammar inducers to exploit any available linguistic meta-data (e.g., HTML, which is ordinarily stripped out during pre-processing). And yet in unsupervised dependency parsing, sentence-internal punctuation has long been ignored [44, 244, 172, 33, *inter alia*].

This chapter proposes exploring punctuation's potential to aid grammar induction. Consider a motivating example (all of this chapter's examples are from WSJ), in which all (six) marks align with constituent boundaries:

[$_{\textbf{SBAR}}$ Although it probably has reduced the level of expenditures for some purchasers], [$_{\textbf{NP}}$ utilization management] — [$_{\textbf{PP}}$ like most other cost containment strategies] — [$_{\textbf{VP}}$ doesn't appear to have altered the long-term rate of increase in health-care costs], [$_{NP}$ the Institute of Medicine], [$_{NP}$ an affiliate of the National Academy of Sciences], [$_{VP}$ concluded after a two-year study].

This link between punctuation and constituent boundaries suggests that parsing could be approximated by treating inter-punctuation fragments independently. In training, an algorithm could first parse each fragment separately, then parse the sequence of the resulting head words. In inference, a better approximation could be used to allow heads of fragments to be attached by arbitrary external words, e.g.:

The Soviets complicated the issue by offering to [$_{VP}$ include light tanks], [$_{SBAR}$ which are as light as ...].

	Count	*POS Sequence*	*Frac*	*Cum*
1	3,492	NNP	2.8%	
2	2,716	CD CD	2.2	5.0
3	2,519	NNP NNP	2.0	7.1
4	2,512	RB	2.0	9.1
5	1,495	CD	1.2	10.3
6	1,025	NN	0.8	11.1
7	1,023	NNP NNP NNP	0.8	11.9
8	916	IN NN	0.7	12.7
9	795	VBZ NNP NNP	0.6	13.3
10	748	CC	0.6	13.9
11	730	CD DT NN	0.6	14.5
12	705	PRP VBD	0.6	15.1
13	652	JJ NN	0.5	15.6
14	648	DT NN	0.5	16.1
15	627	IN DT NN	0.5	16.6
WSJ	+103,148	*more with Count* ≤ 621	83.4%	

Table 7.1: Top 15 fragments of POS tag sequences in WSJ.

7.2 Definitions, Analyses and Constraints

Punctuation and syntax are related [240, 39, 157, 85, *inter alia*]. But are there simple enough connections between the two to aid in grammar induction? This section explores the regularities. This chapter's study of punctuation in WSJ parallels the previous chapter's

	Count	*Non-Terminal*	*Frac*	*Cum*
1	40,223	S	32.5%	
2	33,607	NP	27.2	59.7
3	16,413	VP	13.3	72.9
4	12,441	PP	10.1	83.0
5	8,350	SBAR	6.7	89.7
6	4,085	ADVP	3.3	93.0
7	3,080	QP	2.5	95.5
8	2,480	SINV	2.0	97.5
9	1,257	ADJP	1.0	98.5
10	369	PRN	0.3	98.8
WSJ	+1,446	*more with Count* ≤ 356	1.2%	

Table 7.2: Top 99% of the lowest dominating non-terminals deriving complete inter-punctuation fragments in WSJ.

analysis of markup from a web-log, since the proposed constraints turn out to be useful. Throughout, an inter-punctuation ***fragment*** is defined as a maximal (non-empty) consecutive sequence of words that does not cross punctuation boundaries and is shorter than its source sentence.

7.2.1 A Linguistic Analysis

Out of 51,558 sentences, most — 37,076 (71.9%) — contain sentence-internal punctuation. These punctuated sentences contain 123,751 fragments, nearly all — 111,774 (90.3%) — of them multi-token. Common POS sequences comprising fragments are diverse (note also their flat distribution — see Table 7.1). The plurality of fragments are dominated by a clause, but most are dominated by one of several kinds of phrases (see Table 7.2). As expected, punctuation does not occur at all constituent boundaries: Of the top 15 productions that yield fragments, five do *not* match the exact bracketing of their lowest dominating non-terminal (see ranks 6, 11, 12, 14 and 15 in Table 7.3). Four of them miss a left-adjacent clause, e.g., S $\to$ S $\underline{\text{NP VP}}$:

[$_{S}$ [$_{S}$ It's an overwhelming job], [$_{NP}$ she] [$_{VP}$ says.]]

This production is flagged because the fragment $\underline{\text{NP VP}}$ is not *a* constituent — it is two;

	Count	*Constituent Production*	*Frac*	*Cum*
1	7,115	PP → IN NP	5.7%	
2	5,950	S → NP VP	4.8	10.6
3	3,450	NP → NP PP	2.8	13.3
4	2,799	SBAR → WHNP S	2.3	15.6
5	2,695	NP → NNP	2.2	17.8
6	2,615	S → S NP VP	2.1	19.9
7	2,480	SBAR → IN S	2.0	21.9
8	2,392	NP → NNP NNP	1.9	23.8
9	2,354	ADVP → RB	1.9	25.7
10	2,334	QP → CD CD	1.9	27.6
11	2,213	S → PP NP VP	1.8	29.4
12	1,441	S → S CC S	1.2	30.6
13	1,317	NP → NP NP	1.1	31.6
14	1,314	S → SBAR NP VP	1.1	32.7
15	1,172	SINV → S VP NP NP	0.9	33.6
WSJ	+82,110	*more with Count* ≤ 976	66.4%	

Table 7.3: Top 15 productions yielding punctuation-induced fragments in WSJ, viewed as constituents, after recursively expanding any internal nodes that do not align with the associated fragmentation (underlined).

still, **49.4**% of all fragments do align with whole constituents.

Inter-punctuation fragments correspond more strongly to dependencies (see Table 7.4). Only one production (rank 14) shows a daughter outside her mother's fragment. Some number of such productions is inevitable and expected, since fragments must coalesce (i.e., the root of at least one fragment — in every sentence with sentence-internal punctuation — must be attached by some word from a different, external fragment). It is noteworthy that in 14 of the 15 most common cases, a word in an inter-punctuation fragment derives precisely the rest of that fragment, attaching none of the other, external words. This is true for **39.2**% of all fragments, and if fragments whose heads attach other fragments' heads are also included, agreement increases to **74.0**% (see *strict* and *loose* constraints, next).

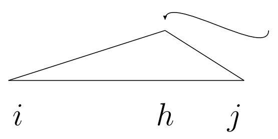
	Count	*Head-Outward Spawn*	*Frac*	*Cum*
1	11,928	IN	9.6%	
2	8,852	NN	7.2	16.8
3	7,802	NNP	6.3	23.1
4	4,750	CD	3.8	26.9
5	3,914	VBD	3.2	30.1
6	3,672	VBZ	3.0	33.1
7	3,436	RB	2.8	35.8
8	2,691	VBG	2.2	38.0
9	2,304	VBP	1.9	39.9
10	2,251	NNS	1.8	41.7
11	1,955	WDT	1.6	43.3
12	1,409	MD	1.1	44.4
13	1,377	VBN	1.1	45.5
14	1,204	IN $\curvearrowleft$ VBD	1.0	46.5
15	927	JJ	0.7	47.3
WSJ	+65,279	*more with Count* ≤ 846	52.8%	

Table 7.4: Top 15 productions yielding punctuation-induced fragments in WSJ, viewed as dependencies, after dropping all daughters that fell entirely in the same region as their mother (i.e., both inside a fragment, both to its left or both to its right), keeping only crossing attachments (just one).

7.2.2 Five Parsing Constraints

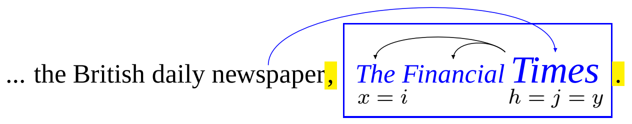
The previous chapter showed how to express similar correspondences with markup as parsing constraints, proposing four but employing only the strictest three constraints, and omitting implementation details. This chapter revisits those constraints, specifying precise logical formulations used in the code, and introduces a fifth (most relaxed) constraint.

Let $[x, y]$ be a fragment (or markup) spanning positions x through y (inclusive, with $1 \leq x < y \leq l$), in a sentence of length l. And let $[i, j]_h$ be a sealed span headed by h ($1 \leq i \leq h \leq j \leq l$), i.e., the word at position h dominates precisely $i \ldots j$ (but none other):

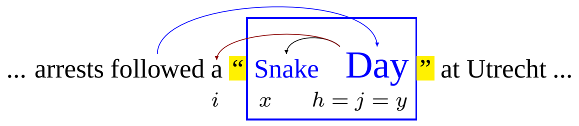


Define $inside(h, x, y)$ as true iff $x \leq h \leq y$; and also let $cross(i, j, x, y)$ be true iff $(i < x \;\wedge\; j \geq x \;\wedge\; j < y) \vee (i > x \;\wedge\; i \leq y \;\wedge\; j > y)$. Then the three tightest constraints impose conditions which, when satisfied, disallow sealing $[i, j]_h$ in the presence of an annotation $[x, y]$:

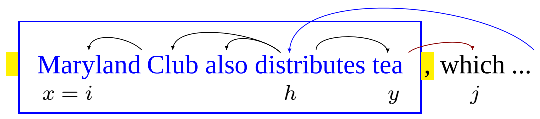
- ***strict*** — requires $[x, y]$ itself to be sealed in the parse tree, voiding all seals that straddle exactly one of $\{x, y\}$ or protrude beyond $[x, y]$ if their head is inside. This constraint holds for **39.2**% of fragments. By contrast, only 35.6% of HTML annotations, such as anchor texts and italics, agree with it. This necessarily fails in every sentence with internal punctuation (since there, *some* fragment must take charge and attach another), when $cross(i, j, x, y) \;\vee\; (inside(h, x, y) \wedge (i < x \;\vee\; j > y))$.



- ***loose*** — if $h \in [x, y]$, requires that everything in $x \ldots y$ fall under h, with only h allowed external attachments. This holds for **74.0**% of fragments — 87.5% of markup, failing when $cross(i, j, x, y)$.

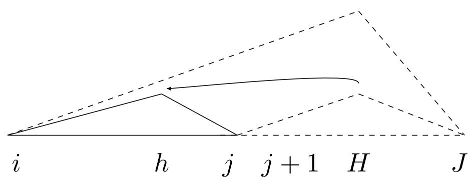


- ***sprawl*** — still requires that h derive $x \ldots y$ but lifts restrictions on external attachments. Holding for **92.9**% of fragments (95.1% of markup), this constraint fails when $cross(i, j, x, y) \;\wedge\; \neg inside(h, x, y)$.

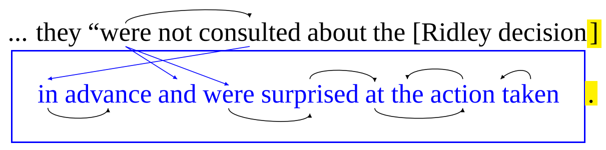


These three strictest constraints lend themselves to a straight-forward implementation as an $O(l^5)$ chart-based decoder. Ordinarily, the probability of $[i, j]_h$ is computed by multiplying the probability of the associated *un*sealed span by two stopping probabilities — that of the word at h on the left (adjacent if $i = h$; non-adjacent if $i < h$) and on the right (adjacent if $h = j$; non-adjacent if $h < j$). To impose a constraint, one could run through all of the annotations $[x, y]$ associated with a sentence and zero out this probability if any of them satisfy disallowed conditions. There are faster — e.g., $O(l^4)$, and even $O(l^3)$ — recognizers for split head automaton grammars [91]. Perhaps a more practical, but still clear, approach would be to generate n-best lists using a more efficient unconstrained algorithm, then apply the constraints as a post-filtering step.

Relaxed constraints disallow joining adjacent subtrees, e.g., preventing the seal $[i, j]_h$ from merging below the *un*sealed span $[j + 1, J]_H$, on the left:

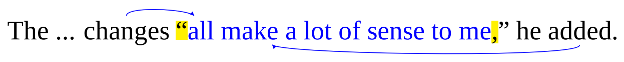


- ***tear*** — prevents $x \ldots y$ from being torn apart by external heads from *opposite* sides. This constraint holds for **94.7**% of fragments (97.9% of markup), and is violated when $(x \leq j \ \wedge \ y > j \ \wedge \ h < x)$, in this case.



... they “were not consulted about the [Ridley decision] in advance and were surprised at the action taken .

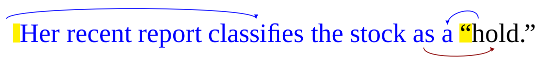
- ***thread*** — requires only that no path from the root to a leaf enter $[x, y]$ twice. This constraint holds for **95.0**% of all fragments (98.5% of markup); it is violated when $(x \leq j \ \wedge \ y > j \ \wedge \ h < x) \ \wedge \ (H \leq y)$, again, in this case. Example that satisfies *thread* but violates *tear*:

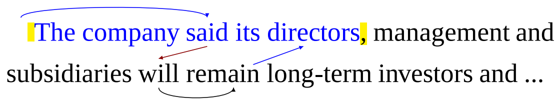


The ... changes “all make a lot of sense to me,” he added.

The case when $[i, j]_h$ is to the right is entirely symmetric, and these constraints could be incorporated in a more sophisticated decoder (since i and J do not appear in the formulae, above). They could be implemented by zeroing out the probability of the word at H attaching that at h (to its left), in case of a violation.

Note that all five constraints are nested. In particular, this means that it does not make sense to combine them, for a given annotation $[x, y]$, since the result would just match the strictest one. The markup number for *tear* in this chapter is lower (97.9 versus 98.9%), compared to the previous one, because that chapter allowed cases where markup was *neither* torn nor threaded. Common structures that violate *thread* (and, consequently, all five of the constraints) include, e.g., "seamless" quotations and even ordinary lists:




The company said its directors, management and
subsidiaries will remain long-term investors and ...

7.2.3 Comparison with Markup

Most punctuation-induced constraints are less accurate than the corresponding markup-induced constraints (e.g., *sprawl*: 92.9 vs. 95.1%; *loose*: 74.0 vs. 87.5%; but not *strict*: 39.2 vs. 35.6%). However, markup is rare: only 10% of the sentences in the blog were annotated; in contrast, over 70% of the sentences in WSJ are fragmented by punctuation.

Fragments are more than 40% likely to be dominated by a clause; for markup, this number is below 10% — nearly 75% of it covered by noun phrases. Further, inter-punctuation fragments are spread more evenly under noun, verb, prepositional, adverbial and adjectival phrases (approximately 27:13:10:3:1 versus 75:13:2:1:1) than markup.[1]

[1]Markup and fragments are as likely to be in verb phrases.

7.3 Methods

The DMV ordinarily strips out punctuation. Since this step already requires identification of marks, the techniques in this chapter are just as “unsupervised.”

7.3.1 A Basic System

The system in this chapter is based on Laplace-smoothed Viterbi EM, using a two-stage scaffolding: the first stage trains with just the sentences up to length 15; the second stage then retrains on nearly all sentences — those with up to 45 words.

Initialization

Since the “ad-hoc harmonic” initializer does not work very well for longer sentences, particularly with Viterbi training (see Figure 4.2), this chapter employs an improved initializer that approximates the attachment probability between two words as an average, over all sentences, of their normalized aggregate *weighted* distances. The weighting function is

$$w(d) = 1 + \lg^{-1}(1 + d);$$

the integer $d \geq 1$ is a distance between two tokens; (and $\lg^{-1}$ is $1/\log_2$).

Termination

Since smoothing can (and does, at times) increase the objective, it is more efficient to terminate early. In this chapter, optimization is stopped after ten steps of suboptimal models, using the lowest-perplexity (not necessarily the last) model found, as measured by the cross-entropy of the training data.

Constrained Training

Training with punctuation replaces ordinary Viterbi parse trees, at every iteration of EM, with the output of a constrained decoder. All experiments other than #2 (§7.5) train with the *loose* constraint. Previous chapter found this setting to be best for markup-induced constraints; this chapter applies it to constraints induced by inter-punctuation fragments.

Constrained Inference

Previous chapter suggested using the *sprawl* constraint in inference. Once again, we follow its suggestion in all experiments except #2 (§7.5).

7.3.2 Forgiving Scoring

One of the baseline systems (below) produces dependency trees containing punctuation. In this case the heads assigned to punctuation were not scored, using *forgiving scoring* for regular words: crediting correct heads separated from their children by punctuation alone (from the point of view of the child, looking up to the nearest non-punctuation ancestor).

7.3.3 Baseline Systems

This chapter's primary baseline is the basic system without constraints (*standard training*). It ignores punctuation, as is standard, scoring 52.0% against WSJ45.

A secondary (*punctuation as words*) baseline incorporates punctuation into the grammar as if it were words, as in *supervised* dependency parsing [237, 191, 290, *inter alia*]. It is worse, scoring only 41.0%.[2,3]

7.4 Experiment #1: Default Constraints

The first experiment compares "punctuation as constraints" to the baseline systems, using the default settings: *loose* in training; and *sprawl* in inference. Both constrained regimes

[2]Exactly the same data sets were used in both cases, not counting punctuation towards sentence lengths.

[3]To get this particular number punctuation was forced to be tacked on, as a layer below the tree of words, to fairly compare systems (using the same initializer). Since improved initialization strategies — both *weighted* and the "ad-hoc harmonic" — rely on distances between tokens, they could be unfairly biased towards one approach or the other, if punctuation counted towards length. Similar baselines were also trained without restrictions, allowing punctuation to appear anywhere in the tree (still with *forgiving scoring*), using the uninformed uniform initializer. Disallowing punctuation as a parent of a real word made things worse, suggesting that not all marks belong near the leaves (sentence stops, semicolons, colons, etc. make more sense as roots and heads). The weighted initializer was also tried without restrictions, and all experiments were repeated without scaffolding, on WSJ15 and WSJ45 alone, but treating punctuation as words never came within even 5% of (comparable) standard training. Punctuation, as words, reliably disrupted learning.

	WSJ$^\infty$	WSJ10
Supervised DMV	69.8	83.6
w/Constrained Inference	73.0	84.3
Punctuation as Words	41.7	54.8
Standard Training	52.0	63.2
w/Constrained Inference	54.0	63.6
Constrained Training	55.6	67.0
w/Constrained Inference	**57.4**	**67.5**

Table 7.5: Directed accuracies on Section 23 of WSJ$^\infty$ and WSJ10 for the supervised DMV, several baseline systems and the punctuation runs (all using the weighted initializer).

improve performance (see Table 7.5). Constrained decoding alone increases the accuracy of a standardly-trained system from 52.0% to 54.0%. And constrained training yields 55.6% — 57.4% in combination with inference. These are multi-point increases, but they could disappear in a more accurate state-of-the-art system. To test this hypothesis, constrained decoding was also applied to a *supervised* system. This (ideal) instantiation of the DMV benefits as much or more than the unsupervised systems: accuracy increases from 69.8% to 73.0%. Punctuation seems to capture the kinds of, perhaps long-distance, regularities that are not accessible to the model, possibly due to its unrealistic independence assumptions.

7.5 Experiment #2: Optimal Settings

The recommendation to train with *loose* and decode with *sprawl* came from the previous chapter's experiments with markup. But are these the right settings for punctuation? Inter-punctuation fragments are quite different from markup — they are more prevalent but less accurate. Furthermore, a new constraint was introduced in this chapter, *thread*, that had not been considered before (along with *tear*).

Next the choices of constraints are re-examined. The full factorial analysis was similar, but significantly smaller, than in the previous chapter: it excluded the larger-scale news and web data sets that are not publicly available. Nevertheless, every meaningful combination of settings was tried, testing both *thread* and *tear* (instead of *strict*, since it can't work with sentences containing sentence-internal punctuation), in both training and inference. Better settings than *loose* for training, and *sprawl* for decoding, were not among the options.

A full analysis is omitted. But the first, high-level observation is that constrained inference, using punctuation, is helpful and robust. It boosted accuracy (on WSJ45) by approximately 1.5%, on average, with all settings. Indeed, *sprawl* was consistently (but only slightly, at 1.6%, on average) better than the rest. Second, constrained training hurt more often than it helped. It degraded accuracy in all but one case, *loose*, where it gained approximately 0.4%, on average. Both improvements are statistically significant: $p \approx 0.036$ for training with *loose*; and $p \approx 5.6 \times 10^{-12}$ for decoding with *sprawl*.

7.6 More Advanced Methods

So far, punctuation has improved grammar induction in a toy setting. But would it help a modern system? The next two experiments employ a slightly more complicated set-up, compared with the one used up until now (§7.3.1). The key difference is that this system is lexicalized, as is standard among the more accurate grammar inducers [33, 117, 133].

Lexicalization

Only in the second (full data) stage is lexicalized, using the method of Headden et al. [133]: for words seen at least 100 times in the training corpus, the gold POS tag is augmented with the lexical item. The first (data poor) stage remains entirely unlexicalized, with gold POS tags for word classes, as in the earlier systems.

Smoothing

Smoothing is not used in the second stage except at the end, for the final lexicalized model. Stage one still applies “add-one” smoothing at every iteration.

7.7 Experiment #3: State-of-the-Art

The purpose of these experiments is to compare the punctuation-enhanced DMV with other, more recent state-of-the-art systems. Lexicalized (§7.6), this chapter’s approach performs

	Brown	WSJ$^\infty$	WSJ10
L-EVG [133]	—	—	68.8
Web Markup (Ch. 6)	53.3	50.4	69.3
Posterior Sparsity [117]	—	53.3	64.3
Tree Substitution Grammars [33]	—	55.7	67.7
Constrained Training	58.4	58.0	69.3
w/Constrained Inference	**59.5**	**58.4**	**69.5**

Table 7.6: Accuracies on the out-of-domain Brown100 set and Section 23 of WSJ$^\infty$ and WSJ10, for the lexicalized punctuation run and other, more recent state-of-the-art systems.

better, by a wide margin; without lexicalization (§7.3.1), it was already better for longer, but not for shorter, sentences (see Tables 7.6 and 7.5).

7.8 Experiment #4: Multilingual Testing

This final batch of experiments probes the generalization of this chapter's approach (§7.6) across languages.[4] The gains are *not* English-specific (see Table 7.7). Every language improves with constrained decoding (more so without constrained training); and all but Italian benefit in combination. Averaged across all eighteen languages, the net change in accuracy is 1.3%. After standard training, constrained decoding alone delivers a 0.7% gain, on average, never causing harm in any of our experiments. These gains are statistically significant: $p \approx 1.59 \times 10^{-5}$ for constrained training; and $p \approx 4.27 \times 10^{-7}$ for inference.

A synergy between the two improvements was not detected. However, it is noteworthy that without constrained training, "full" data sets do not help, on average, despite having more data and lexicalization. Furthermore, *after* constrained training, no evidence of benefits to additional retraining was detected: not with the relaxed *sprawl* constraint, nor unconstrained.

[4]Note that punctuation, which was identified by the CoNLL task organizers, was treated differently in the two years: in 2006, it was always at the leaves of the dependency trees; in 2007, it matched original annotations of the source treebanks. For both, punctuation-insensitive scoring was used (§7.3.2).

	Unlexicalized, Unpunctuated				*Lexicalized*		*... **and** Punctuated*		
CoNLL Year	*Initialization @15*		*Training @15*		*Retraining @45*		*Retraining @45*		*Net*
& Language	*1.*	*w/Inference*	*2.*	*w/Inference*	*3.*	*w/Inference*	*3′.*	*w/Inference*	*Gain*
Arabic 2006	23.3	23.6 (+0.3)	32.8	33.1 (+0.4)	31.5	31.6 (+0.1)	32.1	32.6 (+0.5)	+1.1
’7	25.6	26.4 (+0.8)	33.7	34.2 (+0.5)	32.7	33.6 (+0.9)	34.9	35.3 (+0.4)	+2.6
Basque ’7	19.3	20.8 (+1.5)	29.9	30.9 (+1.0)	29.3	30.1 (+0.8)	29.3	29.9 (+0.6)	+0.6
Bulgarian ’6	23.7	24.7 (+1.0)	39.3	40.7 (+1.4)	38.8	39.9 (+1.1)	39.9	40.5 (+0.6)	+1.6
Catalan ’7	33.2	34.1 (+0.8)	54.8	55.5 (+0.7)	54.3	55.1 (+0.8)	54.3	55.2 (+0.9)	+0.9
Czech ’6	18.6	19.6 (+1.0)	34.6	35.8 (+1.2)	34.8	35.7 (+0.9)	37.0	37.8 (+0.8)	+3.0
’7	17.6	18.4 (+0.8)	33.5	35.4 (+1.9)	33.4	34.4 (+1.0)	35.2	36.2 (+1.0)	+2.7
Danish ’6	22.9	24.0 (+1.1)	35.6	36.7 (+1.2)	36.9	37.8 (+0.9)	36.5	37.1 (+0.6)	+0.2
Dutch ’6	15.8	16.5 (+0.7)	*11.2*	12.5 (+1.3)	11.0	11.9 (+1.0)	13.7	14.0 (+0.3)	+3.0
English ’7	25.0	25.4 (+0.5)	47.2	49.5 (+2.3)	47.5	48.8 (+1.3)	49.3	50.3 (+0.9)	+2.8
German ’6	19.2	19.6 (+0.4)	27.4	28.0 (+0.7)	27.0	27.8 (+0.8)	28.2	28.6 (+0.4)	+1.6
Greek ’7	18.5	18.8 (+0.3)	20.7	21.4 (+0.7)	20.5	21.0 (+0.5)	20.9	21.2 (+0.3)	+0.7
Hungarian ’7	17.4	17.7 (+0.3)	*6.7*	7.2 (+0.5)	6.6	7.0 (+0.4)	7.8	8.0 (+0.2)	+1.4
Italian ’7	25.0	26.3 (+1.2)	29.6	29.9 (+0.3)	29.7	29.7 (+0.1)	28.3	28.8 (+0.5)	*-0.8*
Japanese ’6	30.0	30.0 (+0.0)	*27.3*	27.3 (+0.0)	27.4	27.4 (+0.0)	27.5	27.5 (+0.0)	+0.1
Portuguese ’6	27.3	27.5 (+0.2)	32.8	33.7 (+0.9)	32.7	33.4 (+0.7)	33.3	33.5 (+0.3)	+0.8
Slovenian ’6	21.8	21.9 (+0.2)	28.3	30.4 (+2.1)	28.4	30.4 (+2.0)	29.8	31.2 (+1.4)	+2.8
Spanish ’6	25.3	26.2 (+0.9)	31.7	32.4 (+0.7)	31.6	32.3 (+0.8)	31.9	32.3 (+0.5)	+0.8
Swedish ’6	31.0	31.5 (+0.6)	44.1	45.2 (+1.1)	45.6	46.1 (+0.5)	46.1	46.4 (+0.3)	+0.8
Turkish ’6	22.3	22.9 (+0.6)	39.1	39.5 (+0.4)	39.9	39.9 (+0.1)	40.6	40.9 (+0.3)	+1.0
’7	22.7	23.3 (+0.6)	41.7	42.3 (+0.6)	41.9	42.1 (+0.2)	41.6	42.0 (+0.4)	+0.1
Average:	23.4	24.0 (**+0.7**)	31.9	32.9 (**+1.0**)	31.9	32.6 (**+0.7**)	32.6	33.2 (**+0.5**)	**+1.3**

Table 7.7: Multilingual evaluation for CoNLL sets, measured at all three stages of training, with and without constraints.

7.9 Related Work

Punctuation has been used to improve parsing since rule-based systems [157]. Statistical parsers reap dramatic gains from punctuation [98, 266, 51, 154, 69, *inter alia*]. And it is even known to help in *unsupervised* constituent parsing [283]. But for *dependency* grammar induction, prior to the research described in this chapter, punctuation remained unexploited.

Parsing Techniques Most-Similar to Constraints

A “divide-and-rule” strategy that relies on punctuation has been used in supervised constituent parsing of long Chinese sentences [187]. For English, there has been interest in *balanced* punctuation [39], more recently using rule-based filters [336] in a combinatory categorial grammar (CCG). This chapter’s focus was specifically on *unsupervised* learning of *dependency* grammars and is similar, in spirit, to Eisner and Smith’s [92] “vine grammar” formalism. An important difference is that instead of imposing static limits on allowed dependency lengths, the restrictions are dynamic — they disallow some long (and some short) arcs that would have otherwise crossed nearby punctuation.

Incorporating partial bracketings into grammar induction is an idea tracing back to Pereira and Schabes [245]. It inspired the previous chapter: mining parsing constraints from the web. In that same vein, this chapter prospected a more abundant and natural language-resource — punctuation, using constraint-based techniques developed for web markup.

Modern Unsupervised Dependency Parsing

State-of-the-art in unsupervised dependency parsing [33] uses tree substitution grammars. These are powerful models, capable of learning large dependency fragments. To help prevent overfitting, a non-parametric Bayesian prior, defined by a hierarchical Pitman-Yor process [252], is trusted to nudge training towards fewer and smaller grammatical productions. This chapter pursued a complementary strategy: using the much simpler DMV, but persistently steering training away from certain constructions, as guided by punctuation, to help prevent *underfitting*.

