Breaking Out of Local Optima with Count Transforms and Model Recombination: A Study in Grammar Induction

Valentin I. Spitkovsky
valentin@cs.stanford.edu

Hiyan Alshawi
hiyan@google.com

Daniel Jurafsky
jurafsky@stanford.edu

Abstract

Many statistical learning problems in NLP call for local model search methods. But accuracy tends to suffer with current techniques, which often explore either too narrowly or too broadly: hill-climbers can get stuck in local optima, whereas samplers may be inefficient. We propose to arrange individual local optimizers into organized networks. Our building blocks are operators of two types: (i) *transform*, which suggests new places to search, via non-random restarts from already-found local optima; and (ii) *join*, which merges candidate solutions to find better optima. Experiments on grammar induction show that pursuing different transforms (e.g., discarding parts of a learned model or ignoring portions of training data) results in improvements. Groups of locally-optimal solutions can be further perturbed jointly, by constructing mixtures. Using these tools, we designed several modular dependency grammar induction networks of increasing complexity. Our complete system achieves 48.6% accuracy (directed dependency macro-average over all 19 languages in the 2006/7 CoNLL data) — more than 5% higher than the previous state-of-the-art.

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 (Paskin 2001a; 2001b). That parsing model has since been extended to make unsupervised learning more feasible (Klein and Manning, 2004; Headden et al., 2009; Spitkovsky et al., 2012b). But even the latest techniques 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 (Hu et al., 1994), 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 (Cohn et al., 2011; Mareček and Žabokrtský, 2011; Naseem and Barzilay, 2011).

We propose 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. We show that 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 (Jelinek and Mercer, 1980) 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.

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:

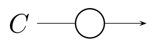
$$\begin{tikzcd} C \arrow[r] & \boxed{L_{\mathcal{D}}} \arrow[r] & C^* \end{tikzcd} \tag{1}$$

Unless C^* is a global optimum, we should be able 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 we had another optimizer L' — or a fresh starting point C' — then more complicated networks could become useful.

2.1 Transforms (Unary)

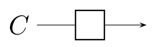
New starts could be chosen by perturbing an existing solution, as in MCMC, or independently of previous results, as in random restarts. We focus on intermediate changes to C, without injecting randomness.

All of our 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. We use circular shapes to represent such model ablation operators:



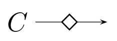
(2)

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. We will use squares to represent data-set filtering:



(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:

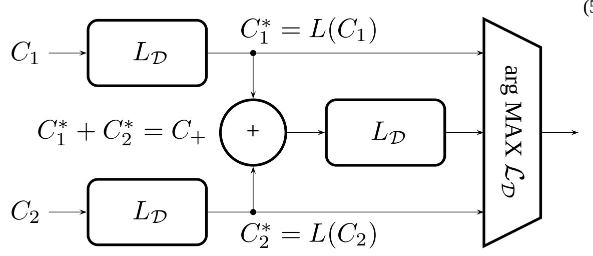


(4)

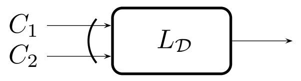
2.2 Joins (Binary)

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.

Our joining technique could do better than either C_1^* or C_2^*, by entertaining also a third possibility, which combines the two candidates. We construct a mixture model 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, $\mathcal{D}$). We return the best of C_1^*, C_2^* and $C_+^* = L(C_+)$. This approach may uncover more (and never returns less) likely solutions than choosing among C_1^*, C_2^* alone:

(5)


We will use a short-hand notation to represent the combiner network diagrammed above, less clutter:

(6)


3 The Task and Methodology

We apply transform and join paradigms to grammar induction, an important problem of computational linguistics that involves notoriously difficult objectives (Pereira and Schabes, 1992; de Marcken, 1995; Gimpel and Smith, 2012, *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.

3.1 Models and Data

We constrain all parse structures to be projective, via dependency-and-boundary grammars (Spitkovsky et al., 2012a; 2012b): DBMs 0–3 are head-outward generative parsing models (Alshawi, 1996) that distinguish complete sentences from incomplete fragments in a corpus $\mathcal{D}$: $\mathcal{D}_{\text{comp}}$ comprises inputs ending with punctuation; $\mathcal{D}_{\text{frag}} = \mathcal{D} - \mathcal{D}_{\text{comp}}$ is everything

[1]If desired, a scaling factor could be used to bias C_+ towards either C_1^* or C_2^*, for example based on their likelihood ratio.