Various Other Uses of Punctuation in NLP

Punctuation is hard to predict,[5] partly because it can signal long-range dependencies [195]. It often provides valuable cues to NLP tasks such as part-of-speech tagging and named-entity recognition [136], information extraction [100] and machine translation [185, 206]. Other applications have included Japanese sentence analysis [241], genre detection [313], bilingual sentence alignment [343], semantic role labeling [255], Chinese creation-title recognition [56] and word segmentation [188], plus, more recently, automatic vandalism detection in Wikipedia [333].

[5]Punctuation has high semantic entropy [216]; for an analysis of the many roles played in the WSJ by the comma — the most frequent and unpredictable punctuation mark in that data set — see Beeferman et al. [20, Table 2].

7.10 Conclusions and Future Work

Punctuation improves dependency grammar induction. Many unsupervised (and supervised) parsers could be easily modified to use *sprawl*-constrained decoding in inference. It applies to pre-trained models and, so far, helped every data set and language.

Tightly interwoven into the fabric of writing systems, punctuation frames most unannotated plain-text. This chapter showed that rules for converting markup into accurate parsing constraints are still optimal for inter-punctuation fragments. Punctuation marks are more ubiquitous and natural than web markup: what little punctuation-induced constraints lack in precision, they more than make up in recall — perhaps both types of constraints would work better yet in tandem. For language acquisition, a natural question is whether prosody could similarly aid grammar induction from speech [159].

The results in this chapter underscore the power of simple models and algorithms, combined with common-sense constraints. They reinforce insights from *joint* modeling in *supervised* learning, where simplified, independent models, Viterbi decoding and expressive constraints excel at sequence labeling tasks [269]. Such evidence is particularly welcome in *unsupervised* settings [257], where it is crucial that systems scale gracefully to volumes of data, on top of the usual desiderata — ease of implementation, extension, understanding and debugging. Future work could explore softening constraints [132, 47], perhaps using features [92, 24] or by learning to associate different settings with various marks: Simply adding a hidden tag for "ordinary" versus "divide" types of punctuation [187] may already usefully extend the models covered in this chapter.

Chapter 8

Capitalization

The purpose of this chapter is to test the applicability of constraints also to capitalization changes in text for languages that use cased alphabets. Supporting peer-reviewed publication is *Capitalization Cues Improve Dependency Grammar Induction* in WILS 2012 [306].

8.1 Introduction

Since sentence structure is underdetermined by raw text, there have been efforts to simplify the task, via (i) pooling features of syntax across languages [64, 213, 66]; as well as (ii) identifying universal rules [228] — such as verbo-centricity [119] — that need not be learned at all. Unfortunately most of these techniques do not apply to plain text, because they require knowing, for example, which words are verbs. As standard practice in grammar induction shifts away from relying on gold part-of-speech (POS) tags [283, 253, 297, *inter alia*] (see also next chapter), lighter cues to inducing linguistic structure become more important. Examples of useful POS-agnostic clues include punctuation boundaries and various other kinds of bracketing constraints, from previous chapters. This chapter proposes adding capitalization to this growing list of sources of partial bracketings. The intuition here stems from English, where (maximal) spans of capitalized words — such as *Apple II*, *World War I*, *Mayor William H. Hudnut III*, *International Business Machines Corp.* and *Alexandria, Va* — tend to demarcate proper nouns.

Consider a motivating example (all of the examples in this chapter are also from WSJ) without punctuation, in which all (eight) capitalized word clumps and uncased numerals match base noun phrase constituent boundaries:

[$_{\textbf{NP}}$ Jay Stevens] of [$_{\textbf{NP}}$ Dean Witter] actually cut his per-share earnings estimate to [$_{\textbf{NP}}$ \$9] from [$_{\textbf{NP}}$ \$9.50] for [$_{\textbf{NP}}$ 1989] and to [$_{\textbf{NP}}$ \$9.50] from [$_{\textbf{NP}}$ \$10.35] in [$_{\textbf{NP}}$ 1990] because he decided sales would be even weaker than he had expected.

and another (whose first word happens to be a leaf), where capitalization complements punctuation cues:

[$_{\textbf{NP}}$ Jurors] in [$_{\textbf{NP}}$ U.S. District Court] in [$_{\textbf{NP}}$ Miami] cleared [$_{\textbf{NP}}$ Harold Hershhenson], a former executive vice president; [$_{\textbf{NP}}$ John Pagones], a former vice president; and [$_{\textbf{NP}}$ Stephen Vadas] and [$_{\textbf{NP}}$ Dean Ciporkin], who had been engineers with [$_{\textbf{NP}}$ Cordis].

Could such chunks help bootstrap grammar induction and/or improve the accuracy of already-trained unsupervised parsers? In answering these questions, this chapter will focus predominantly on sentence-internal capitalization. But it will also show that first words — those capitalized by convention — and uncased segments — whose characters are not even drawn from an alphabet — could play a useful role as well.

8.2 English Capitalization from a Treebank

As in the two previous chapters, this study begins by consulting the 51,558 parsed sentences of the WSJ corpus: 30,691 (59.5%) of them contain non-trivially capitalized *fragments* — maximal (non-empty and not sentence-initial) consecutive sequences of words that each differs from its own lower-cased form. Nearly all — 59,388 (96.2%) — of the 61,731 fragments are dominated by noun phrases; slightly less than half — 27,005 (43.8%) — perfectly align with constituent boundaries in the treebank; and about as many — 27,230 (44.1%) are multi-token. Table 8.1 shows the top POS sequences comprising fragments.

8.3 Analytical Experiments with Gold Trees

The suitability of capitalization-induced fragments for guiding dependency grammar induction can be gauged by assessing accuracy, in WSJ, of parsing constraints derived from their

	Count	*POS Sequence*	*Frac*	*Cum*
1	27,524	NNP	44.6%	
2	17,222	NNP NNP	27.9	72.5
3	4,598	NNP NNP NNP	7.5	79.9
4	2,973	JJ	4.8	84.8
5	1,716	NNP NNP NNP NNP	2.8	87.5
6	1,037	NN	1.7	89.2
7	932	PRP	1.5	90.7
8	846	NNPS	1.4	92.1
9	604	NNP NNPS	1.0	93.1
10	526	NNP NNP NNP NNP NNP	0.9	93.9
WSJ	+3,753	*more with Count* ≤ 498	6.1%	

Table 8.1: Top 10 fragments of POS tag sequences in WSJ.

end-points. Several such heuristics are tested, following the suite of increasingly-restrictive constraints on how dependencies may interact with fragments introduced in previous chapters. The most lenient constraint, *thread*, only asks that no dependency path from the root to a leaf enter the fragment twice; *tear* requires any incoming arcs to come from the same side of the fragment; *sprawl* demands that there be exactly one incoming arc; *loose* further constrains any outgoing arcs to be from the fragment's head; and *strict* — the most stringent constraint — bans external dependents. Since only *strict* is binding for single words, this chapter experiments also with *strict*′: applying *strict* solely to multi-token fragments (ignoring singletons). In sum, it explores six ways in which dependency parse trees can be constrained by fragments whose end-points could be defined by capitalization (or in other various ways, e.g., semantic annotations [227], punctuation or HTML tags in web pages).

For example, in the sentence about Cordis, the *strict* hypothesis would be wrong about five of the eight fragments: *Jurors* attaches *in*; *Court* takes the second *in*; *Hershhenson* and *Pagones* derive their titles, *president*; and (at least in one reference) *Vadas* attaches *and*, *Ciporkin* and *who*. Based on this, *strict* would be considered 37.5%-accurate. But *loose* — and the rest of the more relaxed constraints — would get perfect scores. (And *strict*′ would retract the mistake about *Jurors* but also the correct guesses about *Miami* and *Cordis*, scoring only 20%.) Table 8.2 (*capital*) shows scores averaged over the entire treebank. Columns *markup* and *punct* indicate that capitalization yields across-the-board

	markup	*punct.*	*capital*	*initial*	*uncased*
thread	98.5	95.0	**99.5**	98.4	99.2
tear	97.9	94.7	**98.6**	98.4	98.5
sprawl	95.1	92.9	**98.2**	97.9	96.4
loose	87.5	74.0	**97.9**	96.9	96.4
strict′	32.7	35.6	38.7	40.3	**55.6**
strict	35.6	39.2	59.3	**66.9**	61.1

Table 8.2: Several sources of fragments' end-points and %-correctness of their derived constraints (for English).

more accurate constraints (for English) compared with fragments derived from punctuation or markup (i.e., anchor text, bold, italics and underline tags in HTML), for which such constraints were originally intended.

8.4 Pilot Experiments on Supervised Parsing

To further test the potential of capitalization-induced constraints, they were applied in the Viterbi-decoding phase of a simple (unlexicalized) supervised dependency parser — an instance of DBM-1 (Ch. 10), trained on WSJ sentences with up to 45 words (excluding Section 23). Table 8.3 shows evaluation results on held-out data (all sentences), using "add-one" smoothing. All constraints other than *strict* improve accuracy by about a half-a-point,

	punct.:	*thread*	*tear*	*sprawl*	*loose*
none:	71.8	74.3	74.4	74.5	73.3
capital:thread	72.3	74.6	74.7	74.9	73.6
tear	72.4	74.7	74.7	74.9	73.6
sprawl	72.4	74.7	74.7	74.9	73.4
loose	72.4	74.8	74.7	**74.9**	73.3
strict′	*71.4*	73.7	73.7	73.9	72.7
strict	*71.0*	73.1	73.1	73.2	72.1

Table 8.3: Supervised (directed) accuracy on Section 23 of WSJ using capitalization-induced constraints (vertical) jointly with punctuation (horizontal) in Viterbi-decoding.

CoNLL Year	*Filtered Training*		*Directed Accuracies with Initial Constraints*							*Fragments*	
& Language	*Tokens*	*/ Sentences*	*none*	*thread*	*tear*	*sprawl*	*loose*	*strict′*	*strict*	*Multi*	*Single*
German 2006	139,333	12,296	36.3	36.3	36.3	**39.1**	*36.2*	36.3	*30.1*	3,287	30,435
Czech ’6	187,505	20,378	51.3	51.3	51.3	51.3	**52.5**	**52.5**	51.4	1,831	6,722
English ’7	74,023	5,087	29.2	*28.5*	*28.3*	*29.0*	**29.3**	*28.3*	*27.7*	1,135	2,218
Bulgarian ’6	46,599	5,241	59.4	*59.3*	*59.3*	59.4	*59.1*	*59.3*	**59.5**	184	1,506
Danish ’6	14,150	1,599	21.3	*17.7*	22.7	21.5	21.4	**31.4**	27.9	113	317
Greek ’7	11,943	842	28.1	46.1	46.3	46.3	**46.4**	31.1	31.0	113	456
Dutch ’6	72,043	7,107	**45.9**	*45.8*	45.9	*45.8*	*45.8*	*45.7*	*29.6*	89	4,335
Italian ’7	9,142	921	41.7	52.6	**52.7**	52.6	44.2	52.6	45.8	41	296
Catalan ’7	62,811	4,082	61.3	61.3	61.3	61.3	61.3	**61.3**	*36.5*	28	2,828
Turkish ’6	17,610	2,835	32.9	32.9	*32.2*	33.0	33.0	33.6	**33.9**	27	590
Portuguese ’6	24,494	2,042	68.9	*67.1*	69.1	**69.2**	68.9	68.9	*38.5*	9	953
Hungarian ’7	10,343	1,258	43.2	43.2	*43.1*	43.2	43.2	**43.7**	*25.5*	7	277
Swedish ’6	41,918	4,105	48.6	48.6	48.6	*48.5*	*48.5*	*48.5*	**48.8**	3	296
Slovenian ’6	3,627	477	30.4	30.5	30.5	30.4	30.5	30.5	**30.8**	1	63
		Median:	42.5	46.0	**46.1**	46.0	45.0	44.7	*32.5*		
		Mean:	42.8	44.4	44.8	**45.0**	44.3	44.6	*36.9*		

Table 8.4: Parsing performance for grammar inducers trained with capitalization-based initial constraints, tested against 14 held-out sets from 2006/7 CoNLL shared tasks, and ordered by number of multi-token fragments in training data.

from 71.8 to 72.4%, suggesting that capitalization is informative of certain regularities not captured by DBM grammars; moreover, it still continues to be useful when punctuation-based constraints are also enforced, boosting accuracy from 74.5 to 74.9%.

8.5 Multi-Lingual Grammar Induction

So far, this chapter showed only that capitalization information can be helpful in parsing a very specific genre of English. Its ability to generally aid dependency grammar induction is tested next, focusing on situations when other bracketing cues are unavailable. These experiments cover 14 CoNLL languages, excluding Arabic, Chinese and Japanese (which lack case), as well as Basque and Spanish (which are pre-processed in a way that loses relevant capitalization information). For all remaining languages training was only on simple sentences — those lacking sentence-internal punctuation — from the relevant training sets (for blind evaluation). Restricting attention to a subset of the available training data serves a dual purpose. First, it allows estimation of capitalization’s impact where no other (known or obvious) cues could also be used. Otherwise, unconstrained baselines would not yield the strongest possible alternative, and hence not the most interesting comparison. Second, to the extent that presence of punctuation may correlate with sentence complexity [107], there are benefits to "starting small" [95]: e.g., relegating full data to later stages helps training, as in many of the previous chapters.

The base systems induced DBM-1, starting from uniformly-at-random chosen parse trees [67] of each sentence, followed by inside-outside re-estimation [14] with "add-one" smoothing.[1] Capitalization-constrained systems differed from controls in exactly one way: each learner got a slight nudge towards more promising structures by choosing initial seed trees satisfying an appropriate constraint (but otherwise still uniformly). Table 8.4 contains the stats for all 14 training sets, ordered by number of multi-token fragments. Final accuracies on respective (disjoint, full) evaluation sets are improved by all constraints other than *strict*, with the highest average performance resulting from *sprawl*: 45.0% directed dependency accuracy,[2] on average. This increase of about two points over the base system's 42.8% is driven primarily by improvements in two languages (Greek and Italian).

8.6 Capitalizing on Punctuation in Inference

Until now this chapter avoided using punctuation in grammar induction, except to filter data. Yet the pilot experiments indicated that both kinds of information are helpful in the decoding stage of a supervised system. Indeed, this is also the case in unsupervised parsing.

Taking the trained models obtained using the *sprawl* nudge (from §8.5) and proceeding to again apply constraints in inference (as in §8.4), capitalization alone increased parsing accuracy only slightly, from 45.0 to 45.1%, on average. Using punctuation constraints instead led to more improved performance: 46.5%. Combining both types of constraints again resulted in slightly higher accuracies: 46.7%. Table 8.5 breaks down this last average performance number by language and shows the combined approach to be competitive with the previous state-of-the-art. Further improvements could be attained by also incorporating both constraints in training and with full data.

[1]Using "early-stopping lateen EM" (Ch. 5) instead of thresholding or waiting for convergence.

[2]Starting from five parse trees for each sentence (using constraints *thread* through *strict*$'$) was no better, at 44.8% accuracy.

CoNLL Year & Language		*this Chapter*	*State-of-the-Art Systems: POS- (i) Agnostic*		*(ii) Identified*	
Bulgarian	2006	**64.5**	44.3	L$_5$	**70.3**	S$_{pt}$
Catalan	’7	61.5	**63.8**	L$_5$	56.3	MZ$_{NR}$
Czech	’6	**53.5**	50.5	L$_5$	33.3*	MZ$_{NR}$
Danish	’6	20.6	**46.0**	RF	**56.5**	S$_{ar}$
Dutch	’6	**46.7**	32.5	L$_5$	**62.1**	MPH$_{el}$
English	’7	29.2	**50.3**	P	45.7	MPH$_{el}$
German	’6	**42.6**	33.5	L$_5$	**55.8**	MPH$_{nl}$
Greek	’7	**49.3**	39.0	MZ	**63.9**	MPH$_{en}$
Hungarian	’7	**53.7**	48.0	MZ	48.1	MZ$_{NR}$
Italian	’7	50.5	**57.5**	MZ	**69.1**	MPH$_{pt}$
Portuguese	’6	**72.4**	43.2	MZ	**76.9**	S$_{bg}$
Slovenian	’6	**34.8**	33.6	L$_5$	34.6	MZ$_{NR}$
Swedish	’6	**50.5**	50.0	L$_6$	**66.8**	MPH$_{pt}$
Turkish	’6	34.4	**40.9**	P	**61.3**	RF$_{H1}$
	Median:	**48.5**	45.2		**58.9**	
	Mean:	**46.7**	45.2		**57.2***	

Table 8.5: Unsupervised parsing with both capitalization- and punctuation-induced constraints in inference, tested against the 14 held-out sets from 2006/7 CoNLL shared tasks, and state-of-the-art results (all sentence lengths) for systems that: (i) are also POS-agnostic and monolingual, including L (Lateen EM, Tables 5.5–5.6) and P (Punctuation, Ch. 7); and (ii) rely on gold POS-tag identities to (a) discourage noun roots [202, MZ], (b) encourage verbs [259, RF], or (c) transfer delexicalized parsers [296, S] from resource-rich languages with parallel translations [213, MPH].

8.7 Discussion and A Few Post-Hoc Analyses

The discussion, thus far, has been English-centric. Nevertheless, languages differ in how they use capitalization (and even the rules governing a given language tend to change over time — generally towards having fewer capitalized terms). For instance, adjectives derived from proper nouns are not capitalized in French, German, Polish, Spanish or Swedish, unlike in English (see Table 8.1: JJ). And while English forces capitalization of the first-person pronoun in the nominative case, *I* (see Table 8.1: PRP), in Danish it is the plural second-person pronoun (also *I*) that is capitalized; further, formal pronouns (and their case-forms) are capitalized in German (*Sie* and *Ihre*, *Ihres*...), Italian, Slovenian, Russian and

CoNLL Year	*Capitalization-Induced Constraints*						*Punctuation-Induced Constraints*					
& Language	*thread*	*tear*	*sprawl*	*loose*	*strict*′	*strict*	*thread*	*tear*	*sprawl*	*loose*	*strict*′	*strict*
Arabic 2006	—	—	—	—	—	—	89.6	89.5	*81.9*	61.2	29.7	33.4
’7	—	—	—	—	—	—	90.9	90.6	83.1	*61.2*	29.5	35.2
Basque ’7	—	—	—	—	—	—	96.2	95.7	92.3	81.9	42.8	50.6
Bulgarian ’6	99.8	99.5	96.6	96.4	51.8	81.0	97.6	97.2	**96.1**	74.7	36.7	41.2
Catalan ’7	100	99.5	*95.0*	94.6	15.8	57.9	96.1	95.5	94.6	73.7	36.0	42.6
Chinese ’6	—	—	—	—	—	—	—	—	—	—	—	—
’7	—	—	—	—	—	—	—	—	—	—	—	—
Czech ’6	99.7	98.3	96.2	95.4	42.4	68.0	*89.4*	*89.2*	87.7	68.9	37.2	41.7
’7	99.7	98.3	96.1	95.4	42.6	67.6	89.5	89.3	87.8	69.3	37.4	41.9
Danish ’6	99.9	99.4	98.3	97.0	**59.0**	69.7	96.9	96.9	95.2	68.3	39.6	40.9
Dutch ’6	99.9	99.1	98.4	96.6	16.6	46.3	89.6	89.5	86.4	69.6	42.5	46.2
English ’7	*99.3*	98.7	98.0	96.0	17.5	*24.8*	91.5	91.4	90.6	76.5	39.6	42.3
German ’6	99.6	*98.0*	96.7	96.4	41.7	57.1	94.5	93.9	90.7	71.1	37.2	40.7
Greek ’7	99.9	99.3	98.5	96.6	13.6	50.1	91.3	91.0	89.8	75.7	43.7	47.0
Hungarian ’7	99.9	98.1	95.7	94.4	46.6	62.0	96.1	94.0	89.0	77.1	*28.9*	*32.6*
Italian ’7	99.9	99.6	**99.0**	98.8	*12.8*	68.2	97.1	96.8	96.0	77.8	44.7	47.9
Japanese ’6	—	—	—	—	—	—	**100**	**100**	95.4	**89.0**	**48.9**	**63.5**
Portuguese ’6	100	99.0	97.6	97.0	14.4	37.7	96.0	95.8	94.9	74.5	40.3	45.0
Slovenian ’6	**100**	99.8	98.9	**98.9**	52.0	**84.7**	93.3	93.3	92.6	72.7	42.7	45.8
Spanish ’6	—	—	—	—	—	—	96.5	96.0	95.2	75.4	33.4	40.9
Swedish ’6	99.8	99.6	99.0	97.0	24.7	58.4	90.8	90.4	87.4	66.8	31.1	33.9
Turkish ’6	**100**	99.8	96.2	*94.0*	22.8	42.8	99.8	99.7	95.1	76.9	37.7	42.0
’7	**100**	**99.9**	96.1	94.2	21.6	42.9	99.8	99.7	94.6	76.7	38.2	42.8
Max:	100	99.9	99.0	98.9	59.0	84.7	100	100	96.1	89.0	48.9	63.5
Mean:	99.8	99.1	97.4	96.4	30.8	57.7	94.6	94.2	91.7	74.0	38.5	43.3
Min:	99.3	98.0	95.0	94.0	12.8	24.8	89.4	89.2	81.9	61.2	28.9	32.6

Table 8.6: Accuracies for capitalization- and punctuation-induced constraints on all (full) 2006/7 CoNLL training sets.

Bulgarian.

In contrast to pronouns, single-word proper nouns — including personal names — are capitalized in nearly all European languages. Such shortest bracketings are not particularly useful for constraining sets of possible parse trees in grammar induction, however, compared to multi-word expressions; from this perspective, German appears less helpful than most cased languages, because of noun compounding, despite prescribing capitalization of all nouns. Another problem with longer word-strings in many languages is that, e.g., in French (as in English) lower-case prepositions may be mixed in with contiguous groups of proper nouns: even in surnames, the German particle *von* is not capitalized, although the Dutch *van* is, unless preceded by a given name or initial — hence *Van Gogh*, yet *Vincent van Gogh*.

8.7.1 Constraint Accuracies Across Languages

Since even related languages (e.g., Flemish, Dutch, German and English) can have quite different conventions regarding capitalization, one would not expect the same simple strategy to be uniformly useful — or useful in the same way — across disparate languages. To get a better sense of how universal our constraints may be, their accuracies were tabulated for the full training sets of the CoNLL data, *after* all grammar induction experiments had been executed. Table 8.6 shows that the less-strict capitalization-induced constraints all fall within narrow (yet high) bands of accuracies of just a few percentage points: 99–100% in the case of *thread*, 98–100% for *tear*, 95–99% for *sprawl* and 94–99% for *loose*. By contrast, the ranges for punctuation-induced constraints are all at least 10%. Nothing seems particularly special about Greek or Italian in these summaries that could explain their substantial improvements (18 and 11%, respectively — see Table 8.4), though Italian does appear to mesh best with the *sprawl* constraint (not by much, closely followed by Swedish). And English — the language from which the inspiration for this chapter was drawn — barely improved with capitalization-induced constraints (see Table 8.4) and caused the lowest accuracies of *thread* and *strict*.

These outcomes are not entirely surprising: some best- and worst-performing results are due to noise, since learning via non-convex optimization can be chaotic: e.g., in the case of Greek, applying 113 constraints to initial parse trees could have a significant impact on the first grammar estimated in training — and consequently also on a learner's final, converged model instance. Averages (i.e., means and medians) — computed over many data sets — could be expected to be more stable and meaningful than the outliers.

8.7.2 Immediate Impact from Capitalization

Next, consider two settings that are less affected by training noise: grammar inducers immediately after an initial step of constrained Viterbi EM and supervised DBM parsers (trained on sentences with up to 45 words), for various languages in the CoNLL sets. Table 8.7 shows effects of capitalization to be exceedingly mild, both if applied alone and in tandem with punctuation. Exploring better ways of incorporating this informative resource — perhaps as soft features, rather than as hard constraints — and in combination

CoNLL Year		*Bracketings*		*Unsupervised Training*				*Supervised Parsing*			
& Language		*capital.*	*punct.*	*init.*	*1-step*	*constrained*		*none*	*capital.*	*punct.*	*both*
Arabic	2006	—	101	18.4	20.6	—	—	59.8	—	—	—
	’7	—	311	19.0	23.5	—	—	63.5	—	—	—
Basque	’7	—	547	17.4	22.4	—	—	58.4	—	—	—
Bulgarian	’6	44	552	19.4	28.9	28.4	-0.5	76.7	76.8	78.1	78.2
Catalan	’7	24	398	18.0	25.1	25.4	+0.3	78.1	78.3	78.6	78.9
Chinese	’6	—	—	23.5	27.2	—	—	83.7	—	—	—
	’7	—	—	19.4	25.0	—	—	81.0	—	—	—
Czech	’6	48	549	18.6	19.7	19.8	+0.1	64.9	64.8	67.0	66.9
	’7	57	466	18.0	21.7	—	—	62.8	—	—	—
Danish	’6	85	590	19.5	27.4	26.0	-1.3	71.9	72.0	74.2	74.3
Dutch	’6	28	318	18.7	17.9	17.7	-0.1	*60.9*	*60.9*	*62.7*	*62.8*
English	’7	151	423	17.6	24.0	21.9	*-2.1*	65.2	65.6	68.5	68.4
German	’6	135	523	*16.4*	23.0	23.7	+0.7	70.7	70.7	71.5	71.4
Greek	’7	47	372	17.1	*17.1*	*16.6*	-0.5	71.3	71.6	73.5	73.7
Hungarian	’7	28	893	17.1	18.5	18.6	+0.1	67.3	67.2	69.8	69.6
Italian	’7	71	505	18.6	**32.5**	**34.2**	**+1.7**	66.0	65.9	67.0	66.8
Japanese	’6	—	0	26.5	36.8	—	—	85.1	—	—	—
Portuguese	’6	29	559	19.3	24.2	24.0	-0.1	**80.5**	**80.5**	**81.6**	**81.6**
Slovenian	’6	7	785	18.3	22.5	22.4	-0.1	67.5	67.4	70.9	70.9
Spanish	’6	—	453	18.0	19.3	—	—	69.5	—	—	—
Swedish	’6	14	417	20.2	31.4	31.4	+0.0	74.9	74.9	74.7	74.6
Turkish	’6	18	683	**20.4**	26.4	26.7	+0.3	66.1	66.0	66.9	66.7
	’7	4	305	20.3	24.8	—	—	67.3	—	—	—
			Max:	20.4	32.5	34.2	+1.7	80.5	80.5	81.6	81.6
			Mean:	18.5	24.2	24.1	-0.1	70.1	70.2	71.8	71.8
			Min:	16.4	17.1	16.6	-2.1	60.9	60.9	62.7	62.8

Table 8.7: Unsupervised accuracies for uniform-at-random projective parse trees (init), also after a step of Viterbi EM, and supervised performance with induced constraints, on 2006/7 CoNLL evaluation sets (sentences under 145 tokens).

with punctuation- and markup-induced bracketings could be a fruitful direction.

8.7.3 Odds and Ends

Earlier analyses in this chapter excluded sentence-initial words because their capitalization is, in a way, trivial. But for completeness, constraints derived from this source were also tested, separately (see Table 8.2: *initials*). As expected, the new constraints scored worse (despite many automatically-correct single-word fragments) except for *strict*, whose binding constraints over singletons drove *up* accuracy. It turns out, most first words in WSJ are leaves — possibly due to a dearth of imperatives (or just English's determiners).

The investigation of the "first leaf" phenomenon was broadened, discovering that in 16 of the 19 CoNLL languages first words are more likely to be leaves than other words without dependents on the left;[3] last words, by contrast, are *more* likely to take dependents than expected. These propensities may be related to the functional tendency of languages to place old information before new [334] and could also help bias grammar induction.

Lastly, capitalization points to yet another class of words: those with identical upper- and lower-case forms. Their constraints too tend to be accurate (see Table 8.2: *uncased*), but the underlying text is not particularly interesting. In WSJ, caseless multi-token fragments are almost exclusively percentages (e.g., the two tokens of *10%*), fractions (e.g., *1 1/4*) or both. Such boundaries could be useful in dealing with financial data, as well as for breaking up text in languages without capitalization (e.g., Arabic, Chinese and Japanese). More generally, transitions between different fonts and scripts should be informative too.

8.8 Conclusion

Orthography provides valuable syntactic cues. This chapter showed that bounding boxes signaled by capitalization changes can help guide grammar induction and boost unsupervised parsing performance. As with punctuation-delimited segments and tags from web markup, it is profitable to assume only that a single word derives the rest, in such text fragments, without further restricting relations to external words — possibly a useful feature for supervised parsing models. The results in this chapter should be regarded with some caution, however, since improvements due to capitalization in grammar induction experiments came mainly from two languages, Greek and Italian. Further research is clearly needed to understand the ways that capitalization can continue to improve parsing.

[3]Arabic, Basque, Bulgarian, Catalan, Chinese, Danish, Dutch, English, German, Greek, Hungarian, Italian, Japanese, Portuguese, Spanish, Swedish *vs.* Czech, Slovenian, Turkish.

Part III

Models

Chapter 9

Unsupervised Word Categories

The purpose of this chapter is to understand which properties of syntactic categories annotated by linguists, used up until now, make them particularly suitable as word classes for English grammar induction, and to construct a tagger, based on unsupervised word clustering algorithms, that eliminates this blatant source of supervision. Supporting peer-reviewed publication is *Unsupervised Dependency Parsing without Gold Part-of-Speech Tags* in EMNLP 2011 [310].

9.1 Introduction

Not all research on grammar induction has been fully unsupervised. For example, every new state-of-the-art dependency grammar inducer since the DMV relied on gold part-of-speech tags. For some time, multi-point performance degradations caused by switching to automatically induced word categories have been interpreted as indications that "good enough" parts-of-speech induction methods exist, justifying the focus on grammar induction with supervised part-of-speech tags [34], pace [75]. One of several drawbacks of this practice is that it weakens any conclusions that could be drawn about how computers (and possibly humans) learn in the absence of explicit feedback [213].

In turn, not all unsupervised taggers actually induce word categories: Many systems — known as part-of-speech *disambiguators* [219] — rely on external dictionaries of possible tags. The work in this chapter builds on two older part-of-speech *inducers* — word clustering algorithms of Clark [61] and Brown et al. [41] — that were recently shown to be more robust than other well-known fully unsupervised techniques [60].

This chapter investigates which properties of gold part-of-speech tags are useful in grammar induction and parsing, and how these properties could be introduced into induced tags. It also explores the number of word classes that is good for grammar induction: in particular, whether categorization is needed at all. By removing the "unrealistic simplification" of using gold tags [249, §3.2, Footnote 4], it goes on to demonstrate why grammar induction from plain text is no longer "still too difficult."

9.2 Methodology

All experiments model the English grammar, via the DMV, induced from subsets of not-too-long sentences of WSJ. This chapter imitates Klein's [170] set-up, initializing from an "ad-hoc harmonic" completion, followed by training with 40 steps of EM. Most of its experiments (#1–4, §9.3–9.4) also iterate without actually verifying convergence — but using more data (WSJ15 instead of WSJ10) — and are evaluated against the training set. Experiments #5–6 (§9.5) employ a state-of-the-art grammar inducer (from Ch. 7), which uses constrained Viterbi EM (details in §9.5). The final experiments (#5–6, §9.5) employ a simple scaffolding strategy (as in Ch. 3) that follows up initial training at WSJ15 ("less is more") with an additional training run ("leapfrog") that incorporates most sentences of the data set, at WSJ45. For a meaningful comparison with previous work, some of the models from earlier experiments (#1,3) — and both models from final experiments (#5,6) — are tested against Section 23 of WSJ$^{\infty}$, after applying Laplace (a.k.a. "add one") smoothing.

9.3 Motivation and Ablative Analyses

The concepts of polysemy and synonymy are of fundamental importance in linguistics. For words that can take on multiple parts of speech, knowing the gold tag can reduce ambiguity, improving parsing by limiting the search space. Furthermore, pooling the statistics of words that play similar syntactic roles, as signaled by shared gold part-of-speech tags, can

	Accuracy		*Viable*
1. **manual** tags	*Unsupervised*	*Sky*	*Groups*
gold	50.7	*78.0*	36
mfc	**47.2**	74.5	34
mfp	40.4	76.4	160
ua	44.3	**78.4**	328
2. tagless **lexicalized** models			
full	25.8	**97.3**	49,180
partial	29.3	60.5	176
none	**30.7**	*24.5*	1
3. tags from a **flat** [61] clustering			
	47.8	**83.8**	197
4. prefixes of a **hierarchical** [41] clustering			
first 7 bits	46.4	73.9	96
8 bits	**48.0**	77.8	165
9 bits	46.8	**82.3**	262

Table 9.1: Directed accuracies for the “less is more” DMV, trained on WSJ15 (after 40 steps of EM) and evaluated also against WSJ15, using various lexical categories in place of gold part-of-speech tags. For each tag-set, its effective number of (non-empty) categories in WSJ15 and the oracle skylines (supervised performance) are also reported.

simplify the learning task, improving generalization by reducing sparsity. This chapter begins with two sets of experiments that explore the impact that each of these factors has on grammar induction with the DMV.

9.3.1 Experiment #1: Human-Annotated Tags

The first set of experiments attempts to isolate the effect that replacing gold POS tags with deterministic *one class per word* mappings has on performance, quantifying the cost of switching to a monosemous clustering (see Table 9.1: manual; and Table 9.4). Grammar induction with gold tags scores 50.7%, while the oracle skyline (an ideal, supervised instance of the DMV) could attain 78.0% accuracy. It may be worth noting that only 6,620 (13.5%) of 49,180 unique tokens in WSJ appear with multiple POS tags. Most words, like

token	*mfc*	*mfp*	*ua*
it	{PRP}	{PRP}	{PRP}
gains	{NNS}	{VBZ, NNS}	{VBZ, NNS}
the	{DT}	{JJ, DT}	{VBP, NNP, NN, JJ, DT, CD}

Table 9.2: Example most frequent class, most frequent pair and union all reassignments for tokens *it*, *the* and *gains*.

it, are always tagged the same way (5,768 times PRP). Some words, like *gains*, usually serve as one part of speech (227 times NNS, as in *the gains*) but are occasionally used differently (5 times VBZ, as in *he gains*). Only 1,322 tokens (2.7%) appear with three or more different gold tags. However, this minority includes the most frequent word — *the* (50,959 times DT, 7 times JJ, 6 times NNP and once as each of CD, NN and VBP).[1]

This chapter experiments with three natural reassignments of POS categories (see Table 9.2). The first, *most frequent class* (mfc), simply maps each token to its most common gold tag in the entire WSJ (with ties resolved lexicographically). This approach discards two gold tags (types PDT and RBR are not most common for any of the tokens in WSJ15) and costs about three-and-a-half points of accuracy, in both supervised and unsupervised regimes. Another reassignment, *union all* (ua), maps each token to the *set* of all of its observed gold tags, again in the entire WSJ. This inflates the number of groupings by nearly a factor of ten (effectively lexicalizing the most ambiguous words),[2] yet improves the oracle skyline by half-a-point over actual gold tags; however, learning is harder with this tag-set, losing more than six points in unsupervised training. The last reassignment, *most frequent pair* (mfp), allows up to two of the most common tags into a token's label set (with ties, once again, resolved lexicographically). This intermediate approach performs strictly worse than *union all*, in both regimes.

[1]Some of these are annotation errors in the treebank [16, Figure 2]: such (mis)taggings can severely degrade the accuracy of part-of-speech disambiguators, without additional supervision [16, §5, Table 1].

[2]Kupiec [177] found that the 50,000-word vocabulary of the Brown corpus similarly reduces to ∼400 ambiguity classes.

9.3.2 Experiment #2: Lexicalization Baselines

The next set of experiments assesses the benefits of categorization, turning to lexicalized baselines that avoid grouping words altogether. All three models discussed below estimated the DMV *without* using the gold tags in any way (see Table 9.1: lexicalized).

First, not surprisingly, a fully-lexicalized model over nearly 50,000 unique words is able to essentially memorize the training set, supervised. (Without smoothing, it is possible to deterministically attach most rare words in a dependency tree correctly, etc.) Of course, local search is unlikely to find good instantiations for so many parameters, causing unsupervised accuracy for this model to drop in half.

The next experiment is an intermediate, partially-lexicalized approach. It mapped frequent words — those seen at least 100 times in the training corpus [133] — to their own individual categories, lumping the rest into a single "unknown" cluster, for a total of under 200 groups. This model is significantly worse for supervised learning, compared even with the monosemous clusters derived from gold tags; yet it is only slightly more learnable than the broken fully-lexicalized variant.

Finally, for completeness, a model that maps every token to the same one "unknown" category was trained. As expected, such a trivial "clustering" is ineffective in supervised training; however, it outperforms both lexicalized variants unsupervised,[3] strongly suggesting that lexicalization alone may be insufficient for the DMV and hinting that some degree of categorization is essential to its learnability.

9.4 Grammars over Induced Word Clusters

So far, this chapter has demonstrated the need for grouping similar words and estimated a bound on performance losses due to monosemous clusterings, in preparation for experimenting with induced POS tags. Two sets of established, publicly-available hard clustering assignments, each computed from a much larger data set than WSJ (approximately a million words) are used. The first is a flat mapping (200 clusters) constructed by training Clark's [61] distributional similarity model over several hundred million words from the

[3]Note that it also beats supervised training; this isn't a bug (Ch. 4 explain this paradox in the DMV).

Cluster #173		*Cluster #188*	
1.	open	1.	get
2.	free	2.	make
3.	further	3.	take
4.	higher	4.	find
5.	lower	5.	give
6.	similar	6.	keep
7.	leading	7.	pay
8.	present	8.	buy
9.	growing	9.	win
10.	increased	10.	sell
⋮		⋮	
37.	cool	42.	improve
⋮		⋮	
1,688.	up-wind	2,105.	zero-out

Table 9.3: Representative members for two of the flat word groupings: cluster #173 (left) contains adjectives, especially ones that take comparative (or other) complements; cluster #188 comprises bare-stem verbs (infinitive stems). (Of course, many of the words have other syntactic uses.)

British National and the English Gigaword corpora.[4] The second is a hierarchical clustering — binary strings up to eighteen bits long — constructed by running Brown et al.'s [41] algorithm over 43 million words from the BLLIP corpus, minus WSJ.[5]

9.4.1 Experiment #3: A Flat Word Clustering

This chapter's main purely unsupervised results are with a flat clustering [61],[4] that groups words having similar context distributions, according to Kullback-Leibler divergence. (A word's context is an ordered pair: its left- and right-adjacent neighboring words.) To avoid overfitting, an implementation from previous literature [103] was employed. The number of clusters (200) and the sufficient amount of training data (several hundred-million words) were tuned to a task (NER) that is not directly related to dependency parsing. (Table 9.3

[4]`http://nlp.stanford.edu/software/stanford-postagger-2008-09-28.tar.gz`: `models/egw.bnc.200`

[5]`http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz`

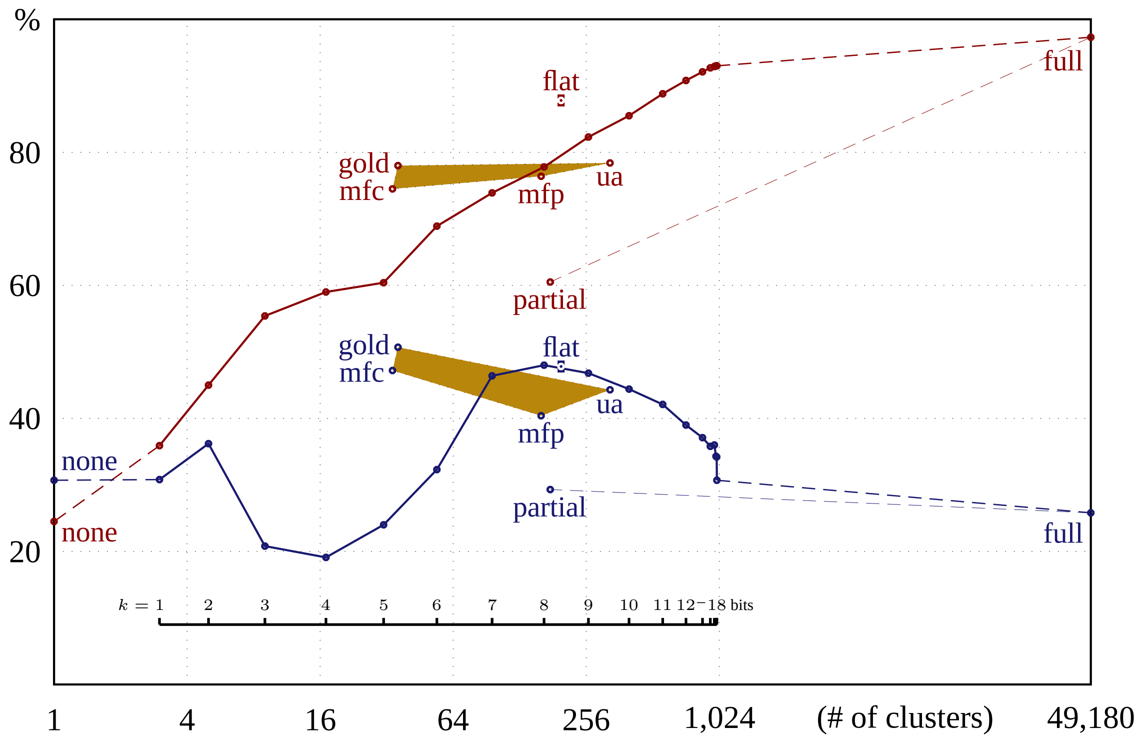


Figure 9.1: Parsing performance (accuracy on WSJ15) as a "function" of the number of syntactic categories, for all prefix lengths — $k \in \{1, \ldots, 18\}$ — of a hierarchical [41] clustering, connected by solid lines (dependency grammar induction in blue; supervised oracle skylines in red, above). Tagless lexicalized models (*full*, *partial* and *none*) connected by dashed lines. Models based on *gold* part-of-speech tags, and derived monosemous clusters (*mfc*, *mfp* and *ua*), shown as vertices of gold polygons. Models based on a *flat* [61] clustering indicated by squares.

shows representative entries for two of the clusters.)

One more category (#0) was added for unknown words. Now every token in WSJ could again be replaced by a coarse identifier (one of at most 201, instead of just 36), in both supervised and unsupervised training. (The training code did not change.) The resulting supervised model, though not as good as the fully-lexicalized DMV, was more than five points more accurate than with gold part-of-speech tags (see Table 9.1: flat). Unsupervised accuracy was lower than with gold tags (see also Table 9.4) but higher than with *all* three derived hard assignments. This suggests that polysemy (i.e., ability to tag a word differently in context) may be the primary advantage of manually constructed categorizations.

	System Description		*Accuracy*
#1 (§9.3.1)	"less is more"	(Ch. 3)	44.0
#3 (§9.4.1)	"less is more" with monosemous induced tags		41.4 (-2.6)

Table 9.4: Directed accuracies on Section 23 of WSJ (all sentences) for two experiments with the base system.

9.4.2 Experiment #4: A Hierarchical Clustering

The purpose of this batch of experiments is to show that Clark's [61] algorithm isn't unique in its suitability for grammar induction. Brown et al.'s [41] older information-theoretic approach, which does not explicitly address the problems of rare and ambiguous words [61] and was designed to induce large numbers of plausible syntactic *and* semantic clusters, can perform just as well, as it turns out (despite using less data).[6] Once again, the sufficient amount of text (43 million words) was tuned in earlier work [174]. Koo's task of interest was, in fact, dependency parsing. But since the algorithm is hierarchical (i.e., there isn't a parameter for the number of categories), it is doubtful that there was a strong risk of overfitting to question the clustering's unsupervised nature.

As there isn't a set number of categories, binary prefixes of length k from each word's address in the computed hierarchy were used as cluster labels. Results for $7 \leq k \leq 9$ bits (approximately 100–250 non-empty clusters, close to the 200 used before) are similar to those of flat clusters (see Table 9.1: hierarchical). Outside of this range, however, performance can be substantially worse (see Figure 9.1), consistent with earlier findings: Headden et al. [134] demonstrated that (constituent) grammar induction, using the singular-value decomposition (SVD-based) tagger of Schütze [281], also works best with 100–200 clusters. Important future research directions may include learning to automatically select a good number of word categories (in the case of flat clusterings) and ways of using multiple clustering assignments, perhaps of different granularities/resolutions, in tandem (e.g., in the case of a hierarchical clustering).

[6]One issue with traditional bigram class HMM objective functions, articulated by Martin et al. [204, §5.4], is that resulting clustering processes are dominated by the most frequent words, which are pushed towards a uniform distribution over the word classes. As a result, without a morphological component [62], there will not be a homogenous class of numbers or function words, in the end, because some such words appear often.

	System Description		*Accuracy*	
(§9.5)	"punctuation"	(Ch. 7)	58.4	
#5 (§9.5.1)	"punctuation" with monosemous induced tags		58.2	(-0.2)
#6 (§9.5.2)	"punctuation" with **context-sensitive** induced tags		**59.1**	(+**0.7**)

Table 9.5: Directed accuracies on Section 23 of WSJ (all sentences) for experiments with the state-of-the-art system.

9.4.3 Further Evaluation

It is important to enable easy comparison with previous and future work. Since WSJ15 is not a standard test set, two key experiments — "less is more" with gold part-of-speech tags (#1, Table 9.1: gold) and with Clark's [61] clusters (#3, Table 9.1: flat) — were re-evaluated on all sentences (not just length fifteen and shorter, which required smoothing both final models), in Section 23 of WSJ (see Table 9.4). This chapter thus showed that two classic unsupervised word clusterings — one flat and one hierarchical — can be better for dependency grammar induction than monosemous syntactic categories derived from gold part-of-speech tags. And it confirmed that the unsupervised tags are worse than the actual gold tags, in a simple dependency grammar induction system.

9.5 State-of-the-Art without Gold Tags

Until now, this chapter's experimental methods have been deliberately kept simple and nearly identical to the early work based on the DMV, for clarity. Next, let's explore how its main findings generalize beyond this toy setting. A preliminary test will simply quantify the effect of replacing gold part-of-speech tags with the monosemous flat clustering (as in experiment #3, §9.4.1) on a more modern grammar inducer. And the last experiment will gauge the impact of using a polysemous (but still unsupervised) clustering instead, obtained by executing standard sequence labeling techniques to introduce context-sensitivity into the original (independent) assignment of words to categories.

These final experiments are with a later state-of-the-art system (Ch. 7) — a partially lexicalized extension of the DMV that uses constrained Viterbi EM to train on nearly all of the data available in WSJ, at WSJ45. The key contribution that differentiates this model from its predecessors is that it incorporates punctuation into grammar induction (by turning it into parsing constraints, instead of ignoring punctuation marks altogether). In training, the model makes a simplifying assumption — that sentences can be split at punctuation and that the resulting fragments of text could be parsed independently of one another (these parsed fragments are then reassembled into full sentence trees, by parsing the sequence of their own head words). Furthermore, the model continues to take punctuation marks into account in inference (using weaker, more accurate constraints, than in training). This system scores 58.4% on Section 23 of WSJ$^{\infty}$ (see Table 9.5).

9.5.1 Experiment #5: A Monosemous Clustering

As in experiment #3 (§9.4.1), the base system was modified in exactly one way: gold POS tags were swapped out and replaced them with a flat distributional similarity clustering. In contrast to simpler models, which suffer multi-point drops in accuracy from switching to unsupervised tags (e.g., 2.6%), the newer system's performance degrades only slightly, by 0.2% (see Tables 9.4 and 9.5). This result improves over substantial performance degradations previously observed for unsupervised dependency parsing with induced word categories [172, 134, *inter alia*].

One risk that arises from using gold tags is that newer systems could be finding cleverer ways to exploit manual labels (i.e., developing an over-reliance on gold tags) instead of actually learning to acquire language. Part-of-speech tags are *known* to contain significant amounts of information for unlabeled dependency parsing [213, §3.1], so it is reassuring that this latest grammar inducer is *less* dependent on gold tags than its predecessors.

9.5.2 Experiment #6: A Polysemous Clustering

Results of experiments #1 and 3 (§9.3.1, 9.4.1) suggest that grammar induction stands to gain from relaxing the *one class per word* assumption. This conjecture is tested next, by inducing a polysemous unsupervised word clustering, then using it to induce a grammar.

Previous work [134, §4] found that simple bitag hidden Markov models, classically trained using the Baum-Welch [19] variant of EM (HMM-EM), perform quite well,[7] on average, across different grammar induction tasks. Such sequence models incorporate a sensitivity to context via state transition probabilities $\mathbb{P}_{\mathtt{TRAN}}(t_i \mid t_{i-1})$, capturing the likelihood that a tag t_i immediately follows the tag t_{i-1}; emission probabilities $\mathbb{P}_{\mathtt{EMIT}}(w_i \mid t_i)$ capture the likelihood that a word of type t_i is w_i.

A context-sensitive tagger is needed here, and HMM models are good — relative to other tag-inducers. However, they are not better than gold tags, at least when trained using a modest amount of data.[8] For this reason, the monosemous flat clustering will be relaxed, plugging it in as an initializer for the HMM [123]. The main problem with this approach is that, at least without smoothing, every monosemous labeling is trivially at a local optimum, since $\mathbb{P}(t_i \mid w_i)$ is deterministic. To escape the initial assignment, a "noise injection" technique [285] will be used, inspired by the contexts of [61, new]. First, the MLE statistics for $\mathbb{P}_{\mathtt{R}}(t_{i+1} \mid t_i)$ and $\mathbb{P}_{\mathtt{L}}(t_i \mid t_{i+1})$ will be collected from WSJ, using the flat monosemous tags. Next, WSJ text will be replicated 100-fold. Finally, this larger data set will be retagged, as follows: with probability 80%, a word keeps its monosemous tag; with probability 10%, a new tag is sampled from the left context ($\mathbb{P}_{\mathtt{L}}$) associated with the original (monosemous) tag of its rightmost neighbor; and with probability 10%, a tag is drawn from the right context ($\mathbb{P}_{\mathtt{R}}$) of its leftmost neighbor.[9] Given that the initializer — and later the input to the grammar inducer — are hard assignments of tags to words, (the faster and simpler) Viterbi training will be used to estimate this HMM's parameters.

In the spirit of reproducibility, again, an off-the-shelf component was used for tagging-related work.[10] Viterbi training converged after just 17 steps, replacing the original monosemous tags for 22,280 (of 1,028,348 non-punctuation) tokens in WSJ. For example, the first changed sentence is #3 (of 49,208):

Some "circuit breakers" installed after the October 1987 crash failed their first

[7]They are also competitive with Bayesian estimators, on larger data sets, with cross-validation [110].

[8]All of Headden et al.'s [134] grammar induction experiments with induced POS were worse than their best results with gold tags, most likely because of a very small corpus (half of WSJ10) used to cluster words.

[9]The sampling split (80:10:10) and replication parameter (100) were chosen somewhat arbitrarily, so better results could likely be obtained with tuning. However, the real gains would likely come from using soft clustering techniques [137, 246, *inter alia*] and propagating (joint) estimates of tag distributions into a parser. The ad-hoc approach presented here is intended to serve solely as a proof of concept.

[10]David Elworthy's `C+ tagger`, with options `-i t -G -l`, available from `http://friendly-moose.appspot.com/code/NewCpTag.zip`.

test, traders say, unable to ***cool*** *the selling panic in both stocks and futures.*

Above, the word *cool* gets relabeled as #188 (from #173 — see Table 9.3), since its context is more suggestive of an infinitive verb than of its usual grouping with adjectives.[11] Using this new context-sensitive hard assignment of tokens to unsupervised categories the latest grammar inducer attained a directed accuracy of 59.1%, nearly a full point better than with the monosemous hard assignment (see Table 9.5). To the best of my knowledge, it is also the first state-of-the-art unsupervised dependency parser to perform better with induced categories than with gold part-of-speech tags.

9.6 Related Work

Early work in dependency grammar induction already relied on gold part-of-speech tags [44]. Some later models [345, 244, *inter alia*] attempted full lexicalization. However, Klein and Manning [172] demonstrated that effort to be worse at recovering dependency arcs than choosing parse structures at random, leading them to incorporate gold tags into the DMV.

Klein and Manning [172, §5, Figure 6] had also tested their own models with induced word classes, constructed using a distributional similarity clustering method [281]. Without gold POS tags, their combined DMV+CCM model was about five points worse, both in (directed) unlabeled dependency accuracy (42.3% vs. 47.5%)[12] and unlabeled bracketing F_1 (72.9% vs. 77.6%), on WSJ10. In constituent parsing, earlier Seginer [283, §6, Table 1] built a fully-lexicalized grammar inducer that was competitive with DMV+CCM despite not using gold tags. His CCL parser has since been improved via a "zoomed learning" technique [263]. Moreover, Abend et al. [1] reused CCL's internal distributional representation of words in a cognitively-motivated part-of-speech inducer. Unfortunately their tagger did not make it into Christodoulopoulos et al.'s [60] excellent and otherwise comprehensive evaluation.

[11] A proper analysis of all changes, however, is beyond the scope of this work.

[12] On the same evaluation set (WSJ10), the context-sensitive system without gold tags (Experiment #6, §9.5.2) scores 66.8%.

Outside monolingual grammar induction, fully-lexicalized statistical dependency transduction models have been trained from unannotated parallel bitexts for machine translation [9]. More recently, McDonald et al. [213] demonstrated an impressive alternative to grammar induction by projecting reference parse trees from languages that have annotations to ones that are resource-poor.[13] It uses graph-based label propagation over a bilingual similarity graph for a sentence-aligned parallel corpus [77], inducing part-of-speech tags from a universal tag-set [249]. Even in supervised parsing there are signs of a shift away from using gold tags. For example, Alshawi et al. [10] demonstrated good results for mapping text to underspecified semantics via dependencies without resorting to gold tags. And Petrov et al. [248, §4.4, Table 4] observed only a small performance loss "going POS-less" in question parsing.

I am not aware of any systems that induce both syntactic trees and their part-of-speech categories. However, aside from the many systems that induce trees from gold tags, there are also unsupervised methods for inducing syntactic categories from gold trees [102, 246], as well as for inducing dependencies from gold constituent annotations [274, 58]. Considering that Headden et al.'s [134] study of part-of-speech taggers found no correlation between standard tagging metrics and the quality of induced grammars, it may be time for a unified treatment of these very related syntax tasks.

9.7 Discussion and Conclusions

Unsupervised word clustering techniques of Brown et al. [41] and Clark [61] are well-suited to dependency parsing with the DMV. Both methods outperform gold parts-of-speech in supervised modes. And both can do better than monosemous clusters derived from gold tags in unsupervised training. This chapter showed how Clark's [61] flat tags can be relaxed, using context, with the resulting polysemous clustering outperforming gold part-of-speech tags for the English dependency grammar induction task.

[13]When the target language is English, however, their best accuracy (projected from Greek) is low: 45.7% [213, §4, Table 2]; tested on the same CoNLL 2007 evaluation set [236], this chapter's "punctuation" system with context-sensitive induced tags (trained on WSJ45, without gold tags) performs substantially better, scoring 51.6%. Note that this is also an improvement over the same system trained on the CoNLL set using gold tags: 50.3% (see Table 7.7).

Monolingual evaluation is a significant flaw in this chapter's methodology, however. One (of many) take-home points made in Christodoulopoulos et al.'s [60] study is that results on one language do not necessarily correlate with other languages.[14] Assuming that the results do generalize, it will still remain to remove the present reliance on gold tokenization and sentence boundary labels. Nevertheless, eliminating gold tags has been an important step towards the goal of fully-unsupervised dependency parsing.

This chapter has cast the utility of a categorization scheme as a combination of two effects on parsing accuracy: a synonymy effect and a polysemy effect. Results of its experiments with both full and partial lexicalization suggest that grouping similar words (i.e., synonymy) is vital to grammar induction with the DMV. This is consistent with an established view-point, that simple tabulation of frequencies of words participating in certain configurations cannot be reliably used for comparing their likelihoods [246, §4.2]: "The statistics of natural languages is inherently ill defined. Because of Zipf's law, there is never enough data for a reasonable estimation of joint object distributions." Seginer's [284, §1.4.4] argument, however, is that the Zipfian distribution — a property of words, not parts-of-speech — should allow frequent words to successfully guide parsing and learning: "A relatively small number of frequent words appears almost everywhere and most words are never too far from such a frequent word (this is also the principle behind successful part-of-speech induction)." It is important to thoroughly understand how to reconcile these only seemingly conflicting insights, balancing them both in theory and in practice. A useful starting point may be to incorporate frequency information in the parsing models directly — in particular, capturing the relationships between words of various frequencies.

The polysemy effect appears smaller but is less controversial: This chapter's experimental results suggest that the primary drawback of the classic clustering schemes stems from their *one class per word* nature — and not a lack of supervision, as may be widely believed. Monosemous groupings, even if they are themselves derived from human-annotated syntactic categories, simply cannot disambiguate words the way gold tags can. By relaxing

[14]Furthermore, it would be interesting to know how sensitive different head-percolation schemes [339, 152] would be to gold versus unsupervised tags, since the Magerman-Collins rules [199, 70] agree with gold dependency annotations only 85% of the time, even for WSJ [274]. Proper intrinsic evaluation of dependency grammar inducers is not yet a solved problem [282].