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. Spitkovsky et al. 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 (2012b); 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 (2012a). In our 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 sparse 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. We will exploit both DBM-0 and the full DBM,[2] drawing also on split, simple and raw views of input text.

All experiments prior to final multi-lingual evaluation will use the Penn English Treebank's Wall Street Journal (WSJ) portion (Marcus et al., 1993) as the underlying tokenized and sentence-broken corpus $\mathcal{D}$. Instead of gold parts-of-speech, we plugged in 200 context-sensitive unsupervised tags, from Spitkovsky et al. (2011c),[3] for the word categories.

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, we will always obtain a delexicalized-and-smoothed instance first.

3.3 Optimization and Viterbi Decoding

We use "early-switching lateen" EM (Spitkovsky et al., 2011a, §2.4) 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 handles only our shorter inputs ($l \leq 15$), starting with soft EM ($L = \text{SL}$, for "soft lateen"). Lexicalized models will cover full data ($l \leq 45$) and employ "early-stopping lateen" EM (2011a, §2.3), 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 ($L = \text{H}$) is the one better suited to long inputs (Spitkovsky et al., 2010).

Our decoders always force an inter-punctuation fragment to derive itself (Spitkovsky et al., 2011b, §2.2).[4] 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).

3.4 Final Evaluation and Metrics

Evaluation is against held-out CoNLL shared task data (Buchholz and Marsi, 2006; Nivre et al., 2007), spanning 19 languages. We compute performance as directed dependency accuracies (DDA), fractions of correct unlabeled arcs in parsed output (an extrinsic metric).[5] For most WSJ experiments we include also sentence and parse tree cross-entropies (soft and hard EMs' intrinsic metrics), in bits per token (bpt).

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

[3]`http://nlp.stanford.edu/pubs/goldtags-data.tar.bz2`

[4]But these constraints do not impact training with shorter inputs, since there is no internal punctuation in $\mathcal{D}_{\text{split}}$ or $\mathcal{D}_{\text{simp}}$.

[5]We converted gold labeled constituents in WSJ to unlabeled reference dependencies using deterministic "head-percolation" rules (Collins, 1999); sentence root symbols, though not punctuation arcs, contribute to scores, as is standard (Paskin, 2001b).

4 Concrete Operators

We will now instantiate the operators sketched out in §2 specifically for the grammar induction task.

Throughout, we repeatedly employ single steps of Viterbi training 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.[6]

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:

$$C \;—\boxed{F}\diamond\!\!\longrightarrow \tag{7}$$

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, we showed, aids grammar induction (Spitkovsky et al., 2012b, §7.1).

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. We implemented symmetrization also 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:

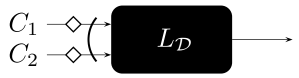
$$C \;—\!\!\textcircled{S}\diamond\!\!\longrightarrow \tag{8}$$

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 (Spitkovsky et al., 2009, Figure 3: 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, our implementation attributes half its weight to the intended construction, ⓐ↷ⓩ, reserving the other half for the symmetric structure, ⓩ attaching ⓐ to its left: ⓐ↶ⓩ. 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 we don't turn word attachment scores into probabilities, consider sentences ⓐ ⓩ and ⓒ ⓩ. The fact that ⓩ co-occurs with ⓐ introduces an asymmetry into ⓩ's relation with ⓒ: $\mathbb{P}$(ⓩ | ⓒ) $= 1$ differs from $\mathbb{P}$(ⓒ | ⓩ) $= 1/2$. Normalizing might force the interpretation ⓒ↷ⓩ (and also ⓐ↷ⓩ), 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.

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, we convert each input C_i 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:[7]



(9)

5 Basic Networks

We are ready to propose a non-trivial subnetwork for grammar induction, based on the transform and join operators, which we will reuse in larger networks.

5.1 Fork/Join (FJ)

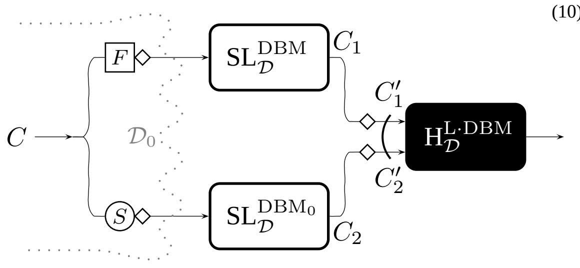
Given a model that parses a base data set $\mathcal{D}_0$, the fork/join subnetwork will output an adaptation of that model for $\mathcal{D}$. It could facilitate a grammar induction process, e.g., by advancing it from smaller

[6]A related approach — initializing EM training with an M-step — was advocated by Klein and Manning (2004, §3).