Clark’s [61] flat clustering, using contextual cues, dependency grammar induction was improved: directed accuracy on Section 23 (all sentences) of the WSJ benchmark increased from 58.2% to 59.1% — from slightly worse to better than with gold tags (58.4%, previous state-of-the-art).

Finally, since Clark’s [61] word clustering algorithm is already context-sensitive in training, it is likely possible to do better simply by preserving the polysemous nature of its internal representation. Importing the relevant distributions into a sequence tagger directly would make more sense than going through an intermediate monosemous summary. And exploring other uses of *soft* clustering algorithms — perhaps as inputs to part-of-speech disambiguators — may be another fruitful research direction. A *joint* treatment of grammar and parts-of-speech induction could fuel major advances in both tasks.

Chapter 10

Dependency-and-Boundary Models

The purpose of this chapter is to introduce a new family of models for unsupervised dependency parsing, which is specifically designed to exploit the various informative cues that are observable at sentence and punctuation boundaries. Supporting peer-reviewed publication is *Three Dependency-and-Boundary Models for Grammar Induction* in EMNLP-CoNLL 2012 [307].

10.1 Introduction

Natural language is ripe with all manner of boundaries at the surface level that align with hierarchical syntactic structure. From the significance of function words [23] and punctuation marks [284, 253] as separators between constituents in longer sentences — to the importance of isolated words in children's early vocabulary acquisition [37] — word boundaries play a crucial role in language learning. This chapter will show that boundary information can also be useful in dependency grammar induction models, which traditionally focus on head rather than fringe words [44].

Consider again the example in Figure 1.1: *The check is in the mail*. Because the determiner (`DT`) appears at the left edge of the sentence, it should be possible to learn that determiners may generally be present at left edges of phrases. This information could then be used to correctly parse the sentence-internal determiner in *the mail*. Similarly, the fact that the noun head (`NN`) of the object *the mail* appears at the right edge of the sentence could help identify the noun *check* as the right edge of the subject NP. As with jigsaw puzzles, working inwards from boundaries helps determine sentence-internal structures of both noun phrases, neither of which would be quite so clear if viewed separately.

Furthermore, properties of noun-phrase edges are partially shared with prepositional- and verb-phrase units that contain these nouns. Because typical head-driven grammars model valency separately for each class of head, however, they cannot grasp that the left fringe boundary, *The check*, of the verb-phrase is shared with its daughter's, *check*. Neither of these insights is available to traditional dependency formulations, which could learn from the boundaries of this sentence only that determiners might have no left- and that nouns might have no right-dependents.

This chapter proposes a family of dependency parsing models that are capable of inducing longer-range implications from sentence edges than just fertilities of their fringe words. Its ideas conveniently lend themselves to implementations that can reuse much of the standard grammar induction machinery, including efficient dynamic programming routines for the relevant expectation-maximization algorithms.

10.2 The Dependency and Boundary Models

The new models follow a standard generative story for head-outward automata [7], restricted to the split-head case (see below), over lexical word classes $\{c_w\}$: first, a sentence root c_r is chosen, with probability $\mathbb{P}_{\texttt{ATTACH}}(c_r \mid \diamond;\ \texttt{L})$; $\diamond$ is a special start symbol that, by convention [172, 93], produces exactly one child, to its left. Next, the process recurses. Each (head) word c_h generates a left-dependent with probability $1 - \mathbb{P}_{\texttt{STOP}}(\ \cdot \mid \texttt{L};\ \cdots)$, where dots represent additional parameterization on which it may be conditioned. If the child is indeed generated, its identity c_d is chosen with probability $\mathbb{P}_{\texttt{ATTACH}}(c_d \mid c_h;\ \cdots)$, influenced by the identity of the parent c_h and possibly other parameters (again represented by dots). The child then generates its own subtree recursively and the whole process continues, moving away from the head, until c_h fails to generate a left-dependent. At that point, an analogous procedure is repeated to c_h's right, this time using stopping factors $\mathbb{P}_{\texttt{STOP}}(\ \cdot \mid \texttt{R};\ \cdots)$. All parse trees derived in this way are guaranteed to be projective and can be described by split-head grammars.

Instances of these split-head automata have been heavily used in grammar induction [244, 172, 133, *inter alia*], in part because they allow for efficient implementations [91, §8] of the inside-outside re-estimation algorithm [14]. The basic tenet of split-head grammars is that every head word generates its left-dependents independently of its right-dependents. This assumption implies, for instance, that words' left- and right-valencies — their numbers of children to each side — are also independent. But it does *not* imply that descendants that are closer to the head cannot influence the generation of farther dependents on the same side. Nevertheless, many popular grammars for unsupervised parsing behave as if a word had to generate all of its children (to one side) — or at least their count — *before* allowing any of these children themselves to recurse.

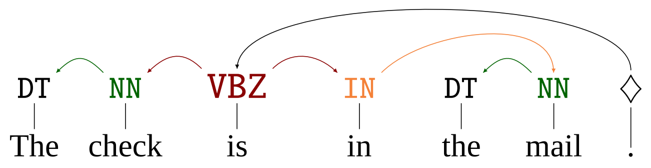
For example, the DMV could be implemented as both head-outward and head-inward automata. (In fact, arbitrary permutations of siblings to a given side of their parent would not affect the likelihood of the modified tree, with such models.) This chapter proposes to make fuller use of split-head automata's head-outward nature by drawing on information in partially-generated parses, which contain useful predictors that, previously, had not been exploited even in featurized systems for grammar induction [66, 24].

Some of these predictors, including the identity — or even number [207] — of already-generated siblings, can be prohibitively expensive in sentences above a short length k. For example, they break certain modularity constraints imposed by the charts used in $O(k^3)$-optimized algorithms [243, 89]. However, in bottom-up parsing and training from text, everything about the yield — i.e., the ordered sequence of all already-generated descendants, on the side of the head that is in the process of spawning off an additional child — is not only known but also readily accessible. This chapter introduces three new models for dependency grammar induction, designed to take advantage of this availability.

10.2.1 Dependency and Boundary Model One

DBM-1 conditions all stopping decisions on adjacency and the identity of the fringe word c_e — the currently-farthest descendant (edge) derived by head c_h in the given head-outward direction ($dir \in \{\mathtt{L}, \mathtt{R}\}$):

$$\mathbb{P}_{\mathtt{STOP}}(\,\cdot \mid dir;\ adj, c_e).$$



$$\begin{aligned}
\mathbb{P} \;=\; & (1-\overbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{L};\ \texttt{T})}^{0}) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{VBZ} \mid \diamond;\ \texttt{L})\\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T},\texttt{VBZ})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{NN} \mid \texttt{VBZ};\ \texttt{L})\\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T},\texttt{VBZ})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{IN} \mid \texttt{VBZ};\ \texttt{R})\\
\times\ & \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F},\texttt{DT})\ \texttt{// VBZ} && \times\ \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{F},\texttt{NN})\ \texttt{// VBZ}\\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T},\texttt{NN}))^2 && \times\ \mathbb{P}^2_{\texttt{ATTACH}}(\texttt{DT} \mid \texttt{NN};\ \texttt{L})\\
\times\ & (1-\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T},\texttt{IN})) && \times\ \mathbb{P}_{\texttt{ATTACH}}(\texttt{NN} \mid \texttt{IN};\ \texttt{R})\\
\times\ & \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T},\texttt{NN}) && \times\ \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F},\texttt{DT})\ \texttt{// NN}\\
\times\ & \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T},\texttt{IN}) && \times\ \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{F},\texttt{NN})\ \texttt{// IN}\\
\times\ & \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{T},\texttt{DT}) && \times\ \mathbb{P}^2_{\texttt{STOP}}(\,\cdot \mid \texttt{R};\ \texttt{T},\texttt{DT})\\
\times\ & \underbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{L};\ \texttt{F})}_{1} && \times\ \underbrace{\mathbb{P}_{\texttt{STOP}}(\diamond \mid \texttt{R};\ \texttt{T})}_{1}.
\end{aligned}$$

Figure 10.1: The running example — a simple sentence and its unlabeled dependency parse structure's probability, as factored by DBM-1; highlighted comments specify heads associated to non-adjacent stopping probability factors.

In the adjacent case ($adj = \texttt{T}$), c_h is deciding whether to have any children on a given side: a first child's subtree would be right next to the head, so the head and the fringe words coincide ($c_h = c_e$). In the non-adjacent case ($adj = \texttt{F}$), these will be different words and their classes will, in general, not be the same.[1] Thus, non-adjacent stopping decisions will be made independently of a head word's identity. Therefore, all word classes will be equally likely to continue to grow or not, for a specific proposed fringe boundary.

For example, production of *The check is* involves two non-adjacent stopping decisions on the left: one by the noun *check* and one by the verb *is*, both of which stop after generating a first child. In DBM-1, this outcome is captured by squaring a shared parameter belonging to the left-fringe determiner *The*: $\mathbb{P}_{\texttt{STOP}}(\ \cdot \mid \texttt{L};\ \texttt{F},\texttt{DT})^2$ — instead of by a product of two factors, such as $\mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F},\texttt{NN}) \cdot \mathbb{P}_{\texttt{STOP}}(\,\cdot \mid \texttt{L};\ \texttt{F},\texttt{VBZ})$. In DBM grammars, dependents' attachment probabilities, given heads, are additionally conditioned only on their relative positions — as in traditional models [172, 244]: $\mathbb{P}_{\texttt{ATTACH}}(c_d \mid c_h;\ dir)$.

Figure 10.1 shows a completely factored example.

[1]Fringe words differ also from standard dependency features [93, §2.3]: parse siblings and adjacent words.

10.2.2 Dependency and Boundary Model Two

DBM-2 allows different but related grammars to coexist in a single model. Specifically, it presupposes that all sentences are assigned to one of two classes: complete and incomplete ($comp \in \{\mathtt{T}, \mathtt{F}\}$, for now taken as exogenous). This model assumes that word-word (i.e., head-dependent) interactions in the two domains are the same. However, sentence lengths — for which stopping probabilities are responsible — and distributions of root words may be different. Consequently, an additional $comp$ parameter is added to the context of two relevant types of factors:

$$\mathbb{P}_{\texttt{STOP}}(\ \cdot \mid dir;\ adj, c_e, comp);$$
$$\text{and } \mathbb{P}_{\texttt{ATTACH}}(c_r \mid \diamond;\ \mathtt{L}, comp).$$

For example, the new stopping factors could capture the fact that incomplete fragments — such as the noun-phrases *George Morton*, headlines *Energy* and *Odds and Ends*, a line item *c - Domestic car*, dollar quantity *Revenue: $3.57 billion*, the time *1:11am*, and the like — tend to be much shorter than complete sentences. The new root-attachment factors could further track that incomplete sentences generally lack verbs, in contrast to other short sentences, e.g., *Excerpts follow:*, *Are you kidding?*, *Yes, he did.*, *It's huge.*, *Indeed it is.*, *I said, 'NOW?'*, *"Absolutely," he said.*, *I am waiting.*, *Mrs. Yeargin declined.*, *McGraw-Hill was outraged.*, *"It happens."*, *I'm OK, Jack.*, *Who cares?*, *Never mind.* and so on.

All other attachment probabilities $\mathbb{P}_{\texttt{ATTACH}}(c_d \mid c_h;\ dir)$ remain unchanged, as in DBM-1. In practice, $comp$ can indicate presence of sentence-final punctuation.

10.2.3 Dependency and Boundary Model Three

DBM-3 adds further conditioning on punctuation context. It introduces another boolean parameter, $cross$, which indicates the presence of intervening punctuation between a proposed head word c_h and its dependent c_d. Using this information, longer-distance punctuation-crossing arcs can be modeled separately from other, lower-level dependencies, via

$$\mathbb{P}_{\texttt{ATTACH}}(c_d \mid c_h;\ dir, cross).$$

For instance, in *Continentals believe <u>that</u> the strongest growth area <u>will</u> be southern Europe.*, four words appear between *that* and *will*. Conditioning on (the absence of) intervening punctuation could help tell true long-distance relations from impostors. All other

Split-Head Dependency Grammar		$\mathbb{P}_{\texttt{ATTACH}}$ *(head-root)*	$\mathbb{P}_{\texttt{ATTACH}}$ *(dependent-head)*	$\mathbb{P}_{\texttt{STOP}}$ *(adjacent and not)*
GB	[244]	$1\ /\ \lvert\{w\}\rvert$	$d \mid h;\ dir$	$1\ /\ 2$
DMV	[172]	$c_r \mid \diamond;\ \texttt{L}$	$c_d \mid c_h;\ dir$	$\cdot \mid dir;\ adj,\ c_h$
EVG	[133]	$c_r \mid \diamond;\ \texttt{L}$	$c_d \mid c_h;\ dir,\ adj$	$\cdot \mid dir;\ adj,\ c_h$
DBM-1	(§10.2.1)	$c_r \mid \diamond;\ \texttt{L}$	$c_d \mid c_h;\ dir$	$\cdot \mid dir;\ adj,\ c_e$
DBM-2	(§10.2.2)	$c_r \mid \diamond;\ \texttt{L},\ comp$	$c_d \mid c_h;\ dir$	$\cdot \mid dir;\ adj,\ c_e,\ comp$
DBM-3	(§10.2.3)	$c_r \mid \diamond;\ \texttt{L},\ comp$	$c_d \mid c_h;\ dir,\ cross$	$\cdot \mid dir;\ adj,\ c_e,\ comp$

Table 10.1: Parameterizations of the split-head-outward generative process used by DBMs and in previous models.

probabilities, $\mathbb{P}_{\texttt{STOP}}(\ \cdot \mid dir;\ adj, c_e, comp)$ and $\mathbb{P}_{\texttt{ATTACH}}(c_r \mid \diamond;\ \texttt{L}, comp)$, remain the same as in DBM-2.

10.2.4 Summary of DBMs and Related Models

Head-outward automata [7, 8, 9] played a central part as generative models for probabilistic grammars, starting with their early adoption in supervised split-head constituent parsers [69, 71]. Table 10.1 lists some parameterizations that have since been used by unsupervised dependency grammar inducers sharing their backbone split-head process.

10.3 Experimental Set-Up and Methodology

Let's first motivate each model by analyzing WSJ text, before delving into grammar induction experiments. Although motivating solely from this treebank biases the discussion towards a very specific genre of just one language, it has the advantage of allowing one to make concrete claims that are backed up by significant statistics.[2]

In the grammar induction experiments that follow, each model's incremental contribution to accuracies will be tested empirically, across many disparate languages. For each CoNLL data set, a baseline grammar will be induced using the DMV. Sentences with more than 15 tokens will be excluded, to create a conservative bias, because in this set-up the baseline is known to excel. All grammar inducers were initialized using (the same) uniformly-at-random chosen parse trees of training sentences [67]; thereafter, "add one" smoothing was applied at every training step.

[2]A kind of bias-variance trade-off, if you will...

To fairly compare the models under consideration — which could have quite different starting perplexities and ensuing consecutive relative likelihoods — two termination strategies where employed. In one case, each learner was blindly run through 40 steps of inside-outside re-estimation, ignoring any convergence criteria; in the other case, learners were run until numerical convergence of soft EM's objective function or until the likelihood of resulting Viterbi parse trees suffered — an "early-stopping lateen EM" strategy (Ch. 5).

Table 10.2 shows experimental results, averaged over all 19 CoNLL languages, for the DMV baselines and DBM-1 and 2. DBM-3 was not tested in this set-up because most sentence-internal punctuation occurs in longer sentences; instead, DBM-3 will be tested later (see §10.7), using most sentences,[3] in the final training step of a curriculum strategy [22] that will be proposed for DBMs. For the three models tested on shorter inputs (up to 15 tokens) both terminating criteria exhibited the same trend; lateen EM consistently scored slightly higher than 40 EM iterations.

Termination Criterion	DMV	DBM-1	DBM-2
40 steps of EM	33.5	38.8	40.7
early-stopping lateen EM	34.0	39.0	**40.9**

Table 10.2: Directed dependency accuracies, averaged over all 2006/7 CoNLL evaluation sets (all sentences), for the DMV and two new dependency-and-boundary grammar inducers (DBM-1 and 2) — using two termination strategies.

10.4 Dependency and Boundary Model One

The primary difference between DBM-1 and traditional models, such as the DMV, is that DBM-1 conditions non-adjacent stopping decisions on the identities of fringe words in partial yields (see §10.2.1).

[3]Results for DBM-3, given only standard input (up to length 15), would be nearly identical to DBM-2's.

Non-Adjacent Stop Predictor	R^2_{adj}	AIC_c
(dir)	0.0149	1,120,200
(n, dir)	0.0726	1,049,175
(c_h, dir)	0.0728	1,047,157
(c_e, dir)	0.2361	904,102.4
(c_h, c_e, dir)	0.3320	789,594.3

Table 10.3: Coefficients of determination (R^2) and Akaike information criteria (AIC), both adjusted for the number of parameters, for several single-predictor logistic models of non-adjacent stops, given direction dir; c_h is the class of the head, n is its number of descendants (so far) to that side, and c_e represents the farthest descendant (the edge).

10.4.1 Analytical Motivation

Treebank data suggests that the class of the fringe word — its part-of-speech, c_e — is a better predictor of (non-adjacent) stopping decisions, in a given direction dir, than the head's own class c_h. A statistical analysis of logistic regressions fitted to the data shows that the (c_h, dir) predictor explains only about 7% of the total variation (see Table 10.3). This seems low, although it is much better compared to direction alone (which explains less than 2%) and slightly better than using the (current) number of the head's descendants on that side, n, instead of the head's class. In contrast, using c_e in place of c_h boosts explanatory power to 24%, keeping the number of parameters the same. If one were willing to roughly square the size of the model, explanatory power could be improved further, to 33% (see Table 10.3), using both c_e and c_h.

Fringe boundaries thus appear to be informative even in the supervised case, which is not surprising, since using just one probability factor (and its complement) to generate very short (geometric coin-flip) sequences is a recipe for high entropy. But as suggested earlier, fringes should be extra attractive in unsupervised settings because yields are observable, whereas heads almost always remain hidden. Moreover, every sentence exposes two true edges [131]: integrated over many sample sentence beginnings and ends, cumulative knowledge about such markers can guide a grammar inducer inside long inputs, where structure is murky. Table 10.4 shows distributions of all POS tags in the treebank versus

POS	*% of All Tokens*	*First Tokens*	*Last Tokens*	*Sent. Roots*	*Frag. Roots*
NN	15.94	4.31	36.67	0.10	23.40
IN	11.85	13.54	0.57	0.24	4.33
NNP	11.09	20.49	12.85	0.02	32.02
DT	9.84	23.34	0.34	0.00	0.04
JJ	7.32	4.33	3.74	0.01	1.15
NNS	7.19	4.49	20.64	0.15	17.12
CD	4.37	1.29	6.92	0.00	3.27
RB	3.71	5.96	3.88	0.00	1.50
VBD	3.65	0.09	3.52	46.65	0.93
VB	3.17	0.44	1.67	0.48	6.81
CC	2.86	5.93	0.00	0.00	0.00
TO	2.67	0.37	0.05	0.02	0.44
VBZ	2.57	0.17	1.65	28.31	0.93
VBN	2.42	0.61	2.57	0.65	1.28
PRP	2.08	9.04	1.34	0.00	0.00
VBG	1.77	1.26	0.64	0.10	0.97
VBP	1.50	0.05	0.61	14.33	0.71
MD	1.17	0.07	0.05	8.88	0.57
POS	1.05	0.00	0.11	0.01	0.04
PRP$	1.00	0.90	0.00	0.00	0.00
WDT	0.52	0.08	0.00	0.01	0.13
JJR	0.39	0.18	0.43	0.00	0.09
RP	0.32	0.00	0.42	0.00	0.00
NNPS	0.30	0.20	0.56	0.00	2.96
WP	0.28	0.42	0.01	0.01	0.04
WRB	0.26	0.78	0.02	0.01	0.31
JJS	0.23	0.27	0.06	0.00	0.00
RBR	0.21	0.20	0.54	0.00	0.04
EX	0.10	0.75	0.00	0.00	0.00
RBS	0.05	0.06	0.01	0.00	0.00
PDT	0.04	0.08	0.00	0.00	0.00
FW	0.03	0.01	0.05	0.00	0.09
WP$	0.02	0.00	0.00	0.00	0.00
UH	0.01	0.08	0.05	0.00	0.62
SYM	0.01	0.11	0.01	0.00	0.18
LS	0.01	0.09	0.00	0.00	0.00

Table 10.4: Empirical distributions for non-punctuation POS tags in WSJ, ordered by overall frequency, as well as distributions for sentence boundaries and for the roots of complete and incomplete sentences. (A uniform distribution would have $1/36 = 2.\overline{7}\%$ for all tags.)

in sentence-initial, sentence-final and sentence-root positions. WSJ often leads with determiners, proper nouns, prepositions and pronouns — all good candidates for starting English

$\sqrt{1-\sum_x \sqrt{p_x q_x}}$	*All*	*First*	*Last*	*Sent.*	*Frag.*
Uniform	0.48	0.58	0.64	0.79	0.65
All		0.35	0.40	0.79	0.42
First			0.59	**0.94**	0.57
Last				**0.83**	0.29
Sent.					**0.86**

Table 10.5: A distance matrix for all pairs of probability distributions over POS-tags shown in Table 10.4 and the uniform distribution; the BC- (or Hellinger) distance [28, 235] between discrete distributions p and q (over $x \in \mathcal{X}$) ranges from zero (iff $p = q$) to one (iff $p \cdot q = 0$, i.e., when they do not overlap at all).

phrases; and its sentences usually end with various noun types, again consistent with the running example.

10.4.2 Experimental Results

Table 10.2 shows DBM-1 to be substantially more accurate than the DMV, on average: 38.8 versus 33.5% after 40 steps of EM.[4] Lateen termination improved both models' accuracies slightly, to 39.0 and 34.0%, respectively, with DBM-1 scoring five points higher.

10.5 Dependency and Boundary Model Two

DBM-2 adapts DBM-1 grammars to two classes of inputs (complete sentences and incomplete fragments) by forking off new, separate multinomials for stopping decisions and root-distributions (see §10.2.2).

10.5.1 Analytical Motivation

Unrepresentative short sentences — such as headlines and titles — are common in news-style data and pose a known nuisance to grammar inducers. Previous research sometimes

[4]DBM-1's 39% average accuracy with uniform-at-random initialization is two points above DMV's scores with the "ad-hoc harmonic" strategy, 37% (see Table 5.5).

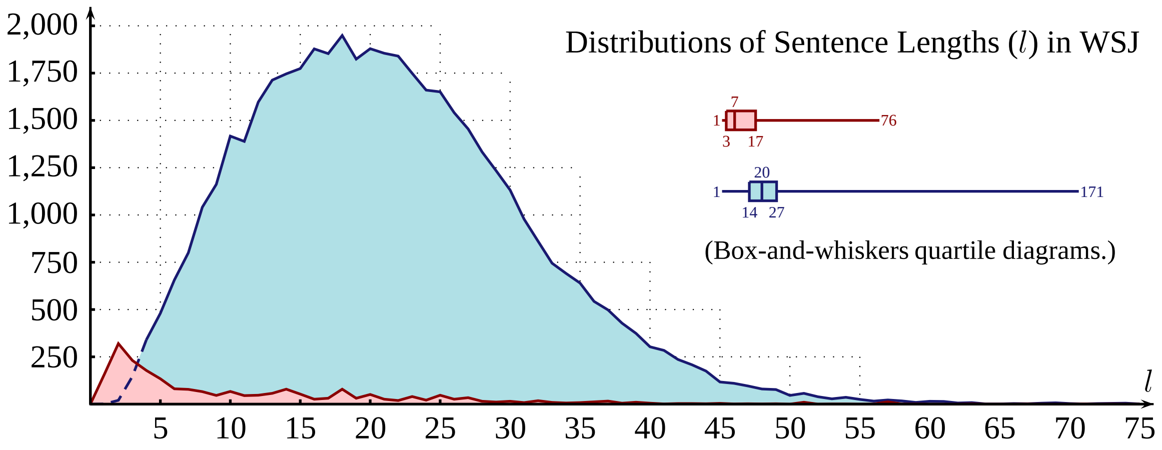


Figure 10.2: Histograms of lengths (in tokens) for 2,261 non-clausal fragments (red) and other sentences (blue) in WSJ.

took radical measures to combat the problem: for example, Gillenwater et al. [118] excluded all sentences with three or fewer tokens from their experiments; and Mareček and Žabokrtský [202] enforced an "anti-noun-root" policy to steer their Gibbs sampler away from the undercurrents caused by the many short noun-phrase fragments (among sentences up to length 15, in Czech data). This chapter will refer to such snippets of text as "incomplete sentences" and focus its study of WSJ on non-clausal data (as signaled by top-level constituent annotations whose first character is not S).[5]

Table 10.4 shows that roots of incomplete sentences, which are dominated by nouns, barely resemble the other roots, drawn from more traditional verb and modal (MD) types. In fact, these two empirical root distributions are more distant from one another than either is from the uniform distribution, in the space of discrete probability distributions over POS-tags (see Table 10.5). Of the distributions under consideration, only sentence boundaries are as or more different from (complete) roots, suggesting that heads of fragments too may warrant their own multinomial in a model.

Further, incomplete sentences are uncharacteristically short (see Figure 10.2). It is this property that makes them particularly treacherous to grammar inducers, since by offering few options of root positions they increase the chances that a learner will incorrectly induce

[5]I.e., separating top-level types {S, SINV, SBARQ, SQ, SBAR} from the rest (ordered by frequency): {NP, FRAG, X, PP, . . .}.

nouns to be heads. Given that expected lengths are directly related to stopping decisions, it makes sense to also model the stopping probabilities of incomplete sentences separately.

10.5.2 Experimental Results

Since it is not possible to consult parse trees during grammar induction (to check whether an input sentence is clausal), a simple proxy was used instead: presence of sentence-final punctuation. Using punctuation to divide input sentences into two groups, DBM-2 scored higher: 40.9, up from 39.0% accuracy (see Table 10.2).

After evaluating these multilingual experiments, the quality of the proxy's correspondence to actual clausal sentences in WSJ was examined. Table 10.6 shows the binary confusion matrix having a fairly low (but positive) Pearson correlation coefficient. False positives include parenthesized expressions that are marked as noun-phrases, such as *(See related story: "Fed Ready to Inject Big Funds": WSJ Oct. 16, 1989)*; false negatives can be headlines having a main verb, e.g., *Population Drain Ends For Midwestern States*. Thus, the proxy is not perfect but seems to be tolerable in practice. Identities of punctuation marks [71, Footnote 13] — both sentence-final and sentence-initial — could be of extra assistance in grammar induction, for grouping imperatives, questions, and so forth.

$r_\phi \approx 0.31$	*Clausal*	*non-Clausal*	*Total*
Punctuation	46,829	1,936	48,765
no Punctuation	118	325	443
Total	46,947	2,261	49,208

Table 10.6: A contingency table for clausal sentences and trailing punctuation in WSJ; the mean square contingency coefficient r_ϕ signifies a low degree of correlation. (For two binary variables, r_ϕ is equivalent to Karl Pearson's better-known product-moment correlation coefficient, ρ.)

10.6 Dependency and Boundary Model Three

DBM-3 exploits sentence-internal punctuation contexts by modeling punctuation-crossing dependency arcs separately from other attachments (see §10.2.3).

10.6.1 Analytical Motivation

Many common syntactic relations, such as between a determiner and a noun, are unlikely to hold over long distances. (In fact, 45% of all head-percolated dependencies in WSJ are between adjacent words.) However, some common constructions are more remote: e.g., subordinating conjunctions are, on average, 4.8 tokens away from their dependent modal verbs. Sometimes longer-distance dependencies can be vetted using sentence-internal punctuation marks.

It happens that the presence of punctuation between such conjunction (IN) and verb (MD) types serves as a clue that they are not connected (see Table 10.7*a*); by contrast, a simpler cue — whether these words are adjacent — is, in this case, hardly of any use (see Table 10.7*b*). Conditioning on crossing punctuation could be of help then, playing a role similar to that of comma-counting [69, §2.1] — and "verb intervening" [29, §5.1] — in early head-outward models for supervised parsing.

a) $r_\phi \approx -0.40$	*Attached*	*not Attached*	*Total*
Punctuation	337	7,645	7,982
no Punctuation	2,144	4,040	6,184
Total	2,481	11,685	14,166
non-Adjacent	2,478	11,673	14,151
Adjacent	3	12	15
b) $r_\phi \approx +0.00$	*Attached*	*not Attached*	*Total*

Table 10.7: Contingency tables for IN right-attaching MD, among closest ordered pairs of these tokens in WSJ sentences with punctuation, versus: (a) presence of intervening punctuation; and (b) presence of intermediate words.

10.6.2 Experimental Results Postponed

As mentioned earlier (see §10.3), there is little point in testing DBM-3 with shorter sentences, since most sentence-internal punctuation occurs in longer inputs. Instead, this model will be tested in a final step of a staged training strategy, with more data (see §10.7.3).

10.7 A Curriculum Strategy for DBMs

The proposal is to train up to DBM-3 iteratively — by beginning with DBM-1 and gradually increasing model complexity through DBM-2, drawing on the intuitions of IBM translation models 1–4 [40]. Instead of using sentences of up to 15 tokens, as in all previous experiments (§10.4–10.5), nearly all available training data will now be used: up to length 45 (out of concern for efficiency), during later stages. In the first stage, however, DBM-1 will make use of only a subset of the data, in a process sometimes called *curriculum learning* [22, 175, *inter alia*]. The grammar inducers will thus be "starting small" in both senses suggested by Elman [95]: simultaneously scaffolding on model- *and* data-complexity.

10.7.1 Scaffolding Stage #1: DBM-1

DBM-1 training begins from sentences without sentence-internal punctuation but with at least one trailing punctuation mark. The goal here is to avoid, when possible, overly specific arbitrary parameters like the "15 tokens or less" threshold used to select training sentences. Unlike DBM-2 and 3, DBM-1 does not model punctuation or sentence fragments, so it makes sense to instead explicitly restrict its attention to this cleaner subset of the training data, which takes advantage of the fact that punctuation may generally correlate with sentence complexity [107]. (Next chapters cover even more incremental training strategies.)

Aside from input sentence selection, the experimental set-up here remained identical to previous training of DBMs (§10.4–10.5). Using this new input data, DBM-1 averaged 40.7% accuracy (see Table 10.8). This is slightly higher than the 39.0% when using sentences up to length 15, suggesting that the proposed heuristic for clean, simple sentences may be a useful one.

			Directed Dependency Accuracies for:					*Best of State-of-the-Art Systems*					
CoNLL Year			*this Work*				(@10)	*Monolingual; POS-*				*Crosslingual*	
& Language		DMV	DBM-1	DBM-2	DBM-3	+*inference*		*(i) Agnostic*		*(ii) Identified*		*(iii) Transfer*	
Arabic	2006	12.9	10.6	11.0	11.1	10.9	*(34.5)*	**33.4**	L_6	—		**50.2**	S_{bg}
	’7	36.6	43.9	44.0	44.4	44.9	*(48.8)*	**55.6**	RF	*54.6*	RF_{H1}	—	
Basque	’7	32.7	34.1	33.0	32.7	33.3	*(36.5)*	**43.6**	L_5	*34.7*	MZ_{NR}	—	
Bulgarian	’7	24.7	59.4	63.6	64.6	**65.2**	*(70.4)*	44.3	L_5	*53.9*	$RF_{H1\&2}$	**70.3**	S_{pt}
Catalan	’7	41.1	61.3	61.1	61.1	62.1	*(78.1)*	**63.8**	L_5	*56.3*	MZ_{NR}	—	
Chinese	’6	50.4	63.1	63.0	63.2	63.2	*(65.7)*	**63.6**	L_6	—		—	
	’7	55.3	56.8	57.0	57.1	57.0	*(59.8)*	**58.5**	L_6	*34.6*	MZ_{NR}	—	
Czech	’6	31.5	51.3	52.8	53.0	**55.1**	*(61.8)*	50.5	L_5	—		—	
	’7	34.5	50.5	51.2	53.3	**54.2**	*(67.3)*	49.8	L_5	*42.4*	$RF_{H1\&2}$	—	
Danish	’6	22.4	21.3	19.9	21.8	22.2	*(27.4)*	**46.0**	RF	**53.1**	$RF_{H1\&2}$	**56.5**	S_{ar}
Dutch	’6	44.9	45.9	46.5	46.0	**46.6**	*(48.6)*	32.5	L_5	**48.8**	$RF_{H1\&2}$	**65.7**	$MPH_{m:p}$
English	’7	32.3	29.2	28.6	29.0	29.6	*(51.4)*	**50.3**	P	*23.8*	MZ_{NR}	*45.7*	MPH_{el}
German	’6	27.7	36.3	37.9	38.4	**39.1**	*(52.1)*	33.5	L_5	*21.8*	MZ_{NR}	**56.7**	$MPH_{m:d}$
Greek	’6	36.3	28.1	26.1	26.1	26.9	*(36.8)*	**39.0**	MZ	*33.4*	MZ_{NR}	**65.1**	$MPH_{m:p}$
Hungarian	’7	23.6	43.2	52.1	57.4	**58.2**	*(68.4)*	48.0	MZ	*48.1*	MZ_{NR}	—	
Italian	’7	25.5	41.7	39.8	39.9	40.7	*(41.8)*	**57.5**	MZ	**60.6**	MZ_{NR}	**69.1**	MPH_{pt}
Japanese	’6	42.2	22.8	22.7	22.7	22.7	*(32.5)*	**56.6**	L_5	*53.5*	MZ_{NR}	—	
Portuguese	’6	37.1	68.9	72.3	71.1	**72.4**	*(80.6)*	43.2	MZ	*55.8*	$RF_{H1\&2}$	**76.9**	S_{bg}
Slovenian	’6	33.4	30.4	33.0	34.1	**35.2**	*(36.8)*	33.6	L_5	*34.6*	MZ_{NR}	—	
Spanish	’6	22.0	25.0	26.7	27.1	28.2	*(51.8)*	**53.0**	MZ	**54.6**	MZ_{NR}	**68.4**	MPH_{it}
Swedish	’6	30.7	48.6	50.3	50.0	**50.7**	*(63.2)*	50.0	L_6	*34.3*	$RF_{H1\&2}$	**68.0**	$MPH_{m:p}$
Turkish	’6	**43.4**	32.9	33.7	33.4	34.4	*(38.1)*	40.9	P	**61.3**	RF_{H1}	—	
	’7	**58.5**	44.6	44.2	43.7	44.8	*(44.4)*	48.8	L_6	—		—	
	Average:	33.6	40.7	41.7	42.2	**42.9**	*(51.9)*	38.2	L_6	(*best average*, **not** an average of bests)			

Table 10.8: Average accuracies over CoNLL evaluation sets (all sentences), for the DMV baseline, DBM1–3 trained with a curriculum strategy, and state-of-the-art results for systems that: (i) are also POS-agnostic and monolingual, including L (Lateen EM, Tables 5.5–5.6) and P (Punctuation, Ch. 7); (ii) rely on gold tag identities to discourage noun roots [202, MZ] or to encourage verbs [259, RF]; and (iii) transfer delexicalized parsers [296, S] from resource-rich languages with translations [213, MPH]. DMV and DBM-1 were trained on simple sentences, starting from (the same) parse trees chosen uniformly-at-random; DBM-2 and 3 were trained on most sentences, starting from DBM-1 and 2’s output, respectively; +*inference* is DBM-3 with punctuation constraints.

10.7.2 Scaffolding Stage #2: DBM-2 ← DBM-1

Next comes training on all sentences up to length 45. Since these inputs are punctuation-rich, both remaining stages employed the constrained Viterbi EM set-up (Ch. 7) instead of plain soft EM; also, an early termination strategy was used, quitting hard EM as soon as soft EM’s objective suffered (Ch. 5). Punctuation was converted into Viterbi-decoding constraints during training using the so-called *loose* method, which stipulates that all words in an inter-punctuation fragment must be dominated by a single (head) word, also from that fragment — with only these head words allowed to attach the head words of other fragments, across punctuation boundaries. To adapt to full data, DBM-2 was initialized using Viterbi parses from the previous stage (§10.7.1), plus uniformly-at-random chosen dependency trees for any new complex and incomplete sentences, subject to punctuation-induced constraints. This approach improved parsing accuracies to 41.7% (see Table 10.8).

10.7.3 Scaffolding Stage #3: DBM-3 $\leftarrow$ DBM-2

Next, the training process of the previous stage (§10.7.2) was repeated using DBM-3. To initialize this model, the final instance of DBM-2 was combined with uniform multinomials for punctuation-crossing attachment probabilities (see §10.2.3). As a result, average performance improved to 42.2% (see Table 10.8). Lastly, punctuation constraints were applied also in inference. Here the *sprawl* method was used — a more relaxed approach than in training, allowing arbitrary words to attach inter-punctuation fragments (provided that each entire fragment still be derived by one of its words). This technique increased DBM-3's average accuracy to 42.9% (see Table 10.8). The final result substantially improves over the baseline's 33.6% and compares favorably to previous work.[6]

10.8 Discussion and the State-of-the-Art

DBMs come from a long line of head-outward models for dependency grammar induction yet their generative processes feature important novelties. One is conditioning on more observable state — specifically, the left and right end words of a phrase being constructed — than in previous work. Another is allowing multiple grammars — e.g., of complete and incomplete sentences — to coexist in a single model. These improvements could make DBMs quick-and-easy to bootstrap directly from any available partial bracketings [245], for example web markup (Ch. 6) or capitalized phrases (Ch. 8).

The second part of this chapter — the use of a curriculum strategy to train DBM-1 through 3 — eliminates having to know tuned cut-offs, such as sentences with up to a predetermined number of tokens. Although this approach adds some complexity, choices were

[6]Note that DBM-1's 39% average accuracy with standard training (see Table 10.2) was already nearly a full point higher than that of any single previous best system (L_6 — see Table 10.8).

made conservatively, to avoid overfitting settings of sentence length, convergence criteria, etc.: stage one's data is dictated by DBM-1 (which ignores punctuation); subsequent stages initialize additional pieces uniformly: uniform-at-random parses for new data and uniform multinomials for new parameters.

Even without curriculum learning — trained with vanilla EM — DBM-2 and 1 are already strong. Further boosts to accuracy could come from employing more sophisticated optimization algorithms, e.g., better EM [273], constrained Gibbs sampling [202] or locally-normalized features [24]. Other orthogonal dependency grammar induction techniques — including ones based on universal rules [228] — may also benefit in combination with DBMs. Direct comparisons to previous work require some care, however, as there are several classes of systems that make different assumptions about training data (see Table 10.8).

10.8.1 Monolingual POS-Agnostic Inducers

The first type of grammar inducers, including this chapter's approach, uses standard training and test data sets for each language, with gold POS tags as anonymized word classes. For the purposes of this discussion, transductive learners that may train on data from the test sets will also be included in this group. DBM-3 (decoded with punctuation constraints) does well among such systems — for which accuracies on *all* sentence lengths of the evaluation sets are reported — attaining highest scores for 8 of 19 languages; the DMV baseline is still state-of-the-art for one language; and the remaining 10 bests are split among five other recent systems (see Table 10.8).[7] Half of the five came from various lateen EM strategies (Ch. 5) for escaping and/or avoiding local optima. These heuristics are compatible with how the DBMs were trained and could potentially provide further improvement to accuracies.

Overall, the final scores of DBM-3 were better, on average, than those of any other single system: 42.9 versus 38.2% (Ch. 5). The progression of scores for DBM-1 through 3 without using punctuation constraints in inference — 40.7, 41.7 and 42.2% — fell entirely above this previous state-of-the-art result as well; the DMV baseline — also trained on

[7]But for Turkish '06, the "right-attach" baseline performs better, at 65.4% [259, Table 1] (an important difference between 2006 and 7 CoNLL data has to do with segmentation of morphologically-rich languages).

sentences without internal but with final punctuation — averaged 33.6%.

10.8.2 Monolingual POS-Identified Inducers

The second class of techniques assumes knowledge about identities of POS tags [228], i.e., which word tokens are verbs, which ones are nouns, etc. Such grammar inducers generally do better than the first kind — e.g., by encouraging verbocentricity [119] — though even here DBMs' results appear to be competitive. In fact, perhaps surprisingly, only in 5 of 19 languages a "POS-identified" system performed better than all of the "POS-agnostic" ones (see Table 10.8).

10.8.3 Multilingual Semi-Supervised Parsers

The final broad class of related algorithms considered here extends beyond monolingual data and uses both identities of POS-tags and/or parallel bitexts to transfer (supervised) delexicalized parsers across languages. Parser projection is by far the most successful approach to date (and it too may stand to gain from this chapter's modeling improvements). Of the 10 languages for which results could be found in the literature, transferred parsers underperformed the grammar inducers in only one case: on English (see Table 10.8). The unsupervised system that performed better used a special "weighted" initializer (Ch. 4) that worked well for English (but less so for many other languages). DBMs may be able to improve initialization. For example, modeling of incomplete sentences could help in incremental initialization strategies like *baby steps* (Ch. 3), which are likely sensitive to the proverbial "bum steer" from unrepresentative short fragments, *pace* Tu and Honavar [323].

10.8.4 Miscellaneous Systems on Short Sentences

Several recent systems [64, 297, 228, 117, 25, *inter alia*] are absent from Table 10.8 because they do not report performance for all sentence lengths. To facilitate comparison with this body of important previous work, final accuracies for the "up-to-ten words" task were also tabulated, under heading *@10*: 51.9%, on average.

10.9 Conclusion

Although a dependency parse for a sentence can be mapped to a constituency parse [337], the probabilistic models generating them use different conditioning: dependency grammars focus on the relationship between arguments and heads, constituency grammars on the coherence of chunks covered by non-terminals. Since redundant views of data can make learning easier [32], integrating aspects of both constituency and dependency ought to be able to help grammar induction. This chapter showed that this insight is correct: dependency grammar inducers can gain from modeling boundary information that is fundamental to constituency (i.e., phrase-structure) formalisms. DBMs are a step in the direction towards modeling constituent boundaries jointly with head dependencies. Further steps must involve more tightly coupling the two frameworks, as well as showing ways to incorporate both kinds of information in other state-of-the art grammar induction paradigms.

Chapter 11

Reduced Models

The purpose of this chapter is to explore strategies that capitalize on the advantages of DBMs, which track the words at the fringes, as well as sentence completeness status, by feeding them more and simpler implicitly constrained data (text fragments chopped up at punctuation boundaries), as well as modeling simplifications that are well suited to bootstrapping from such artificial input snippets. Supporting peer-reviewed publication is *Bootstrapping Dependency Grammar Inducers from Incomplete Sentence Fragments via Austere Models* in ICGI 2012 [305].

11.1 Introduction

"Starting small" strategies [95] that gradually increase complexities of training models [178, 40, 107, 119] and/or input data [37, 22, 175, 323] have long been known to aid various aspects of language learning. In dependency grammar induction, pre-training on sentences up to length 15 before moving on to full data can be particularly effective (Chs. 4, 6, 7, 9). Focusing on short inputs first yields many benefits: faster training, better chances of guessing larger fractions of correct parse trees, and a preference for more local structures, to name a few. But there are also drawbacks: notably, unwanted biases, since many short sentences are not representative, and data sparsity, since most typical complete sentences can be quite long.

This chapter proposes starting with short inter-punctuation fragments of sentences, rather than with small whole inputs exclusively. Splitting text on punctuation allows more and simpler word sequences to be incorporated earlier in training, alleviating data sparsity and complexity concerns. Many of the resulting fragments will be phrases and clauses (see Ch. 7), since punctuation correlates with constituent boundaries [253, 254], and may not fully exhibit sentence structure. Nevertheless, these and other unrepresentative short inputs can be accommodated using dependency-and-boundary models (DBMs), which distinguish complete sentences from incomplete fragments (Ch. 10).

DBMs consist of overlapping grammars that share all information about head-dependent interactions, while modeling sentence root propensities and head word fertilities separately, for different types of input. Consequently, they can glean generalizable insights about local substructures from incomplete fragments without allowing their unrepresentative lengths and root word distributions to corrupt grammars of complete sentences. In addition, chopping up data plays into other strengths of DBMs — which learn from phrase boundaries, such as the first and last words of sentences — by increasing the number of visible edges.

11.2 Methodology

All of the experiments in this chapter make use of DBMs, which are head-outward [7] class-based models, to generate projective dependency parse trees for WSJ. They begin by choosing a class for the root word (c_r). Remainders of parse structures, if any, are produced recursively. Each node spawns off ever more distant left dependents by (i) deciding whether to have more children, conditioned on direction (left), the class of the (leftmost) fringe word in the partial parse (initially, itself), and other parameters (such as adjacency of the would-be child); then (ii) choosing its child's category, based on direction, the head's own class, etc. Right dependents are generated analogously, but using separate factors. Unlike traditional head-outward models, DBMs condition their generative process on more observable state: left and right end words of phrases being constructed. Since left and right child sequences are still generated independently, DBM grammars are split-head.

DBM-2 maintains two related grammars: one for complete sentences ($comp = \texttt{T}$), approximated by presence of final punctuation, and another for incomplete fragments. These grammars communicate through shared estimates of word attachment parameters, making

	Stage I	*Stage II*	*DDA*	*TA*
Baseline (§11.2)	DBM-2	constrained DBM-3	59.7	3.4
Experiment #1 (§11.3)	split DBM-2	constrained DBM-3	60.2	3.5
Experiment #2 (§11.4)	split DBM-i	constrained DBM-3	60.5	4.9
Experiment #3 (§11.5)	split DBM-0	constrained DBM-3	61.2	5.0
Unsupervised Tags (Ch. 9)	constrained DMV	constrained L-DMV	59.1	—

Table 11.1: Directed dependency and exact tree accuracies (DDA / TA) for the baseline, experiments with split data, and previous state-of-the-art on Section 23 of WSJ.

it possible to learn from mixtures of input types without polluting root and stopping factors. DBM-3 conditions attachments on additional context, distinguishing arcs that cross punctuation boundaries ($cross = \texttt{T}$) from lower-level dependencies. Only heads of fragments are allowed to attach other fragments as part of (*loose*) constrained Viterbi EM; in inference, entire fragments could be attached by arbitrary external words (*sprawl*).

All missing families of factors (e.g., those of punctuation-crossing arcs) are initialized as uniform multinomials. Instead of gold parts-of-speech, context-sensitive unsupervised tags are now used, obtained by relaxing a hard clustering produced by Clark's [62] algorithm using an HMM [123]. As in the original set-up without gold tags (Ch. 9), training is split into two stages of Viterbi EM: first on shorter inputs (15 or fewer tokens), then on most sentences (up to length 45). The baseline system learns DBM-2 in Stage I and DBM-3 (with punctuation-induced constraints) in Stage II, starting from uniform punctuation-crossing attachment probabilities. Smoothing and termination of both stages are as in Stage I of the original system. This strong baseline achieves 59.7% directed dependency accuracy — somewhat higher than the previous state-of-the-art result (59.1%, obtained with the DMV — see also Table 11.1). All experiments make changes to Stage I's training only, initialized from the same exact trees as in the baselines and affecting Stage II only via its initial trees.

11.3 Experiment #1 (DBM-2): Learning from Fragmented Data

Punctuation can be viewed as implicit partial bracketing constraints [245]: assuming that some (head) word from each inter-punctuation fragment derives the entire fragment is a useful approximation in the unsupervised setting (Ch. 7). With this restriction, splitting text at punctuation is equivalent to learning partial parse forests — partial because longer fragments are left unparsed, and forests because even the parsed fragments are left unconnected [224]. Grammar inducers are allowed to focus on modeling lower-level substructures first,[1] before forcing them to learn how these pieces may fit together. Deferring decisions associated with potentially long-distance inter-fragment relations and dependency arcs from longer fragments to a later training stage is thus a variation on the "easy-first" strategy [124], which is a fast and powerful heuristic from the supervised dependency parsing setting.

DBM-2 will now be bootstrapped using snippets of text obtained by slicing up all input sentences at punctuation. Splitting data increased the number of training tokens from 163,715 to 709,215 (and effective short training inputs from 15,922 to 34,856). Ordinarily, tree generation would be conditioned on an exogenous sentence-completeness status ($comp$), using presence of sentence-final punctuation as a binary proxy. This chapter refines this notion to account for new kinds of fragments: (i) for the purposes of modeling roots, only unsplit sentences could remain complete; as for stopping decisions, (ii) leftmost fragments (prefixes of complete original sentences) are left-complete; and, analogously, (iii) rightmost fragments (suffixes) retain their status vis-à-vis right stopping decisions (see Figure 11.1). With this set-up, performance improved from 59.7 to 60.2% (from 3.4 to 3.5% for exact trees — see Table 11.1).

Next, let's make better use of the additional fragmented training data.

[1] About which the *loose* and *sprawl* punctuation-induced constraints agree (Ch. 7).

Odds and Ends

(a) An incomplete fragment.

“It happens.”

(b) A complete sentence that cannot be split on punctuation.

Bach’s “Air” followed.

(c) A complete sentence that can be split into three fragments.

Figure 11.1: Three types of input: (a) fragments lacking sentence-final punctuation are always considered incomplete; (b) sentences with trailing but no internal punctuation are considered complete though unsplittable; and (c) text that can be split on punctuation yields several smaller incomplete fragments, e.g., *Bach’s*, *Air* and *followed*. In modeling stopping decisions, *Bach’s* is still considered left-complete — and *followed* right-complete — since the original input sentence was complete.

11.4 Experiment #2 (DBM-*i*): Learning with a Coarse Model

In modeling head word fertilities, DBMs distinguish between the adjacent case ($adj = \texttt{T}$, deciding whether or not to have any children in a given direction, $dir \in \{\texttt{L}, \texttt{R}\}$) and non-adjacent cases ($adj = \texttt{F}$, whether to cease spawning additional daughters — see $\mathbb{P}_{\texttt{STOP}}$ in Table 11.2). This level of detail can be wasteful for short fragments, however, since non-adjacency will be exceedingly rare there: most words will not have many children. Therefore, a model can be reduced by eliding adjacency. On the down side, this leads to some loss of expressive power; but on the up side, pooled information about phrase edges could flow more easily inwards from input boundaries, since it will not be quite so needlessly subcategorized.

DBM-i is implemented by conditioning all stopping decisions only on the direction in which a head word is growing, the input’s completeness status in that direction and the identity of the head’s farthest descendant on that side (the head word itself, in the adjacent case — see Tables 11.2 and 11.7). With this smaller initial model, directed dependency accuracy on the test set improved only slightly, from 60.2 to 60.5%; however, performance

Model		$\mathbb{P}_{\texttt{ATTACH}}$ *(root-head)*	$\mathbb{P}_{\texttt{ATTACH}}$ *(head-dependent)*	$\mathbb{P}_{\texttt{STOP}}$ *(adjacent/not)*
DBM-3	(Ch. 10)	$(\diamond, \texttt{L}, c_r, comp_{root})$	$(c_h, dir, c_d, cross)$	$(comp_{dir}, c_e, dir, adj)$
DBM-2	(§11.3, Ch. 10)	$(\diamond, \texttt{L}, c_r, comp_{root})$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir, adj)$
DBM-i	(§11.4, 11.7)	$(\diamond, \texttt{L}, c_r, comp_{root})$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir)$
DBM-0	(§11.5, 11.7)	$(\diamond, \texttt{L}, c_r)$ iff $comp_{root}$	(c_h, dir, c_d)	$(comp_{dir}, c_e, dir)$

Table 11.2: Feature-sets parameterizing dependency-and-boundary models three, two, i and zero: if $comp$ is false, then so are $comp_{root}$ and both of $comp_{dir}$; otherwise, $comp_{root}$ is true for unsplit inputs, $comp_{dir}$ for prefixes (if $dir = \texttt{L}$) and suffixes (when $dir = \texttt{R}$).

at the granularities of whole trees increased dramatically, from 3.5 to 4.9% (see Table 11.1).

11.5 Experiment #3 (DBM-0): Learning with an Ablated Model

DBM-i maintains separate root distributions for complete and incomplete sentences (see $\mathbb{P}_{\texttt{ATTACH}}$ for $\diamond$ in Table 11.2), which can isolate verb and modal types heading typical sentences from the various noun types deriving captions, headlines, titles and other fragments that tend to be common in news-style data. Heads of inter-punctuation fragments are less homogeneous than actual sentence roots, however. Therefore, the learning task can be simplified by approximating what would be a high-entropy distribution with a uniform multinomial, which is equivalent to updating DBM-i via a “partial” EM variant [229].

DBM-0 is implemented by modifying DBM-i to hardwire the root probabilities as one over the number of word classes (1/200, in this case), for all incomplete inputs. With this more compact, asymmetric model, directed dependency accuracy improved substantially, from 60.5 to 61.2% (though only slightly for exact trees, from 4.9 to 5.0% — see Table 11.1).

11.6 Conclusion

This chapter presented an effective divide-and-conquer strategy for bootstrapping grammar inducers. Its procedure is simple and efficient, achieving state-of-the-art results on a standard English dependency grammar induction task by simultaneously scaffolding on both model and data complexity, using a greatly simplified DBM with inter-punctuation fragments of sentences. Future work could explore inducing structure from sentence prefixes and suffixes — or even bootstrapping from intermediate n-grams, perhaps via novel parsing models that may be better equipped for handling distituent fragments.

11.7 Appendix on Partial DBMs

Since dependency structures are trees, few heads get to spawn multiple dependents on the same side. High fertilities are especially rare in short fragments, inviting economical models whose stopping parameters can be lumped together (because in adjacent cases heads and fringe words coincide: $adj = \texttt{T} \rightarrow h = e$, hence $c_h = c_e$). Eliminating inessential components, such as the likely-heterogeneous root factors of incomplete inputs, can also yield benefits.

Consider the sentence ⓐ ⓩ. It admits two structures: $\underline{\textcircled{a}}\curvearrowright\textcircled{z}$ and $\textcircled{a}\curvearrowleft\underline{\textcircled{z}}$. In theory, neither should be preferred. In practice, if the first parse occurs $100p\%$ of the time, a multi-component model could re-estimate total probability as $p^n + (1-p)^n$, where n may exceed its number of independent components. Only root and adjacent stopping factors are non-deterministic here: $\mathbb{P}_{\texttt{ROOT}}(\textcircled{a}) = \mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{L}) = p$ and $\mathbb{P}_{\texttt{ROOT}}(\textcircled{z}) = \mathbb{P}_{\texttt{STOP}}(\textcircled{a}, \texttt{R}) = 1-p$; attachments are fixed (ⓐ can only attach ⓩ and vice-versa). Tree probabilities are thus cubes ($n = 3$): a root and two stopping factors (one for each word, on different sides),

$$\begin{aligned}
\mathbb{P}(\textcircled{a}\ \textcircled{z}) &= \mathbb{P}(\underline{\textcircled{a}}\curvearrowright\textcircled{z}) + \mathbb{P}(\textcircled{a}\curvearrowleft\underline{\textcircled{z}}) \\
&= \overbrace{\mathbb{P}_{\texttt{ROOT}}(\textcircled{a})}^{p}\underbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{a}, \texttt{L})}_{1}\overbrace{(1-\mathbb{P}_{\texttt{STOP}}(\textcircled{a}, \texttt{R}))}^{p}\underbrace{\mathbb{P}_{\texttt{ATTACH}}(\textcircled{a}, \texttt{R}, \textcircled{z})}_{1}\overbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{L})}^{p}\underbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{R})}_{1} \\
&+ \overbrace{\mathbb{P}_{\texttt{ROOT}}(\textcircled{z})}^{1-p}\underbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{R})}_{1}\overbrace{(1-\mathbb{P}_{\texttt{STOP}}(\textcircled{z}, \texttt{L}))}^{1-p}\underbrace{\mathbb{P}_{\texttt{ATTACH}}(\textcircled{z}, \texttt{L}, \textcircled{a})}_{1}\overbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{a}, \texttt{R})}^{1-p}\underbrace{\mathbb{P}_{\texttt{STOP}}(\textcircled{a}, \texttt{L})}_{1} \\
&= p^3 + (1-p)^3.
\end{aligned}$$

For $p \in [0, 1]$ and $n \in \mathbb{Z}^+$, $p^n + (1-p)^n \leq 1$, with strict inequality if $p \notin \{0, 1\}$ and

$n > 1$. Clearly, as n grows above one, optimizers will more strongly prefer extreme solutions $p \in \{0, 1\}$, despite lacking evidence in the data. Since the exponent n is related to numbers of input words and independent modeling components, a recipe of short inputs — combined with simpler, partial models — could help alleviate some of this pressure towards arbitrary determinism.

Part IV

Complete System

Chapter 12

An Integrated Training Strategy

The purpose of this chapter is to integrate the insights from all previous parts — i.e., (i) incremental learning and multiple objectives, (ii) punctuation-induced constraints, and (iii) staged training with scaffolding on both input data and dependency-and-boundary model complexity — into a unified grammar induction pipeline. Supporting peer-reviewed publication is *Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction* in EMNLP 2013 [308], which won the "best paper" award.

12.1 Introduction

Statistical methods for grammar induction often boil down to solving non-convex optimization problems. Early work attempted to locally maximize the likelihood of a corpus, using EM to estimate probabilities of dependency arcs between word bigrams [244, 243]. Paskin's parsing model has since been extended to make unsupervised learning more feasible [172, 133]. But even the latest techniques (Chs. 10–11) can be quite error-prone and sensitive to initialization, because of approximate, local search.

In theory, global optima can be found by enumerating all parse forests that derive a corpus, though this is usually prohibitively expensive in practice. A preferable brute force approach is sampling, as in Markov-chain Monte Carlo (MCMC) and random restarts [144], which hit exact solutions eventually. Restarts can be giant steps in a parameter space that undo all previous work. At the other extreme, MCMC may cling to a neighborhood, rejecting most proposed moves that would escape a local attractor. Sampling methods thus take unbounded time to solve a problem (and can't certify optimality) but are useful for finding approximate solutions to grammar induction [68, 202, 227].

This chapter proposes an alternative (deterministic) search heuristic that combines local optimization via EM with *non*-random restarts. Its new starting places are informed by previously found solutions, unlike conventional restarts, but may not resemble their predecessors, unlike typical MCMC moves. One good way to construct such steps in a parameter space is by forgetting some aspects of a learned model. Another is by merging promising solutions, since even simple interpolation [150] of local optima may be superior to all of the originals. Informed restarts can make it possible to explore a combinatorial search space more rapidly and thoroughly than with traditional methods alone.

12.2 Abstract Operators

Let C be a collection of counts — the sufficient statistics from which a candidate solution to an optimization problem could be computed, e.g., by smoothing and normalizing to yield probabilities. The counts may be fractional and solutions could take the form of multinomial distributions. A local optimizer L will convert C into $C^* = L_{\mathcal{D}}(C)$ — an updated collection of counts, resulting in a probabilistic model that is no less (and hopefully more) consistent with a data set $\mathcal{D}$ than the original C:

$$\begin{tikzcd} C \arrow[r] & \boxed{L_{\mathcal{D}}} \arrow[r] & C^* \end{tikzcd} \tag{1}$$

Unless C^* is a global optimum, it should be possible to make further improvements. But if L is idempotent (and ran to convergence) then $L(L(C)) = L(C)$. Given only C and $L_{\mathcal{D}}$, the single-node optimization network above would be the minimal search pattern worth considering. However, if another optimizer L' — or a fresh starting point C' — were available, then more complicated networks could become useful.

12.2.1 Transforms (Unary)

New starts could be chosen by perturbing an existing solution, as in MCMC, or independently of previous results, as in random restarts. The chapter's focus is on intermediate changes to C, without injecting randomness. All of its transforms involve selective forgetting or filtering. For example, if the probabilistic model that is being estimated decomposes into independent constituents (e.g., several multinomials) then a subset of them can be reset to uniform distributions, by discarding associated counts from C. In text classification, this could correspond to eliminating frequent or rare tokens from bags-of-words. Circular shapes will be used to represent such model ablation operators:

$$\begin{tikzcd} C \arrow[r, "\bigcirc" description] & {} \end{tikzcd} \tag{2}$$

An orthogonal approach might separate out various counts in C by their provenance. For instance, if $\mathcal{D}$ consisted of several heterogeneous data sources, then the counts from some of them could be ignored: a classifier might be estimated from just news text. Squares will be used to represent data-set filtering:

$$\begin{tikzcd} C \arrow[r, "\square" description] & {} \end{tikzcd} \tag{3}$$

Finally, if C represents a mixture of possible interpretations over $\mathcal{D}$ — e.g., because it captures the output of a "soft" EM algorithm — contributions from less likely, noisier completions could also be suppressed (and their weights redistributed to the more likely ones), as in "hard" EM. Diamonds will represent plain (single) steps of Viterbi training:

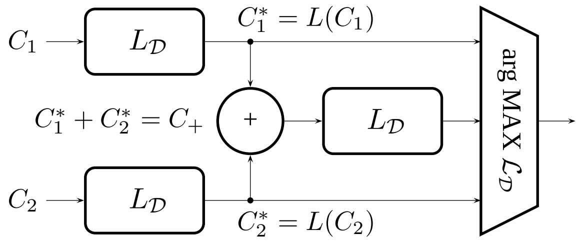
$$\begin{tikzcd} C \arrow[r, "\diamond" description] & {} \end{tikzcd} \tag{4}$$

12.2.2 Joins (Binary)

Starting from different initializers, say C_1 and C_2, it may be possible for L to arrive at distinct local optima, $C_1^* \neq C_2^*$. The better of the two solutions, according to likelihood $\mathcal{L}_\mathcal{D}$ of $\mathcal{D}$, could then be selected — as is standard practice when sampling.

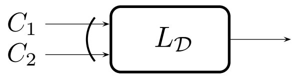
The joining techniques presented in this chapter could do better than either C_1^* or C_2^*, by entertaining also a third possibility, which combines the two candidates. A mixture model can be constructed by adding together all counts from C_1^* and C_2^* into $C_+ = C_1^* + C_2^*$.

Original initializers C_1, C_2 will, this way, have equal pull on the merged model,[1] regardless of nominal size (because C_1^*, C_2^* will have converged using a shared training set, $\mathcal{D}$). The best of C_1^*, C_2^* and $C_+^* = L(C_+)$ can then be returned. This approach may uncover more (and never returns less) likely solutions than choosing among C_1^*, C_2^* alone:



(5)

A short-hand notation will be used to represent the combiner network diagrammed above, less clutter:



(6)

12.3 The Task and Methodology

The transform and join paradigms will now be applied to grammar induction, since it is an important problem of computational linguistics that involves notoriously difficult objectives [245, 82, 119, *inter alia*]. The goal is to induce grammars capable of parsing unseen text. Input, in both training and testing, is a sequence of tokens labeled as: (i) a lexical item and its category, (w, c_w); (ii) a punctuation mark; or (iii) a sentence boundary. Output is unlabeled dependency trees.

12.3.1 Models and Data

All parse structures will be constrained to be projective, via DBMs (Chs. 10–11): DBMs 0 through 3 are head-outward generative parsing models [7] that distinguish complete sentences from incomplete fragments in a corpus $\mathcal{D}$: $\mathcal{D}_{\text{comp}}$ comprises inputs ending with

[1]If desired, a scaling factor could be used to bias C_+ towards either C_i^*, e.g., based on a likelihood ratio.

punctuation; $\mathcal{D}_{\text{frag}} = \mathcal{D} - \mathcal{D}_{\text{comp}}$ is everything else. The "complete" subset is further partitioned into simple sentences, $\mathcal{D}_{\text{simp}} \subseteq \mathcal{D}_{\text{comp}}$, with no internal punctuation, and others, which may be complex. As an example, consider the beginning of an article from (simple) Wikipedia: (i) ***Linguistics*** (ii) *Linguistics (sometimes called philology) is the science that studies language.* (iii) *Scientists who study language are called linguists.* Since the title does not end with punctuation, it would be relegated to $\mathcal{D}_{\text{frag}}$. But two complete sentences would be in $\mathcal{D}_{\text{comp}}$, with the last also filed under $\mathcal{D}_{\text{simp}}$, as it has only a trailing punctuation mark. Previous chapters suggested two curriculum learning strategies: (i) one in which induction begins with clean, simple data, $\mathcal{D}_{\text{simp}}$, and a basic model, DBM-1 (Ch. 10); and (ii) an alternative bootstrapping approach: starting with still more, simpler data — namely, short inter-punctuation fragments up to length $l = 15$, $\mathcal{D}^{l}_{\text{split}} \supseteq \mathcal{D}^{l}_{\text{simp}}$ — and a bare-bones model, DBM-0 (Ch. 11). In this example, $\mathcal{D}_{\text{split}}$ would hold five text snippets: (i) *Linguistics*; (ii) *Linguistics*; (iii) *sometimes called philology*; (iv) *is the science that studies language*; and (v) *Scientists who study language are called linguists.* Only the last piece of text would still be considered complete, isolating its contribution to sentence root and boundary word distributions from those of incomplete fragments. The sparser model, DBM-0, assumes a uniform distribution for roots of incomplete inputs and reduces conditioning contexts of stopping probabilities, which works well with split data. Both DBM-0 and the full DBM,[2] will be exploited, drawing also on split, simple and raw views of input text. All experiments prior to final multilingual evaluation will use WSJ as the underlying tokenized and sentence-broken corpus $\mathcal{D}$. Instead of gold parts-of-speech, 200 context-sensitive unsupervised tags (from Ch. 9) will be plugged in for the word categories.

[2]This chapter will use the short-hand DBM to refer to DBM-3, which is equivalent to DBM-2 if $\mathcal{D}$ has no internally-punctuated sentences (i.e., $\mathcal{D} = \mathcal{D}_{\text{split}}$), and DBM-1 if all inputs also have trailing punctuation (i.e., $\mathcal{D} = \mathcal{D}_{\text{simp}}$); DBM$_0$ will be the short-hand for DBM-0.

12.3.2 Smoothing and Lexicalization

All unlexicalized instances of DBMs will be estimated with "add one" (a.k.a. Laplace) smoothing, using only the word category c_w to represent a token. Fully-lexicalized grammars (L-DBM) are left unsmoothed, and represent each token as both a word and its category, i.e., the whole pair (w, c_w). To evaluate a lexicalized parsing model, a delexicalized-and-smoothed instance will always be obtained first.

12.3.3 Optimization and Viterbi Decoding

This chapter uses "early-switching lateen" EM (Ch. 5) to train unlexicalized models, alternating between the objectives of ordinary (soft) and hard EM algorithms, until neither can improve its own objective without harming the other's. This approach does not require tuning termination thresholds, allowing optimizers to run to numerical convergence if necessary, and will handle only the shorter inputs ($l \leq 15$), starting with soft EM ($L =$ SL, for "soft lateen"). Lexicalized models will cover full data ($l \leq 45$) and employ "early-stopping lateen" EM, re-estimating via hard EM until soft EM's objective suffers. Alternating EMs would be expensive here, since updates take (at least) $O(l^3)$ time, and hard EM's objective ($L =$ H) is the one better suited to long inputs (see Ch. 4).

The decoders will always force an inter-punctuation fragment to derive itself (as in Ch. 7). In evaluation, such (*loose*) constraints may help attach *sometimes* and *philology* to *called* (and *the science...* to *is*). In training, stronger (*strict*) constraints also disallow attachment of fragments' heads by non-heads, to connect *Linguistics*, *called* and *is* (assuming each piece got parsed correctly), though constraints will not impact training with shorter inputs, since there is no internal punctuation in $\mathcal{D}_{\text{split}}$ or $\mathcal{D}_{\text{simp}}$.

12.4 Concrete Operators

Let's now instantiate the operators sketched out in §12.2 specifically for the grammar induction task. Throughout, single steps of Viterbi training will be repeated employed to transfer information between subnetworks in a model-independent way: when a module's output is a set of (Viterbi) parse trees, it necessarily contains sufficient information required to estimate an arbitrarily-factored model down-stream.[3]

12.4.1 Transform #1: A Simple Filter

Given a model that was estimated from (and therefore parses) a data set $\mathcal{D}$, the simple filter (F) attempts to extract a cleaner model, based on the simpler complete sentences of $\mathcal{D}_{\text{simp}}$. It is implemented as a single (unlexicalized) step of Viterbi training:

$$C \longrightarrow \boxed{F} \diamond \longrightarrow \tag{7}$$

The idea here is to focus on sentences that are not too complicated yet grammatical. This punctuation-sensitive heuristic may steer a learner towards easy but representative training text and has been shown to aid grammar induction (see Ch. 10).

12.4.2 Transform #2: A Symmetrizer

The symmetrizer (S) reduces input models to sets of word association scores. It blurs all details of induced parses in a data set $\mathcal{D}$, except the number of times each (ordered) word pair participates in a dependency relation. Symmetrization is also implemented as a single unlexicalized Viterbi training step, but now with proposed parse trees' scores, for a sentence in $\mathcal{D}$, proportional to a product over non-root dependency arcs of one plus how often the left and right tokens (are expected to) appear connected:

$$C \longrightarrow \textcircled{S} \diamond \longrightarrow \tag{8}$$

The idea behind the symmetrizer is to glean information from skeleton parses. Grammar inducers can sometimes make good progress in resolving undirected parse structures despite being wrong about the polarities of most arcs (see Figure 3.2 b: Uninformed). Symmetrization offers an extra chance to make heads or tails of syntactic relations, after learning which words tend to go together.

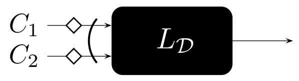
At each instance where a word ⓐ attaches ⓩ on (say) the right, this implementation attributes half its weight to the intended construction, ⓐ$\frown$ⓩ, reserving the other half for the

[3]Klein and Manning [172, §3] advocated a related approach: initializing EM training with an M-step.

symmetric structure, ⓩ attaching ⓐ to its left: $\overset{\curvearrowleft}{ⓐ\,ⓩ}$. For the desired effect, these aggregated counts are left unnormalized, while all other counts (of word fertilities and sentence roots) get discarded. To see why word attachment scores aren't turned into probabilities, consider sentences ⓐ ⓩ and ⓒ ⓩ. The fact that ⓩ co-occurs with ⓐ introduces an asymmetry into ⓩ's relation with ⓒ: $\mathbb{P}(ⓩ \mid ⓒ) = 1$ differs from $\mathbb{P}(ⓒ \mid ⓩ) = 1/2$. Normalizing might force the interpretation $\overset{\curvearrowright}{ⓒ\,ⓩ}$ (and also $\overset{\curvearrowright}{ⓐ\,ⓩ}$), not because there is evidence in the data, but as a side-effect of a model's head-driven nature (i.e., factored with dependents conditioned on heads). Always branching right would be a mistake, however, for example if ⓩ is a noun, since either of ⓐ or ⓒ could be a determiner, with the other a verb.

12.4.3 Join: A Combiner

The combiner must admit arbitrary inputs, including models not estimated from $\mathcal{D}$, unlike the transforms. Consequently, as a preliminary step, each input C_i is converted into parse trees of $\mathcal{D}$, with counts C_i', via Viterbi-decoding with a smoothed, unlexicalized version of the corresponding incoming model. Actual combination is then performed in a more precise (unsmoothed) fashion: C_i^* are the (lexicalized) solutions starting from C_i'; and C_+^* is initialized with their sum, $\sum_i C_i^*$. Counts of the lexicalized model with lowest cross-entropy on $\mathcal{D}$ become the output:[4]



(9)

12.5 Basic Networks

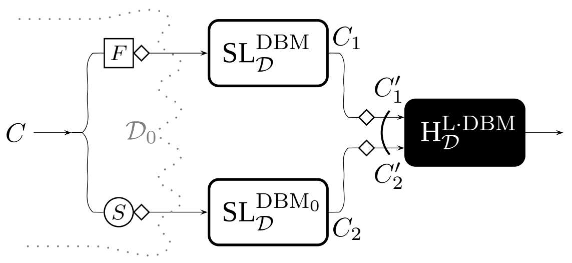
Let's now propose a non-trivial subnetwork for grammar induction, based on the transform and join operators, which will be reused in larger networks.

[4]In this chapter's diagrams, lexicalized modules are shaded black.

12.5.1 Fork/Join (FJ)

Given a model that parses a base data set $\mathcal{D}_0$, the fork/join subnetwork will output an adaptation of that model for $\mathcal{D}$. It could facilitate a grammar induction process, e.g., by advancing it from smaller to larger — or possibly more complex — data sets.

First, two variations of the incoming model, based on $\mathcal{D}_0$, are forked off: (i) a filtered view, which focuses on cleaner, simpler data (transform #1); and (ii) a symmetrized view that backs off to word associations (transform #2). Next is grammar induction over $\mathcal{D}$. A full DBM instance is optimized starting from the first fork, and a reduced DBM$_0$ is bootstrapped from the second. Finally, the two new induced sets of parse trees, for $\mathcal{D}$, are merged (lexicalized join):


C
F
S
$\mathcal{D}_0$
$\text{SL}_{\mathcal{D}}^{\text{DBM}}$
$\text{SL}_{\mathcal{D}}^{\text{DBM}_0}$
C_1
C_2
C'_1
C'_2
$\text{H}_{\mathcal{D}}^{\text{L}\cdot\text{DBM}}$
(10)

The idea here is to prepare for two scenarios: an incoming grammar that is either good or bad for $\mathcal{D}$. If the model is good, DBM should be able to hang on to it and make improvements. But if it is bad, DBM could get stuck fitting noise, whereas DBM$_0$ might be more likely to ramp up to a good alternative. Since it is impossible to know ahead of time which is the true case, both optimization paths are pursued simultaneously, allowing a combiner to make the decision, later.

Note that the forks start (and end) optimizing with soft EM. This is because soft EM integrates previously unseen tokens into new grammars better than hard EM, as evidenced by the failed attempt to reproduce the “baby steps” strategy with Viterbi training (see Figure 4.2b). A combiner then executes hard EM, and since outputs of transforms are trees, the end-to-end process is a chain of lateen alternations that starts and ends with hard EM.

A “grammar inductor” will be used to represent subnetworks that transition from $\mathcal{D}^{l}_{\text{split}}$ to $\mathcal{D}^{l+1}_{\text{split}}$, by taking transformed parse trees of inter-punctuation fragments up to length l (base data set, $\mathcal{D}_0$) to initialize training over fragments up to length $l + 1$:

$$\begin{tikzcd} C \arrow[r, squiggly, "l+1"] & {} \end{tikzcd} \tag{11}$$

The FJ network instantiates a grammar inductor with $l = 14$, thus training on inter-punctuation fragments up to length 15, as in previous work, starting from an empty set of counts, $C = \emptyset$. Smoothing causes initial parse trees to be chosen uniformly at random, as also suggested by Cohen and Smith [67]:

$$\begin{tikzcd} \emptyset \arrow[r, squiggly, "15"] & {} \end{tikzcd} \tag{12}$$

12.5.2 Iterated Fork/Join (IFJ)

The second basic network daisy-chains grammar inductors, starting from the single-word inter-punctuation fragments in $\mathcal{D}^1_{\text{split}}$, then retraining on $\mathcal{D}^2_{\text{split}}$, and so forth, until finally stopping at $\mathcal{D}^{15}_{\text{split}}$, as before:

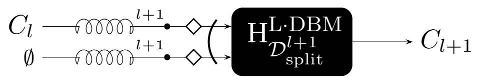
$$\begin{tikzcd} {} \arrow[r, squiggly, "1", maps to] & \bullet \arrow[r, squiggly, "2"] & \bullet \arrow[r, dotted, dash] & {} \arrow[r, "14"] & \bullet \arrow[r, squiggly, "15"] & {} \end{tikzcd} \tag{13}$$

This system is diagrammed as not taking an input, since the first inductor's output is fully determined by unique parse trees of single-token strings. This iterative approach to optimization is akin to deterministic annealing [268], and is patterned after "baby steps" (Ch. 3).

Unlike the basic FJ, where symmetrization was a no-op (since there were no counts in $C = \emptyset$), IFJ makes use of symmetrizers — e.g., in the third inductor, whose input is based on strings with up to two tokens. Although it should be easy to learn words that go together from very short fragments, extracting correct polarities of their relations could be a challenge: to a large extent, outputs of early inductors may be artifacts of how the generative models factor (see §4.2) or how ties are broken in optimization (see Appendix of Ch. 11). One might therefore expect symmetrization to be crucial in earlier stages but to weaken any high quality grammars, nearer the end; it will be up to combiners to handle any such phase transitions correctly (or gracefully).

12.5.3 Grounded Iterated Fork/Join (GIFJ)

So far, basic networks have been either purely iterative (IFJ) or static (FJ). These two approaches can also be combined, by injecting FJ's solutions into IFJ's more dynamic stream. The new transition subnetwork will join outputs of grammar inductors that either (i) continue a previous solution (as in IFJ); or (ii) start over from scratch ("grounding" to an FJ). The full GIFJ network can then be obtained by unrolling the template below from $l = 14$ back to one:


C_l
$l+1$
$\emptyset$
$l+1$
$\text{H}^{\text{L}\cdot\text{DBM}}_{\mathcal{D}^{l+1}_{\text{split}}}$
C_{l+1}
(14)

12.6 Performance of Basic Networks

Basic networks' performance was compared on their final training sets, $\text{WSJ}^{15}_{\text{split}}$ (see Table 12.1, which also tabulates results for a cleaner subset, $\text{WSJ}^{15}_{\text{simp}}$). The first network starts from $C = \emptyset$, helping establish several straw-man baselines. Its empty initializer corresponds to guessing (projective) parse trees uniformly at random, which has 21.4% accuracy and sentence string cross-entropy of 8.76bpt.

12.6.1 Fork/Join (FJ)

FJ's symmetrizer yields random parses of $\text{WSJ}^{14}_{\text{split}}$, which initialize training of DBM_0. This baseline (B) lowers cross-entropy to 6.18bpt and scores 57.0%. FJ's filter starts from parse trees of $\text{WSJ}^{14}_{\text{simp}}$ only, and trains up a full DBM. This choice makes a stronger baseline (A), with 5.89bpt cross-entropy, at 62.2%.

The join operator uses counts from A and B, C_1 and C_2, to obtain parse trees whose own counts C'_1 and C'_2 initialize lexicalized training. From each C'_i, an optimizer arrives at C^*_i. Grammars corresponding to these counts have higher cross-entropies, because of vastly larger vocabularies, but also better accuracies: 59.2 and 62.3%. Their mixture C_+ is a simple sum of counts in C^*_1 and C^*_2: it is not expected to be an improvement but happens to be a good move, resulting in a grammar with higher accuracy (64.0%), though not better Viterbi cross-entropy (7.27 falls between 7.08 and 7.30bpt) than both sources. The combiner's third alternative, a locally optimal C^*_+, is then obtained by re-optimizing from C_+. This solution performs slightly better (64.2%) and will be the local optimum returned by FJ's join operator, because it attains the lowest cross-entropy (7.04bpt).

		$\mathrm{WSJ}^{15}_{\mathrm{split}}$			$\mathrm{WSJ}^{15}_{\mathrm{simp}}$				
Instance Label	*Model*	$\hbar_{\textit{sents}}$	$\hbar_{\textit{trees}}$	*DDA*	$\hbar_{\textit{sents}}$	$\hbar_{\textit{trees}}$	*DDA*	*TA*	*Description*
	DBM	6.54	6.75	83.7	6.05	6.21	85.1	42.7	Supervised (MLE of WSJ^{45})
$\emptyset = C$	—	8.76	10.46	21.4	8.58	10.52	20.7	3.9	Random Projective Parses
$\mathrm{SL}(S(C)) = C_2$	DBM_0	6.18	6.39	57.0	5.90	6.11	57.5	10.4	B } *Unlexicalized*
$\mathrm{SL}(F(C)) = C_1$	DBM	5.89	5.99	62.2	5.79	5.90	60.9	12.0	A } *Baselines*
$\mathrm{H}(C'_2) = C^*_2$	L-DBM	7.28	7.30	59.2	6.87	6.88	58.6	10.4	}
$\mathrm{H}(C'_1) = C^*_1$	L-DBM	7.07	7.08	62.3	6.72	6.73	60.8	12.0	} *Baseline*
$C^*_1 + C^*_2 = C_+$	L-DBM	7.20	7.27	64.0	6.82	6.88	62.5	12.3	} *Combination*
$\mathrm{H}(C_+) = C^*_+$	L-DBM	7.02	7.04	64.2	6.64	6.65	62.7	12.8	Fork/Join }
	L-DBM	6.95	6.96	70.5	6.55	6.56	68.2	14.9	Iterated Fork/Join (IFJ)
	L-DBM	6.91	6.92	71.4	6.52	6.52	69.2	15.6	Grounded Iterated Fork/Join
	L-DBM	6.83	6.83	72.3	6.41	6.41	70.2	17.9	Grammar Transformer (GT)
	L-DBM	6.92	6.93	71.9	6.53	6.53	69.8	16.7	IFJ } *w/Iterated*
	L-DBM	6.83	6.83	72.9	6.41	6.41	70.6	18.0	GT } *Combiners*

Table 12.1: Sentence string and parse tree cross-entropies (in bpt), and accuracies (DDA), on inter-punctuation fragments up to length 15 ($\mathrm{WSJ}^{15}_{\mathrm{split}}$) and its subset of simple, complete sentences ($\mathrm{WSJ}^{15}_{\mathrm{simp}}$, with exact tree accuracies — TA).

12.6.2 Iterated Fork/Join (IFJ)

IFJ's iterative approach results in an improvement: 70.5% accuracy and 6.96bpt cross-entropy. To test how much of this performance could be obtained by a simpler iterated network, several ablated systems that don't fork or join, i.e., the classic "baby steps" schema (chaining together 15 optimizers), were tried, using both DBM and DBM_0, with and without a transform in-between. However, all such "linear" networks scored well below 50%. These results suggest that an ability to branch out into different promising regions of a solution space, and to merge solutions of varying quality into better models, are important properties of FJ subnetworks.

12.6.3 Grounded Iterated Fork/Join (GIFJ)

Grounding improves GIFJ's performance further, to 71.4% accuracy and 6.92bpt cross-entropy. This result shows that fresh perspectives from optimizers that start over can make search efforts more fruitful.

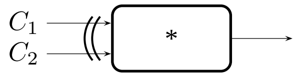
12.7 Enhanced Subnetworks

Modularity and abstraction allow for compact representations of complex systems.[5] Another key benefit is that individual components can be understood and improved in isolation, as will be demonstrated next.

12.7.1 An Iterative Combiner (IC)

The basic combiner introduced a third option, C_+^*, into a pool of candidate solutions, $\{C_1^*, C_2^*\}$. This new entry may not be a simple mixture of the originals, because of non-linear effects from applying L to $C_1^* + C_2^*$, but could most likely still be improved. Rather than stop at C_+^*, when it is better than both originals, one could recombine it with a next best solution, continuing until no further improvement is made. Iterating can't harm a given combiner's cross-entropy (e.g., it lowers FJ's from 7.04 to 7.00bpt), and its advantages can be realized more fully in the larger networks (albeit without any end-to-end guarantees): upgrading all 15 combiners in IFJ would improve performance (slightly) more than grounding (71.5 *vs.* 71.4%), and lower cross-entropy (from 6.96 to 6.93bpt). But even this approach is still a bit timid.

A still more greedy way is to proceed so long as C_+^* is not worse than *both* predecessors. Let's now state this chapter's most general iterative combiner (IC) algorithm: Start with a solution pool $p = \{C_i^*\}_{i=1}^n$. Next, construct p' by adding $C_+^* = L(\sum_{i=1}^n C_i^*)$ to p and removing the worst of $n+1$ candidates in the new set. Finally, if $p = p'$, return the best of the solutions in p; otherwise, repeat from $p := p'$. At $n = 2$, one could think of taking $L(C_1^* + C_2^*)$ as performing a kind of bisection search in some (strange) space. With these new and improved combiners, the IFJ network performs better: 71.9% (up from 70.5 — see Table 12.1), lowering cross-entropy (down from 6.96 to 6.93bpt). A distinguished notation will be used for the ICs:



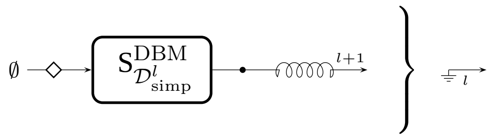
(15)

[5]For instance, the grounded network involves more than one hundred lateen optimizations, not counting individual Viterbi steps: $14 \cdot ((2 \cdot 5) + 3) = 182$.

12.7.2 A Grammar Transformer (GT)

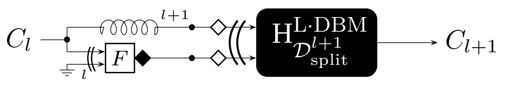
The levels of this chapter's systems' performance at grammar induction thus far suggest that the space of possible networks (say, with up to k components) may itself be worth exploring more thoroughly. This exercise will be left, mostly, to future work, with the exception of two relatively straight-forward extensions for grounded systems.

The static bootstrapping mechanism ("ground" of GIFJ) can be improved by pretraining with simple sentences first — as in the curriculum for learning DBM-1 (Ch. 10), but now with a variable length cut-off l (much lower than the original 45) — instead of starting from $\emptyset$ directly:


$\emptyset$
$\mathrm{S}^{\mathrm{DBM}}_{\mathcal{D}^{l}_{\mathrm{simp}}}$
$l+1$
l

(16)

The output of this subnetwork can then be refined, by reconciling it with a previous dynamic solution. A mini-join of a new ground's counts with C_l will be performed, using the filter transform (single steps of *lexicalized* Viterbi training on clean, simple data), ahead of the main join (over more training data):

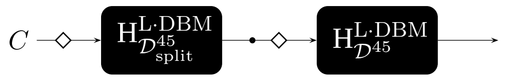

C_l
$l+1$
F
l
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{l+1}_{\mathrm{split}}}$
C_{l+1}

(17)

This template can also be unrolled, as before, to obtain the last network (GT), which achieves 72.9% accuracy and 6.83bpt cross-entropy (slightly less accurate with basic combiners, at 72.3% — see Table 12.1).

12.8 Full Training and System Combination

All systems described so far stop training at $\mathcal{D}^{15}_{\mathrm{split}}$. A two-stage adaptor network will be used to transition their grammars to a full data set, $\mathcal{D}^{45}$:

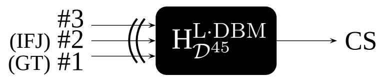

C
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{45}_{\mathrm{split}}}$
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{45}}$

(18)

		System	*DDA*	(@10)
	Concavity and Initialization	[119]	53.1	(64.3)
	Posterior Sparsity	[117]	53.3	(64.3)
	Robust CCGs	[30]	53.3	(71.5)
	Tree Substitution Grammars	[33]	55.7	(67.7)
	Unambiguity Regularization	[324]	57.0	(71.4)
	Punctuation	(Ch. 7)	58.4	(71.4)
	Unsupervised Tags	(Ch. 9)	59.1	(71.4)
#3	Bootstrapping	(Ch. 11)	61.2	(71.4)
#2	*w/Full Training*	IFJ	62.7	(70.3)
#1	*w/Full Training*	GT	63.4	(70.3)
#1 + #2 + #3	**System Combination**	CS	**64.4**	(**72.0**)
	Supervised DBM (also with *loose* decoding)		76.3	(85.4)

Table 12.2: Directed dependency accuracies (DDA) on Section 23 of WSJ (all sentences and up to length ten) for recent systems, our full networks (IFJ and GT), and three-way combination (CS) with the previous state-of-the-art.

The first stage exposes grammar inducers to longer inputs (inter-punctuation fragments with up to 45 tokens); the second stage, at last, reassembles text snippets into actual sentences (also up to $l = 45$).[6] After full training, the IFJ and GT systems parse Section 23 of WSJ at 62.7 and 63.4% accuracy, better than the previous state-of-the-art (61.2% — see Table 12.2). To test the generalized IC algorithm, these three strong grammar induction pipelines were merged into a combined system (CS). It scored highest: 64.4%.



(19)

The quality of bracketings corresponding to (non-trivial) spans derived by heads of dependency structures coming out of the combined system is competitive with the state-of-the-art in unsupervised *constituent* parsing. On the WSJ sentences up to length 40 in Section 23, CS attains similar F_1-measure (54.2 *vs.* 54.6, with higher recall) to PRLG [254], which is

[6]Note that smoothing in the final (unlexicalized) Viterbi step masks the fact that model parts that could not be properly estimated in the first stage (e.g., probabilities of punctuation-crossing arcs) are being initialized to uniform multinomials.

System		F_1	P	R
Binary-Branching	Upper Bound	85.7		
Left-Branching	Baseline	12.0		
CCM	[171]	33.7		
Right-Branching	Baseline	40.7		
F-CCM	[145]	45.1		
HMM	[254]	46.3		
LLCCM	[127]	47.6		
CCL	[283]	52.8	54.6	51.1
PRLG	[254]	**54.6**	**60.4**	49.8
CS	**System Combination**	**54.2**	55.6	**52.8**
Supervised DBM	Skyline	59.3	65.7	54.1
Dependency-Based	Upper Bound	87.2	100	77.3

Table 12.3: Harmonic mean (F_1) of precision (P) and recall (R) for unlabeled constituent bracketings on Section 23 of WSJ (sentences up to length 40) for the combined system (CS), recent state-of-the-art and the baselines.

the strongest system of which I am aware (see Table 12.3).[7]

12.9 Multilingual Evaluation

The final check is to see how this chapter's algorithms generalize outside English WSJ, by testing in 23 more set-ups: all 2006/7 CoNLL test sets. Most recent work evaluates against this multilingual data, though still with the unrealistic assumption of POS tags. But since inducing high quality word clusters for many languages would be beyond the scope of this chapter, here too gold tags are plugged in for word categories (instead of unsupervised tags, as in §12.3–12.8). A comparison will be made to the two strongest systems available during the writing of this chapter:[8] MZ [203] and SAJ (Ch. 10), which report average accuracies

[7]These numbers differ from Ponvert et al.'s [254, Table 6] for the full Section 23 because we restricted their `eval-ps.py` script to a maximum length of 40 words, in our evaluation, to match other previous work: Golland et al.'s [127, Figure 1] for CCM and LLCCM; Huang et al.'s [145, Table 2] for the rest.

[8]Another high-scoring system [201] of possible interest to the reader recently came out, exploiting prior knowledge of stopping probabilities (estimated from large POS-tagged corpora, via reducibility principles).

Directed Dependency Accuracies (DDA)							*(@10)*
CoNLL Data		MZ	SAJ	IFJ	GT	CS	
Arabic	2006	26.5	10.9	**33.3**	8.3	9.3	(30.2)
	'7	27.9	**44.9**	26.1	25.6	26.8	(45.6)
Basque	'7	26.8	**33.3**	23.5	24.2	24.4	(32.8)
Bulgarian	'7	46.0	**65.2**	35.8	64.2	63.4	(69.1)
Catalan	'7	47.0	62.1	65.0	**68.4**	68.0	(79.2)
Chinese	'6	—	**63.2**	56.0	55.8	58.4	(60.8)
	'7	—	**57.0**	49.0	48.6	52.5	(56.0)
Czech	'6	49.5	**55.1**	44.5	43.9	44.0	(52.3)
	'7	48.0	**54.2**	42.9	24.5	34.3	(51.1)
Danish	'6	**38.6**	22.2	37.8	17.1	21.4	(29.8)
Dutch	'6	44.2	46.6	40.8	**51.3**	48.0	(48.7)
English	'7	49.2	29.6	39.3	57.6	**58.2**	(75.0)
German	'6	44.8	39.1	34.1	54.5	**56.2**	(71.2)
Greek	'6	20.2	26.9	23.7	45.0	**45.4**	(52.2)
Hungarian	'7	51.8	58.2	24.8	52.9	**58.3**	(67.6)
Italian	'7	43.3	40.7	**56.8**	31.1	34.9	(44.9)
Japanese	'6	50.8	22.7	32.6	**63.7**	63.0	(68.9)
Portuguese	'6	50.6	72.4	38.0	72.7	**74.5**	(81.1)
Slovenian	'6	18.1	35.2	42.1	50.8	**50.9**	(57.3)
Spanish	'6	51.9	28.2	57.0	**61.7**	61.4	(73.2)
Swedish	'6	48.2	**50.7**	46.6	48.6	49.7	(62.1)
Turkish	'6	—	**34.4**	28.0	32.9	29.2	(33.2)
	'7	15.7	**44.8**	42.1	41.7	37.9	(42.4)
	Average:	40.0	42.9	40.0	47.6	**48.6**	(57.8)

Table 12.4: Blind evaluation on 2006/7 CoNLL test sets (all sentences) for the full networks (IFJ and GT), previous state-of-the-art systems of Mareček and Žabokrtský [203], MZ, and DBMs (from Ch. 10), SAJ, and three-way combination of IFJ, GT and SAJ (CS, including results up to length ten).

of 40.0 and 42.9% for CoNLL data (see Table 12.4). The new fully-trained IFJ and GT systems score 40.0 and 47.6%. As before, combining these networks with an implementation of the best previous state-of-the-art system (SAJ) yields a further improvement, increasing final accuracy to 48.6%.

12.10 Discussion

CoNLL training sets were intended for comparing supervised systems, and aren't all suitable for unsupervised learning: 12 languages have under 10,000 sentences (with Arabic, Basque, Danish, Greek, Italian, Slovenian, Spanish and Turkish particularly small), compared to WSJ's nearly 50,000. In some treebanks sentences are very short (e.g., Chinese and Japanese, which appear to have been split on punctuation), and in others extremely long (e.g., Arabic). Even gold tags aren't always helpful, as their number is rarely ideal for grammar induction (e.g., 42 *vs.* 200 for English). These factors contribute to high variances of this chapter's (and previous) results (see Table 12.4). Nevertheless, looking at the more stable average accuracies reveals a positive trend, moving from a simpler fully-trained system (IFJ, 40.0%), to a more complex system (GT, 47.6%), to system combination (CS, 48.6%). Grounding seems to be more important for the CoNLL sets, possibly because of data sparsity or availability of gold tags.

12.11 Related Work

The surest way to avoid local optima is to craft an objective that doesn't have them. For example, Want et al. [331] demonstrated a convex training method for semi-supervised dependency parsing; Lashkari and Golland [180] introduced a convex reformulation of likelihood functions for clustering tasks; and Corlett and Penn [73] designed a search algorithm for encoding decipherment problems that guarantees to quickly converge on optimal solutions. Convexity can be ideal for comparative analyses, by eliminating dependence on initial conditions. But for many NLP tasks, including grammar induction, the most relevant known objective functions are still riddled with local optima. Renewed efforts to find exact solutions [90, 128] may be a good fit for the smaller and simpler, earlier stages of this chapter's iterative networks.

Multi-start methods [300] can recover certain global extrema almost surely (i.e., with probability approaching one). Moreover, random restarts via uniform probability measures can be optimal, in a worst-case-analysis sense, with parallel processing sometimes leading to exponential speed-ups [144]. This approach is rarely emphasized in NLP literature. For instance, Moore and Quirk [225] demonstrated consistent, substantial gains from random restarts in statistical machine translation (but also suggested better and faster replacements — see below); Ravi and Knight [261, §5, Figure 8] found random restarts for EM to be crucial in parts-of-speech disambiguation. However, other reviews are few and generally negative [168, 205].[9]

Iterated local search methods [142, 153, *inter alia*] escape local basins of attraction by perturbing candidate solutions, without undoing all previous work. "Large-step" moves can come from jittering [137], dithering [256, Ch. 2] or smoothing [27]. Non-improving "sideways" moves offer substantial help with hard satisfiability problems [286]; and injecting non-random noise [285], by introducing "uphill" moves via mixtures of random walks and greedy search strategies, does better than random noise alone or simulated annealing [169]. In NLP, Moore and Quirk's random walks from previous local optima were faster than uniform sampling and also increased BLEU scores; Elsner and Schudy [96] showed that local search can outperform greedy solutions for document clustering and chat disentanglement tasks; and Mei et al. [215] incorporated tabu search [120, 121, Ch. 3] into HMM training for automated speech recognition.

Genetic algorithms are a fusion of what's best in local search and multi-start methods [143], exploiting a problem's structure to combine valid parts of any partial solutions [140, 122]. Evolutionary heuristics proved useful in the induction of phonotactics [21], text planning [217], factored modeling of morphologically-rich languages [88] and plot induction for story generation [214]. Multi-objective genetic algorithms [105] can handle problems with equally important but conflicting criteria [312], using Pareto-optimal ensembles. They are especially well-suited to language, which evolves under pressures from competing (e.g., speaker, listener and learner) constraints, and have been used to model configurations of vowels and tone systems [165].[10] This chapter's transform and join mechanisms also exhibit some features of genetic search, and make use of competing objectives: good sets of parse trees must make sense both lexicalized and with word categories, to rich and impoverished models of grammar, and for both long, complex sentences and short, simple text fragments.

[9] A notable recent exception is the application of a million random restarts to decipherment problems [26].

[10] Following the work on "lateen EM" (Ch. 5), Pareto-optimality has been applied to other multi-metric optimization problems that arise in natural language learning, for example statistical machine translation [276].

This selection of text filters is a specialized case of more general "data perturbation" techniques — even cycling over randomly chosen mini-batches that partition a data set helps avoid some local optima [190]. Elidan et al. [94] suggested how example-reweighing could cause "informed" changes, rather than arbitrary damage, to a hypothesis. Their (adversarial) training scheme guided learning toward improved generalizations, robust against input fluctuations. Language learning has a rich history of reweighing data via (cooperative) "starting small" strategies [95], beginning from simpler or more certain cases. This family of techniques has met with success in semi-supervised named entity classification [72, 341],[11] parts-of-speech induction [61, 62], and language modeling [175, 22], in addition to unsupervised parsing [44, 301, 68, 323].

12.12 Conclusion

This chapter proposed several simple algorithms for combining grammars and showed their usefulness in merging the outputs of iterative and static grammar induction systems. Unlike conventional system combination methods, e.g., in machine translation [338], the ones here do not require incoming models to be of similar quality to make improvements. These properties of the combiners were exploited to reconcile grammars induced by different views of data [32]. One such view retains just the simple sentences, making it easier to recognize root words. Another splits text into many inter-punctuation fragments, helping learn word associations. The induced dependency trees can themselves also be viewed not only as directed structures but also as skeleton parses, facilitating the recovery of correct polarities for unlabeled dependency arcs.

By reusing templates, as in dynamic Bayesian network (DBN) frameworks [173, §6.2.2], it became possible to specify relatively "deep" learning architectures without sacrificing (too much) clarity or simplicity. On a still more speculative note, there are two (admittedly, tenuous) connections to human cognition. First, the benefits of not normalizing probabilities, when symmetrizing, might be related to human language processing through the

[11]The so-called Yarowsky-*cautious* modification of the original algorithm for unsupervised word-sense disambiguation.

base-rate fallacy [17, 162] and the availability heuristic [48, 326], since people are notoriously bad at probability [13, 160, 161]. And second, intermittent "unlearning" — though perhaps not of the kind that takes place inside of our transforms — is an adaptation that can be essential to cognitive development in general, as evidenced by neuronal pruning in mammals [74, 193]. "Forgetful EM" strategies that reset subsets of parameters may thus, possibly, be no less relevant to unsupervised learning than is "partial EM," which only suppresses updates, other EM variants [229], or "dropout training" [138, 332, 329], which can be important in supervised settings.

Future parsing models, in grammar induction, may benefit by modeling head-dependent relations separately from direction. As frequently employed in tasks like semantic role labeling [43] and relation extraction [315], it may be easier to first establish existence, before trying to understand its nature. Other key next steps may include exploring more intelligent ways of combining systems [317, 247] and automating the operator discovery process. Furthermore, there are reasons to be optimistic that both count transforms and model recombination could be usefully incorporated into sampling methods: although symmetrized models may have higher cross-entropies, hence prone to rejection in vanilla MCMC, they could work well as seeds in multi-chain designs; existing algorithms, such as MCMCMC [114], which switch contents of adjacent chains running at different temperatures, may also benefit from introducing the option to combine solutions, in addition to just swapping them.

Chapter 13

Conclusions

Unsupervised parsing and grammar induction are notoriously challenging problems of computational linguistics. One immediate complication that arises in solving these tasks stems from the non-convexity of typical likelihood objectives that are to be optimized. Another is poor correlation between the likelihoods attained by these unsupervised objectives and actual parsing performance. Yet a third is the high degree of disagreement between different linguistic theories and the arbitrariness of how some common syntactic constructions are analyzed, which further complicates evaluation. Because of these and many other issues, such as the fact that hierarchical syntactic structure is underdetermined by raw text, the MATCHLINGUIST task, as it had been at times (playfully?) called by Smith and Eisner [294, 291], exhibits many tell-tale signs of an ill-posed problem. Nonetheless, the work reported in this dissertation represents a number of contributions — to science, methodology and engineering of state-of-the-art systems — spanning the general fields of linguistics, non-convex optimization and machine learning, and, of course, unsupervised parsing and grammar induction specifically. Of these, contributions to improving unsupervised parsing performance are the easiest to describe, since they can be quantified, so I will start there.

This dissertation advanced the state-of-the-art in dependency grammar induction from 42.2% accuracy in 2009, measured on all sentences of a standard English news corpus [66], to 64.4% in 2013, while simultaneously eliminating previously accepted sources of supervision, such as biased initializers, manually tuned input length cutoffs, gold part-of-speech tags, and so forth. This performance jump corresponds to a 2/3 relative reduction in error towards the "skyline" supervised dependency parsing accuracy, 76.3%, that is attainable with the dependency-and-boundary models (DBMs) proposed in this thesis;[1] the phrase bracketings associated to dependency parse structures induced by DBMs happen to also be of state-of-the-art quality, by unsupervised constituent parsing metrics (unlabeled F_1), at 52.8% recall and 55.6% precision. Simultaneously, state-of-the-art macro-average of accuracies across all 19 CoNLL languages' complete held-out test sets increased from 32.6% in 2011, the first such comprehensive evaluation of grammar inducers, to 48.6% in 2013.

In addition to pushing up performance numbers, this thesis covers several methodological innovations that, I hope, will be of a more lasting nature. The first broad class of these contributions has to do with evaluation. To help guard against overfitting, I led by example, introducing into the unsupervised parsing community the standard of using *held-out test sets*, testing against *all sentence lengths*, and also evaluating across *all multilingual corpora* [42, 236], spanning many languages from disparate families. The work described in this dissertation was the earliest to employ comprehensive blind evaluation of this kind. The second broad class of contributions to methodology has to do with eliminating many formerly standard and accepted sources of supervision that have snuck into grammar induction over the years. The most prominent of these are reliance on part-of-speech tags, biased initializers, and manually tuned training subsets and termination criteria for EM. This thesis contains a collection of empirical proofs, showing that such short-cuts are, in fact, inferior to using unsupervised word clusters (Ch. 9), uninformed initializers (Chs. 2–4), nearly all available data (Ch. 11) and multiple objectives that validate proposed moves (Chs. 5, 10). For the larger field of natural language processing, it also provides: (i) an exposition of factorial experimental designs and multi-hypothesis statistical analyses of results (Chs. 5–7), which are standard throughout the natural and social sciences; (ii) a new million-plus-word English text corpus which is novel in overlaying syntactic structure and web markup (Ch. 6);[2] and (iii) a fully-unsupervised context-sensitive "part-of-speech" tagger for English (Ch. 9).[3]

Among the contributions of this dissertation to the science of linguistics are several statistical observations about the structure of natural language, which include the facts that:

[1]During the same time period, *supervised constituent* parsing scores on this evaluation set had gone up from 91.8 to 92.5 [247, 287, 182]: a 0.7 absolute and 8.5% relative reduction in *labeled bracketing* F_1 error.

[2]`http://nlp.stanford.edu/pubs/markup-data.tar.bz2`: `dp.*`

[3]`http://nlp.stanford.edu/pubs/goldtags-data.tar.bz2`: `untagger.model`

(i) syntactic dependency arc lengths tend to follow log-normal distributions, with the standard log-normal serving as a good prior for the parameters (Ch. 2); (ii) first words of sentences tend to have no dependents, whereas last words are unlikely to be leaves (Ch. 8); and (iii) web markup is correlated with syntax, often resulting from (optionally) clipping off the left- and right-most subtrees of a dependency parse, whose root is typically a noun (Ch. 6). Another key observation is that the very same constraints on syntactic structure that tend to hold for web markup are even more accurate if applied to all manner of other types of not-so-subtly bracketed text, e.g., word sequences demarcated by capitalization changes (Ch. 8) and punctuation marks (Ch. 7). Of course, the fact that, for example, punctuation correlates with syntactic structure is not new [240, 39, 157, 85]. What is new in this thesis, however, is a set of clear, simple and general rules, in the form of accurate parsing constraints, that usually govern such relations. Universal tendencies in syntactic structure are important for many aspects of computational linguistics — such as in semi-supervised parsing, where a supervised parser might be self-trained [208] on vast amounts of web data, subject to these regularities — in addition to constraining unsupervised parsing and grammar induction.

Aside from showing how to exploit markup, punctuation and capitalization as partial bracketing constraints [245] to improve dependency grammar induction (Part II), a core contribution of this dissertation to unsupervised parsing is the family of dependency-and-boundary models (Chs. 10–11). Unlike the EVG, DMV or grammatical bigrams before them, DBMs implement a truly head-outward generative process that is specifically tailored for unsupervised learning, taking advantage of more observable state, like words at fringes of phrases, as if solving a jigsaw puzzle. Moreover, DBMs are uniquely equipped to handle different classes of input, such as complete sentences and incomplete fragments. By maintaining distinct but overlapping grammars, DBM-2 and up can learn sentence lengths and root words where it is easy, from simple complete sentences, and local head-dependent relations from the unrepresentative short text fragments, enjoying the best of both worlds. It is precisely this flexibility that allows for bootstrapping of grammar inducers using nearly all available training data, by learning from inter-punctuation fragments, chopping up even the toughest sentences to extract simple, manageable pieces and expose more of the fringes.

Most of the remaining contributions in this thesis are aimed at general machine learning and non-convex optimization. They include several novel design patterns for avoiding and escaping local optima (Chs. 3, 5, 12) by exploiting multiple views of data [32]. In addition to showing how ubiquitous alternative unsupervised objectives [166] can be used to terminate or resume local search, this dissertation illustrates how intelligent non-random restarts, informed by generic transforms of previously found solutions, can be incorporated in comprehensive search networks, helping optimization algorithms like EM find good solutions despite a prevalence of local optima (Part IV). Many of these techniques generalize beyond local search with hill-climbers to global optimization via sampling methods. For example, unary transforms provide informed restarts — medium-size moves in a parameter space that are not so big as to undo all previous work, like conventional random restarts, but not so small at to be typically proposed and accepted in MCMC — which would work well for seeding multi-chain samplers; in turn, binary and higher transforms offer ways to combine systems, reconciling solutions to find improvements and potentially speeding up algorithms like MCMCMC [114], which ordinarily just swap the contents of two adjacent chains.

Turning to future directions, to anyone silly enough to commit a non-trivial fraction of their life to such a venture, my advice is to stop interpreting the grammar induction problem statement in the most strict sense of unsupervised learning, i.e., without *any* labeled examples. This approach, "unsupervised learning for the sake of unsupervised learning," makes the task needlessly difficult (though also more challenging and interesting). Fully unsupervised settings are artificial from multiple perspectives, e.g.: (i) having little to do with the science of human language acquisition, since people learn languages not in a vacuum but rather through grounded interactions with the real world; and (ii) imposing an unrealistic constraint on engineering applications of parsing, since it is not too difficult to manually annotate a handful of sentences of a desired language or genre, which might have to be done in any case for rudimentary quality control purposes. Instead, I propose a very lightly supervised approach to parsing, with minimal expenditures of annotation effort, as also suggested by Smith and Eisner [294] and recently demonstrated for part-of-speech tagging [111]. For example, starting with a single (say, median-length) annotated sentence would help with evaluation. Without a single example parse tree, grammar inducers are left having to guess many idiosyncratic, stylistic choices of a reference treebank [282, 322], such as whether modal or main verbs are heads of sentences, which not only understates their performance but also makes identifying potential algorithmic improvements harder, thereby retarding progress. By introducing a constraint, e.g., that a sample sentence be parsed correctly, it should be possible to resolve most of the major issues that complicate fully unsupervised grammar induction, due to a disconnect with actual parsing accuracies.

If we extended the task to the more general, mixed case, where both a bit of labeled and lots of unlabeled data are available, many of the problems that plague grammar induction might go away, and much of the work put forward in this dissertation could be expanded to address any remaining issues, most notably local optima. Thus, the fact that DBMs also uncover good constituent parse trees suggests yet another view that could be leveraged to make progress: alternating learning of dependency and phrase-structure representations, whenever non-convex optimization with respect to the other is stuck. Perhaps an even more interesting, third direction might include a reclustering of words into categories based on new and improved parse trees. Combined with an incremental training strategy like “baby steps,” such an approach might well learn low-level categories of individual words jointly with their relations in a higher-level hierarchy. A holistic treatment of unsupervised parsing and part-of-speech tagging seems both appropriate and long overdue, since these two intertwined challenges of syntactic structure discovery are intimately related [113], while their intrinsic objectives are not [134]. Due to the limited scope of a Ph.D. program, I did not get a chance to explore these and many other promising research avenues myself. But I believe that I have provided the field with the tools that ought to make next steps relatively painless, should a future cohort of grammar induction researchers choose to pursue them.

Bibliography

[1] Omri Abend, Roi Reichart, and Ari Rappoport. Improved unsupervised POS induction through prototype discovery. In *ACL*, 2010.

[2] Steven Abney. Parsing by chunks. *Principle-Based Parsing: Computation and Psycholinguistics*, 1991.

[3] Steven Abney. Statistical methods and linguistics. In Judith L. Klavans and Philip Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. MIT Press, 1996.

[4] Armen Allahverdyan and Aram Galstyan. Comparative analysis of Viterbi training and maximum likelihood estimation for HMMs. In *NIPS*, 2011.

[5] Eugene L. Allgower and Kurt Georg. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, 1990.

[6] Hiyan Alshawi. Head automata and bilingual tiling: Translation with minimal representations. In *ACL: Invited Talk*, 1996.

[7] Hiyan Alshawi. Head automata for speech translation. In *ICSLP*, 1996.

[8] Hiyan Alshawi. Method and apparatus for an improved language recognition system. US Patent 1999/5870706, 1996.

[9] Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26, 2000.

[10] Hiyan Alshawi, Pi-Chuan Chang, and Michael Ringgaard. Deterministic statistical mapping of sentences to underspecified semantics. In *IWCS*, 2011.

[11] Hiyan Alshawi and Shona Douglas. Learning dependency transduction models from unannotated examples. *Royal Society of London Philosophical Transactions Series A*, 358, 2000.

[12] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4, 2010.

[13] Fred Attneave. Psychological probability as a function of experienced frequency. *Experimental Psychology*, 46, 1953.

[14] James K. Baker. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65, 1979.

[15] Maria-Florina Balcan, Avrim Blum, and Ke Yang. Co-training and expansion: Towards bridging theory and practice. In *NIPS*, 2004.

[16] Michele Banko and Robert C. Moore. Part of speech tagging in context. In *COLING*, 2004.

[17] Maya Bar-Hillel. The base-rate fallacy in probability judgments. *Acta Psychologica*, 44, 1980.

[18] Cory Barr, Rosie Jones, and Moira Regelson. The linguistic structure of English web-search queries. In *EMNLP*, 2008.

[19] Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities*, 1972.

[20] Doug Beeferman, Adam Berger, and John Lafferty. CYBERPUNC: A lightweight punctuation annotation system for speech. In *ICASSP*, 1998.

[21] Anja Belz. Discovering phonotactic finite-state automata by genetic search. In *COLING-ACL*, 1998.

[22] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

[23] Jonathan Berant, Yaron Gross, Matan Mussel, Ben Sandbank, Eytan Ruppin, and Shimon Edelman. Boosting unsupervised grammar induction by splitting complex sentences on function words. In *BUCLD*, 2006.

[24] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *NAACL-HLT*, 2010.

[25] Taylor Berg-Kirkpatrick and Dan Klein. Phylogenetic grammar induction. In *ACL*, 2010.

[26] Taylor Berg-Kirkpatrick and Dan Klein. Decipherment with a million random restarts. In *EMNLP*, 2013.

[27] Aditya Bhargava and Grzegorz Kondrak. Multiple word alignment with profile hidden Markov models. In *NAACL-HLT: Student Research and Doctoral Consortium*, 2009.

[28] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *BCMS*, 35, 1943.

[29] Daniel M. Bikel. Intricacies of Collins' parsing model. *Computational Linguistics*, 30, 2004.

[30] Yonatan Bisk and Julia Hockenmaier. Simple robust grammar induction with combinatory categorial grammars. In *AAAI*, 2012.

[31] Yonatan Bisk and Julia Hockenmaier. An HDP model for inducing combinatory categorial grammars. *TACL*, 1, 2013.

[32] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[33] Phil Blunsom and Trevor Cohn. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*, 2010.

[34] Rens Bod. An all-subtrees approach to unsupervised parsing. In *COLING-ACL*, 2006.

[35] George E. P. Box and Norman R. Draper. *Empirical Model-Building and Response Surfaces*. John Wiley, 1987.

[36] Thorsten Brants. TnT — a statistical part-of-speech tagger. In *ANLP*, 2000.

[37] Michael R. Brent and Jeffrey Mark Siskind. The role of exposure to isolated words in early vocabulary development. *Cognition*, 81, 2001.

[38] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In *WWW*, 1998.

[39] Edward John Briscoe. Parsing (with) punctuation, etc. Technical report, Xerox European Research Laboratory, 1994.

[40] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19, 1993.

[41] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18, 1992.

[42] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*, 2006.

[43] Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *CoNLL*, 2005.

[44] Glenn Carroll and Eugene Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University, 1992.

[45] Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. Optimizing scoring functions and indexes for proximity search in type-annotated corpora. In *WWW*, 2006.

[46] Jason Chan, Irena Koprinska, and Josiah Poon. Co-training with a single natural feature set applied to email classification. In *WI*, 2004.

[47] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, 2007.

[48] Loren J. Chapman. Illusory correlation in observational report. *Verbal Learning and Verbal Behavior*, 6, 1967.

[49] Eugene Charniak. *Statistical Language Learning*. MIT Press, 1993.

[50] Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18, 1997.

[51] Eugene Charniak. A maximum-entropy-inspired parser. In *NAACL*, 2000.

[52] Eugene Charniak. Immediate-head parsing for language models. In *ACL*, 2001.

[53] Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*, 2005.

[54] Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, Shrivaths R. Iyengar, Jeremy Moore, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine PCFG parsing. In *HLT-NAACL*, 2006.

[55] Nick Chater, Matthew J. Crocker, and Martin J. Pickering. The rational analysis of inquiry: The case of parsing. In Mike Oaksford and Nick Chater, editors, *Rational Models of Cognition*. Oxford University Press, 1998.

[56] Conrad Chen and Hsin-Hsi Chen. Integrating punctuation rules and naïve Bayesian model for Chinese creation title recognition. In *IJCNLP*, 2005.

[57] Hsin-Hsi Chen and Yue-Shi Lee. Development of a partially bracketed corpus with part-of-speech information only. In *WVLC*, 1995.

[58] David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *COLING*, 2002.

[59] Noam Chomsky. Remarks on nominalization. In R. Jacobs and P. Rosenbaum, editors, *Readings in English Transformational Grammar*. Ginn, 1970.

[60] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. Two decades of unsupervised POS induction: How far have we come? In *EMNLP*, 2010.

[61] Alexander Clark. Inducing syntactic categories by context distribution clustering. In *CoNLL-LLL*, 2000.

[62] Alexander Clark. Combining distributional and morphological information for part of speech induction. In *EACL*, 2003.

[63] Stephen Clark, James Curran, and Miles Osborne. Bootstrapping POS-taggers using unlabelled data. In *CoNLL*, 2003.

[64] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*, 2011.

[65] Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*, 2008.

[66] Shay B. Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL-HLT*, 2009.

[67] Shay B. Cohen and Noah A. Smith. Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *ACL*, 2010.

[68] Trevor Cohn, Phil Blunsom, and Sharon Goldwater. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, 2011.

[69] Michael Collins. Three generative, lexicalised models for statistical parsing. In *ACL*, 1997.

[70] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[71] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29, 2003.

[72] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *EMNLP*, 1999.

[73] Eric Corlett and Gerald Penn. An exact A^* method for deciphering letter-substitution ciphers. In *ACL*, 2010.

[74] Fergus I. M. Craik and Ellen Bialystok. Cognition through the lifespan: mechanisms of change. *TRENDS in Cognitive Sciences*, 10, 2006.

[75] Bart Cramer. Limitations of current grammar induction algorithms. In *ACL: Student Research*, 2007.

[76] Jeremiah Crim. Co-training re-rankers for improved parser accuracy, 2002.

[77] Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL*, 2011.

[78] Sanjoy Dasgupta, Michael L. Littman, and David McAllester. PAC generalization bounds for co-training. In *NIPS*, 2001.

[79] Hal Daumé, III. Unsupervised search-based structured prediction. In *ICML*, 2009.

[80] Hal Daumé, III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75, 2009.

[81] Dmitry Davidov, Roi Reichart, and Ari Rappoport. Superior and efficient fully unsupervised pattern-based concept acquisition using an unsupervised parser. In *CoNLL*, 2009.

[82] Carl de Marcken. Lexical heads, phrase structure and the induction of grammar. In *WVLC*, 1995.

[83] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39, 1977.

[84] Inderjit S. Dhillon, Yuqiang Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *ICDM*, 2002.

[85] Christine D. Doran. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. PhD thesis, University of Pennsylvania, 1998.

[86] Marco Dorigo and Marco Colombetti. *Robot Shaping: An Experiment in Behavior Engineering*. MIT Press/Bradford Books, 1998.

[87] Gregory Druck, Gideon Mann, and Andrew McCallum. Semi-supervised learning of dependency parsers using generalized expectation criteria. In *ACL-IJCNLP*, 2009.

[88] Kevin Duh and Katrin Kirchhoff. Automatic learning of language model structure. In *COLING*, 2004.

[89] Jason Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry C. Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer Academic Publishers, 2000.

[90] Jason Eisner. Grammar induction: Beyond local search. In *ICGI*, 2012.

[91] Jason Eisner and Giorgio Satta. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *ACL*, 1999.

[92] Jason Eisner and Noah A. Smith. Parsing with soft and hard constraints on dependency length. In *IWPT*, 2005.

[93] Jason M. Eisner. An empirical comparison of probability models for dependency grammar. Technical report, IRCS, 1996.

[94] Gal Elidan, Matan Ninio, Nir Friedman, and Dale Schuurmans. Data perturbation for escaping local maxima in learning. In *AAAI*, 2002.

[95] Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48, 1993.

[96] Micha Elsner and Warren Schudy. Bounding and comparing methods for correlation clustering beyond ILP. In *NAACL-HLT: Integer Linear Programming for NLP*, 2009.

[97] David Elworthy. Does Baum-Welch re-estimation help taggers? In *ANLP*, 1994.

[98] Donald Engel, Eugene Charniak, and Mark Johnson. Parsing and disfluency placement. In *EMNLP*, 2002.

[99] Michael Yoshitaka Erlewine. The constituency of hyperlinks in a hypertext corpus. In *Linguistic Evidence*, 2012.

[100] Benoit Favre, Ralph Grishman, Dustin Hillard, Heng Ji, Dilek Hakkani-Tür, and Mari Ostendorf. Punctuating speech for information extraction. In *ICASSP*, 2008.

[101] David A. Ferrucci. IBM's Watson/DeepQA. In *ISCA*, 2011.

[102] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. The infinite tree. In *ACL*, 2007.

[103] Jenny Rose Finkel and Christopher D. Manning. Joint parsing and named entity recognition. In *NAACL-HLT*, 2009.

[104] Klaus Fischer and Vilmos Ágel. Dependency grammar and valency theory. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*. Oxford University Press, 2010.

[105] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *ICGA*, 1993.

[106] W. Nelson Francis and Henry Kučera. *Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University, 1979.

[107] Robert Frank. From regular to context-free to mildly context-sensitive tree rewriting systems: The path of child language acquisition. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. CSLI Publications, 2000.

[108] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 1997.

[109] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*, 2007.

[110] Jianfeng Gao and Mark Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *EMNLP*, 2008.

[111] Dan Garrette and Jason Baldridge. Learning a part-of-speech tagger from two hours of annotation. In *NAACL-HLT*, 2013.

[112] Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70, 1975.

[113] Douwe Gelling, Trevor Cohn, Phil Blunsom, and João Graça. The PASCAL challenge on grammar induction. In *WILS*, 2012.

[114] Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Interface Symposium*, 1991.

[115] Daniel Gildea. Corpus variation and parser performance. In *EMNLP*, 2001.

[116] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28, 2002.

[117] Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Posterior sparsity in unsupervised dependency parsing. Technical report, University of Pennsylvania, 2010.

[118] Jennifer Gillenwater, Kuzman Ganchev, João Graça, Ben Taskar, and Fernando Pereira. Sparsity in grammar induction. In *GRLL*, 2009.

[119] Kevin Gimpel and Noah A. Smith. Concavity and initialization for unsupervised dependency parsing. In *NAACL-HLT*, 2012.

[120] Fred Glover. Tabu search — Part I. *ORSA Journal on Computing*, 1, 1989.

[121] Fred Glover and Manuel Laguna. Tabu search. In Colin R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, 1993.

[122] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[123] Yoav Goldberg, Meni Adler, and Michael Elhadad. EM can find pretty good HMM POS-taggers (when given a good start). In *HLT-ACL*, 2008.

[124] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *NAACL-HLT*, 2010.

[125] Roy Goldman, Narayanan Shivakumar, Suresh Venkatasubramanian, and Hector Garcia-Molina. Proximity search in databases. In *VLDB*, 1998.

[126] Sally Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. In *ICML*, 2000.

[127] Dave Golland, John DeNero, and Jakob Uszkoreit. A feature-rich constituent context model for grammar induction. In *EMNLP-CoNLL*, 2012.

[128] Matthew R. Gormley and Jason Eisner. Nonconvex global optimization for latent-variable models. In *ACL*, 2013.

[129] Agustín Gravano, Martin Jansche, and Michiel Bacchiani. Restoring punctuation and capitalization in transcribed speech. In *ICASSP*, 2009.

[130] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24, 2009.

[131] Christian Hänig. Improvements in unsupervised co-occurrence based parsing. In *CoNLL*, 2010.

[132] Phil J. Hayes and Geroge V. Mouradian. Flexible parsing. In *ACL*, 1980.

[133] William P. Headden, III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*, 2009.

[134] William P. Headden, III, David McClosky, and Eugene Charniak. Evaluating unsupervised part-of-speech tagging for grammar induction. In *COLING*, 2008.

[135] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*, 1992.

[136] Dustin Hillard, Zhongqiang Huang, Heng Ji, Ralph Grishman, Dilek Hakkani-Tür, Mary Harper, Mari Ostendorf, and Wen Wang. Impact of automatic comma prediction on POS/name tagging of speech. In *IEEE/ACL: SLT*, 2006.

[137] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *NIPS*, 2003.

[138] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *ArXiv*, 2012.

[139] Jerry R. Hobbs. Resolving pronoun references. *Lingua*, 44, 1978.

[140] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[141] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6, 1979.

[142] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2004.

[143] Christopher R. Houck, Jeffrey A. Joines, and Michael G. Kay. Comparison of genetic algorithms, random restart, and two-opt switching for solving large location-allocation problems. *Computers & Operations Research*, 23, 1996.

[144] Xiaohua Hu, Ronald Shonkwiler, and Marcus C. Spruill. Random restarts in global optimization. Technical report, GT, 1994.

[145] Yun Huang, Min Zhang, and Chew Lim Tan. Improved constituent context model with features. In *PACLIC*, 2012.

[146] Zhi-e Huang, En-dong Xun, Gao-qi Rao, and Dong Yu. Chinese natural chunk research based on natural annotations in massive scale corpora. In *CCL/NLP-NABD*, 2013.

[147] Nobuo Inui and Yoshiyuki Kotani. Robust N-gram based syntactic analysis using segmentation words. In *PACLIC*, 2001.

[148] John P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2, 2005.

[149] Ray Jackendoff. *X-bar Syntax: A Study of Phrase Structure*. MIT Press, 1977.

[150] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*, 1980.

[151] Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. Discriminative learning with natural annotations: Word segmentation as a case study. In *ACL*, 2013.

[152] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for English. In *NODALIDA*, 2007.

[153] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37, 1988.

[154] Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24, 1998.

[155] Mark Johnson. How the statistical revolution changes (computational) linguistics. In *EACL: Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, 2009.

[156] Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *NIPS*, 2007.

[157] Bernard E. M. Jones. Exploring the role of punctuation in parsing natural text. In *COLING*, 1994.

[158] Daniel Jurafsky. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20, 1996.

[159] Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. Effective use of prosody in parsing conversational speech. In *HLT-EMNLP*, 2005.

[160] Daniel Kahneman and Amos Tversky. Subjective probability: A judgment of representativeness. *Cognitive Psychology*, 3, 1972.

[161] Daniel Kahneman and Amos Tversky. On the psychology of prediction. *Psychological Review*, 80, 1973.

[162] Daniel Kahneman and Amos Tversky. Evidential impact of base rates. In Daniel Kahneman, Paul Slovic, and Amos Tversky, editors, *Judgment under uncertainty: Heuristics and biases*. Cambridge University Press, 1982.

[163] Robert Kail. *The development of memory in children*. W. H. Freeman and Company, 2nd edition, 1984.

[164] Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a parser for machine translation reordering. In *EMNLP*, 2011.

[165] Jinyun Ke, Mieko Ogura, and William S.-Y. Wang. Optimization models of sound systems using genetic algorithms. *Computational Linguistics*, 29, 2003.

[166] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *UAI*, 1997.

[167] Ji-Hwan Kim and Philip C. Woodland. Implementation of automatic capitalisation generation systems for speech input. In *ICASSP*, 2002.

[168] Joohyun Kim and Raymond J. Mooney. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING*, 2010.

[169] Scott Kirkpatrick, Charles Daniel Gelatt, Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220, 1983.

[170] Dan Klein. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University, 2005.

[171] Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *ACL*, 2002.

[172] Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*, 2004.

[173] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[174] Terry Koo. *Advances in Discriminative Dependency Parsing*. PhD thesis, MIT, 2010.

[175] Kai A. Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110, 2009.

[176] Marco Kuhlmann and Giorgio Satta. Treebank grammar techniques for non-projective dependency parsing. In *EACL*, 2009.

[177] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6, 1992.

[178] Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4, 1990.

[179] Selmer C. Larson. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22, 1931.

[180] Danial Lashkari and Polina Golland. Convex clustering with exemplar-based models. In *NIPS*, 2008.

[181] Tuan M. V. Le, Tru H. Cao, Son M. Hoang, and Junghoo Cho. Ontology-based proximity search. In *iiWAS*, 2011.

[182] Joseph Le Roux, Antoine Rozenknop, and Jennifer Foster. Combining PCFG-LA models with dual decomposition: A case study with function labels and binarization. In *EMNLP*, 2013.

[183] Chin-Hui Lee, Chih-Heng Lin, and Biing-Hwang Juang. A study on speaker adaptation of the parameters of continuous density Hidden Markov Models. *IEEE Transactions on Signal Processing*, 39, 1991.

[184] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39, 2013.

[185] Young-Suk Lee, Salim Roukos, Yaser Al-Onaizan, and Kishore Papineni. IBM spoken language translation system. In *TC-STAR: Speech-to-Speech Translation*, 2006.

[186] Richard L. Lewis and Shravan Vasishth. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29, 2005.

[187] Xing Li, Chengqing Zong, and Rile Hu. A hierarchical parsing approach with punctuation processing for long Chinese sentences. In *IJCNLP*, 2005.

[188] Zhongguo Li and Maosong Sun. Punctuation as implicit annotations for Chinese word segmentation. *Computational Linguistics*, 35, 2009.

[189] Percy Liang and Dan Klein. Analyzing the errors of unsupervised learning. In *HLT-ACL*, 2008.

[190] Percy Liang and Dan Klein. Online EM for unsupervised models. In *NAACL-HLT*, 2009.

[191] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Evaluation of Parsing Systems*, 1998.

[192] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming. Series B*, 45, 1989.

[193] Lawrence K. Low and Hwai-Jong Cheng. Axon pruning: an essential step underlying the developmental plasticity of neuronal connections. *Royal Society of London Philosophical Transactions Series B*, 361, 2006.

[194] Bruce T. Lowerre. *The* HARPY *Speech Recognition System*. PhD thesis, CMU, 1976.

[195] Wei Lu and Hwee Tou Ng. Better punctuation prediction with dynamic conditional random fields. In *EMNLP*, 2010.

[196] Wen-Hsiang Lu, Lee-Feng Chien, and Hsi-Jian Lee. Anchor text mining for translation of Web queries: A transitive translation approach. *ACM Transactions on Information Systems*, 22, 2004.

[197] Leon B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79, 1974.

[198] David Magerman and Mitchell P. Marcus. Parsing a natural language using mutual information statistics. In *AAAI*, 1990.

[199] David M. Magerman. Statistical decision-tree models for parsing. In *ACL*, 1995.

[200] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19, 1993.

[201] David Mareček and Milan Straka. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL*, 2013.

[202] David Mareček and Zdeněk Žabokrtský. Gibbs sampling with treeness constraint in unsupervised dependency parsing. In *ROBUS*, 2011.

[203] David Mareček and Zdeněk Žabokrtský. Exploiting reducibility in unsupervised dependency parsing. In *EMNLP-CoNLL*, 2012.

[204] Sven Martin, Jörg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24, 1998.

[205] Ricardo Martin-Brualla, Enrique Alfonseca, Marius Pasca, Keith Hall, Enrique Robledo-Arnuncio, and Massimiliano Ciaramita. Instance sense induction from attribute sets. In *COLING*, 2010.

[206] Evgeny Matusov, Arne Mauser, and Hermann Ney. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *IWSLT*, 2006.

[207] David McClosky. Modeling valence effects in unsupervised grammar induction. Technical report, Brown University, 2008.

[208] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *NAACL-HLT*, 2006.

[209] David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *COLING-ACL*, 2006.

[210] David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *NAACL-HLT*, 2010.

[211] Ryan McDonald. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania, 2006.

[212] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*, 2005.

[213] Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*, 2011.

[214] Neil McIntyre and Mirella Lapata. Plot induction and evolutionary search for story generation. In *ACL*, 2010.

[215] Xiao-dan Mei, Sheng-he Sun, Jeng-shyang Pan, and Tsong-Yi Chen. Optimization of HMM by the tabu search algorithm. In *ROCLING*, 2001.

[216] I. Dan Melamed. Measuring semantic entropy. In *ACL-SIGLEX: Tagging Text with Lexical Semantics*, 1997.

[217] Chris Mellish, Alistair Knott, Jon Oberlander, and Mick O'Donnell. Experiments using stochastic search for text planning. In *INLG*, 1998.

[218] Xiao-Li Meng. EM and MCMC: Workhorses for scientific computing (thirty years of EM and much more). *Statistica Sinica*, 17, 2007.

[219] Bernard Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20, 1994.

[220] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *CIKM*, 2007.

[221] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*, 2009.

[222] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1, 2003.

[223] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 6th edition, 2005.

[224] Robert Moore, Douglas Appelt, John Dowding, J. Mark Gawron, and Douglas Moran. Combining linguistic and statistical knowledge sources in natural-language processing for ATIS. In *SLST*, 1995.

[225] Robert C. Moore and Chris Quirk. Random restarts in minimum error rate training for statistical machine translation. In *COLING*, 2008.

[226] Markos Mylonakis and Khalil Sima'an. Learning probabilistic synchronous CFGs for phrase-based translation. In *CoNLL*, 2010.

[227] Tahira Naseem and Regina Barzilay. Using semantic cues to learn syntax. In *AAAI*, 2011.

[228] Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. Using universal linguistic knowledge to guide grammar induction. In *EMNLP*, 2010.

[229] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

[230] Elissa L. Newport. Constraints on learning and their role in language acquisition: Studies of the acquisition of American Sign Language. *Language Sciences*, 10, 1988.

[231] Elissa L. Newport. Maturational constraints on language learning. *Cognitive Science*, 14, 1990.

[232] Vincent Ng and Claire Cardie. Weakly supervised natural language learning without redundant views. In *HLT-NAACL*, 2003.

[233] Jian-Yun Nie and Jiang Chen. Exploiting the Web as parallel corpora for cross-language information retrieval. *Web Intelligence*, 2002.

[234] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[235] Mikhail S. Nikulin. Hellinger distance. In Michiel Hazewinkel, editor, *Encyclopaedia of Mathematics*. Kluwer Academic Publishers, 2002.

[236] Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*, 2007.

[237] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13, 2007.

[238] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *CoNLL*, 2006.

[239] Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *HLT-ACL*, 2008.

[240] Geoffrey Nunberg. *The Linguistics of Punctuation*. CSLI Publications, 1990.

[241] Yutaka Ohyama, Toshikazu Fukushima, Tomoki Shutoh, and Masamichi Shutoh. A sentence analysis method for a Japanese book reading machine for the blind. In *ACL*, 1986.

[242] Donald Cort Olivier. *Stochastic Grammars and Language Acquisition Mechanisms*. PhD thesis, Harvard University, 1968.

[243] Mark A. Paskin. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report, UCB, 2001.

[244] Mark A. Paskin. Grammatical bigrams. In *NIPS*, 2001.

[245] Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *ACL*, 1992.

[246] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *ACL*, 1993.

[247] Slav Petrov. Products of random latent variable grammars. In *NAACL-HLT*, 2010.

[248] Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. Uptraining for accurate deterministic question parsing. In *EMNLP*, 2010.

[249] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *ArXiv*, 2011.

[250] Slav Petrov, Aria Haghighi, and Dan Klein. Coarse-to-fine syntactic machine translation using language projections. In *EMNLP*, 2008.

[251] Slav Orlinov Petrov. *Coarse-to-Fine Natural Language Processing*. PhD thesis, University of California, Berkeley, 2009.

[252] Jim Pitman and Marc Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25, 1997.

[253] Elias Ponvert, Jason Baldridge, and Katrin Erk. Simple unsupervised identification of low-level constituents. In *ICSC*, 2010.

[254] Elias Ponvert, Jason Baldridge, and Katrin Erk. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *ACL-HLT*, 2011.

[255] Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. Semantic role chunking combining complementary syntactic views. In *CoNLL*, 2005.

[256] Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[257] Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. Learning and inference over constrained output. In *IJCAI*, 2005.

[258] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2011.

[259] Mohammad Sadegh Rasooli and Heshaam Faili. Fast unsupervised dependency parsing with arc-standard transitions. In *ROBUS-UNSUP*, 2012.

[260] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, 1999.

[261] Sujith Ravi and Kevin Knight. Minimized models for unsupervised part-of-speech tagging. In *ACL-IJCNLP*, 2009.

[262] Sujith Ravi, Kevin Knight, and Radu Soricut. Automatic prediction of parser accuracy. In *EMNLP*, 2008.

[263] Roi Reichart and Ari Rappoport. Improved fully unsupervised parsing with zoomed learning. In *EMNLP*, 2010.

[264] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *ANLP*, 1997.

[265] Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *ACL*, 2002.

[266] Brian Edward Roark. *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models, and Applications*. PhD thesis, Brown University, 2001.

[267] Douglas L. T. Rohde and David C. Plaut. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72, 1999.

[268] Kenneth Rose. Deterministic annealing for clustering, compression, classification, regression and related optmization problems. *Proceedings of the IEEE*, 86, 1998.

[269] Dan Roth and Wen-tau Yih. Integer linear programming inference for conditional random fields. In *ICML*, 2005.

[270] Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. Improved parsing and POS tagging using inter-sentence consistency constraints. In *EMNLP-CoNLL*, 2012.

[271] Lisa M. Saksida, Scott M. Raymond, and David S. Touretzky. Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems*, 22, 1997.

[272] Stan Salvador and Philip Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *ICTAI*, 2004.

[273] Rajhans Samdani, Ming-Wei Chang, and Dan Roth. Unified expectation maximization. In *NAACL-HLT*, 2012.

[274] Federico Sangati and Willem Zuidema. Unsupervised methods for head assignments. In *EACL*, 2009.

[275] Terence D. Sanger. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, 10, 1994.

[276] Baskaran Sankaran, Anoop Sarkar, and Kevin Duh. Multi-metric optimization using ensemble tuning. In *NAACL-HLT*, 2013.

[277] Anoop Sarkar. Applying co-training methods to statistical parsing. In *NAACL*, 2001.

[278] Manabu Sassano. Using a partially annotated corpus to build a dependency parser for Japanese. In *IJCNLP*, 2005.

[279] Tony Savage. Shaping: The link between rats and robots. *Connection Science*, 10, 1998.

[280] Tony Savage. Shaping: A multiple contingencies analysis and its relevance to behaviour-based robotics. *Connection Science*, 13, 2001.

[281] Hinrich Schütze. Distributional part-of-speech tagging. In *EACL*, 1995.

[282] Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *ACL-HLT*, 2011.

[283] Yoav Seginer. Fast unsupervised incremental parsing. In *ACL*, 2007.

[284] Yoav Seginer. *Learning Syntactic Structure*. PhD thesis, University of Amsterdam, 2007.

[285] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *AAAI*, 1994.

[286] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *AAAI*, 1992.

[287] Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *ACL*, 2012.

[288] Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8, 1992.

[289] Burrhus Frederic Skinner. *The behavior of organisms: An experimental analysis*. Appleton-Century-Crofts, 1938.

[290] Daniel D. Sleator and Davy Temperley. Parsing English with a link grammar. In *IWPT*, 1993.

[291] Noah A. Smith. Adversarial evaluation for models of natural language. In *ArXiv*, 2012.

[292] Noah A. Smith and Jason Eisner. Annealing techniques for unsupervised statistical language learning. In *ACL*, 2004.

[293] Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *ACL*, 2005.

[294] Noah A. Smith and Jason Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *IJCAI: Grammatical Inference Applications*, 2005.

[295] Noah A. Smith and Jason Eisner. Annealing structural bias in multilingual weighted grammar induction. In *COLING-ACL*, 2006.

[296] Anders Søgaard. Data point selection for cross-language adaptation of dependency parsers. In *ACL-HLT*, 2011.

[297] Anders Søgaard. From ranked words to dependency trees: two-stage unsupervised non-projective dependency parsing. In *TextGraphs*, 2011.

[298] Anders Søgaard and Christian Rishøj. Semi-supervised dependency parsing using generalized tri-training. In *COLING*, 2010.

[299] Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *PNAS*, 102, 2005.

[300] Francisco J. Solis and Roger J-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6, 1981.

[301] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *GRLL*, 2009.

[302] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. From Baby Steps to Leapfrog: How "Less is More" in unsupervised dependency parsing. In *NAACL-HLT*, 2010.

[303] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*, 2011.

[304] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*, 2011.

[305] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Bootstrapping dependency grammar inducers from incomplete sentence fragments via austere models. In *ICGI*, 2012.

[306] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Capitalization cues improve dependency grammar induction. In *WILS*, 2012.

[307] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Three dependency-and-boundary models for grammar induction. In *EMNLP-CoNLL*, 2012.

[308] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *EMNLP*, 2013.

[309] Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. Viterbi training improves unsupervised dependency parsing. In *CoNLL*, 2010.

[310] Valentin I. Spitkovsky, Angel X. Chang, Hiyan Alshawi, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *EMNLP*, 2011.

[311] Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. Profiting from mark-up: Hyper-text annotations for guided parsing. In *ACL*, 2010.

[312] Wolfram Stadler, editor. *Multicriteria Optimization in Engineering and in the Sciences*. Plenum Press, 1988.

[313] Efstathios Stamatatos, Nikos Fakotakis, and Georgios Kokkinakis. Text genre detection using common word frequencies. In *COLING*, 2000.

[314] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B*, 36, 1974.

[315] Ang Sun, Ralph Grishman, and Satoshi Sekine. Semi-supervised relation extraction with large-scale word clustering. In *ACL*, 2011.

[316] Weiwei Sun and Jia Xu. Enhancing Chinese word segmentation using unlabeled data. In *EMNLP*, 2011.

[317] Mihai Surdeanu and Christopher D. Manning. Ensemble models for dependency parsing: Cheap and good? In *NAACL-HLT*, 2010.

[318] Bin Tan and Fuchun Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW*, 2008.

[319] Lucien Tesnière. *Éléments de syntaxe structurale*. Éditions Klincksieck, 1959.

[320] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*, 2003.

[321] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP-VLC*, 2000.

[322] Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. Evaluating dependency parsing: Robust and heuristics-free cross-annotation evaluation. In *EMNLP*, 2011.

[323] Kewei Tu and Vasant Honavar. On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI*, 2011.

[324] Kewei Tu and Vasant Honavar. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *EMNLP-CoNLL*, 2012.

[325] Kewei Tu, Maria Pavlovskaia, and Song-Chun Zhu. Unsupervised structure learning of stochastic and-or grammars. In *NIPS*, 2014.

[326] Amos Tversky and Daniel Kahneman. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5, 1973.

[327] David Vadas and James R. Curran. Adding noun phrase structure to the Penn Treebank. In *ACL*, 2007.

[328] David Vickrey and Daphne Koller. Sentence simplification for semantic role labeling. In *HLT-ACL*, 2008.

[329] Stefan Wager, Sida Wang, and Percy Liang. Dropout training as adaptive regularization. In *NIPS*, 2014.

[330] Martin J. Wainwright. Estimating the “wrong” graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, 7, 2006.

[331] Qin Iris Wang, Dale Schuurmans, and Dekang Lin. Semi-supervised convex training for dependency parsing. In *HLT-ACL*, 2008.

[332] Sida I. Wang and Christopher D. Manning. Fast dropout training. In *ICML*, 2013.

[333] William Yang Wang and Kathleen R. McKeown. “Got you!”: Automatic vandalism detection in Wikipedia with web-based shallow syntactic-semantic modeling. In *COLING*, 2010.

[334] Gregory Ward and Betty J. Birner. Discourse and information structure. In Deborah Schiffrin, Deborah Tannen, and Heidi Hamilton, editors, *Handbook of Discourse Analysis*. Oxford: Basil Blackwell, 2001.

[335] Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. A graph-based approach to named entity categorization in Wikipedia using conditional random fields. In *EMNLP-CoNLL*, 2007.

[336] Michael White and Rajakrishnan Rajkumar. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *GEAF*, 2008.

[337] Fei Xia and Martha Palmer. Converting dependency structures to phrase structures. In *HLT*, 2001.

[338] Tong Xiao, Jingbo Zhu, Muhua Zhu, and Huizhen Wang. Boosting-based system combination for machine translation. In *ACL*, 2010.

[339] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *IWPT*, 2003.

[340] Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL*, 2001.

[341] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.

[342] Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, and Aitor Soroa. WikiWalk: Random walks on Wikipedia for semantic relatedness. In *TextGraphs*, 2009.

[343] Kevin C. Yeh. Bilingual sentence alignment based on punctuation marks. In *ROCLING: Student*, 2003.

[344] Ainur Yessenalina, Yisong Yue, and Claire Cardie. Multi-level structured models for document-level sentiment classification. In *EMNLP*, 2010.

[345] Deniz Yuret. *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, MIT, 1998.

[346] Kaixu Zhang, Jinsong Su, and Changle Zhou. Improving Chinese word segmentation using partially annotated sentences. In *CCL/NLP-NABD*, 2013.

[347] Yan Zhou and Sally Goldman. Democratic co-learning. In *ICTAI*, 2004.