[7]In our diagrams, lexicalized modules are shaded black.

to larger — or possibly more complex — data sets.

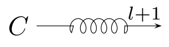
We first fork off two variations of the incoming model based on $\mathcal{D}_0$: (i) a filtered view, which focuses on cleaner, simpler data (transform #1); and (ii) a symmetrized view that backs off to word associations (transform #2). Next is grammar induction over $\mathcal{D}$. We optimize a full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


C
F
S
$\mathcal{D}_0$
$\mathrm{SL}_{\mathcal{D}}^{\mathrm{DBM}}$
$\mathrm{SL}_{\mathcal{D}}^{\mathrm{DBM}_0}$
C_1
C_2
C'_1
C'_2
$\mathrm{H}_{\mathcal{D}}^{\mathrm{L\cdot DBM}}$
(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 we can't know ahead of time which is the true case, we pursue both optimization paths simultaneously and let a combiner later decide for us.

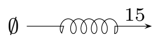
Note that the forks start (and end) optimizing with soft EM. This is because soft EM integrates previously unseen tokens into new grammars better than hard EM, as evidenced by our failed attempt to reproduce the "baby steps" strategy with Viterbi training (Spitkovsky et al., 2010, Figure 4). A combiner 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.

We will use a "grammar inductor" to represent subnetworks that transition from $\mathcal{D}_{\text{split}}^{l}$ to $\mathcal{D}_{\text{split}}^{l+1}$, 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 $l+1$:



(11)

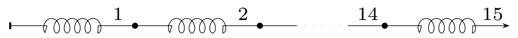
The FJ network instantiates a grammar inductor with $l = 14$, thus training on inter-punctuation fragments up to length 15, as in previous work, starting from an empty set of counts, $C = \emptyset$. Smoothing causes initial parse trees to be chosen uniformly at random, as suggested by Cohen and Smith (2010):



(12)

5.2 Iterated Fork/Join (IFJ)

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



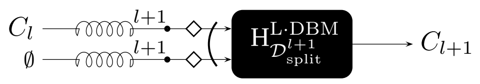
(13)

We diagrammed this system 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 (Rose, 1998), and is patterned after "baby steps" (Spitkovsky et al., 2009, §4.2).

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 our generative models factor (see §4.2) or how ties are broken in optimization (Spitkovsky et al., 2012a, Appendix B). We 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 such phase transitions correctly (or gracefully).

5.3 Grounded Iterated Fork/Join (GIFJ)

So far, our 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. Our 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):


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

The full GIFJ network can then be obtained by unrolling the above template from $l = 14$ back to one.

		$\mathrm{WSJ}^{15}_{\mathrm{split}}$			$\mathrm{WSJ}^{15}_{\mathrm{simp}}$				
Instance Label	*Model*	$\hbar_{sents}$	$\hbar_{trees}$	*DDA*	$\hbar_{sents}$	$\hbar_{trees}$	*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 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).

6 Performance of Basic Networks

We compared our three networks' performance on their final training sets, $\mathrm{WSJ}^{15}_{\mathrm{split}}$ (see Table 1, which also tabulates results for a cleaner subset, $\mathrm{WSJ}^{15}_{\mathrm{simp}}$). The first network starts from $C = \emptyset$, helping us 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.

6.1 Fork/Join (FJ)

FJ's symmetrizer yields random parses of $\mathrm{WSJ}^{14}_{\mathrm{split}}$, 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 $\mathrm{WSJ}^{14}_{\mathrm{simp}}$ 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).

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, we experimented with ablated systems that don't fork or join, i.e., our classic "baby steps" schema (chaining together 15 optimizers), using both DBM and DBM_0, with and without a transform in-between. However, all such "linear" networks scored well below 50%. We conclude from these results 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.

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.

7 Enhanced Subnetworks

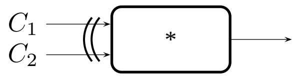
Modularity and abstraction allow for compact representations of complex systems. Another key benefit is that individual components can be understood and improved in isolation, as we will demonstrate next.

7.1 An Iterative Combiner (IC)

Our 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, we 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 this approach is still a bit timid.

A more greedy way is to proceed so long as C^*_+ is not worse than *both* predecessors. We shall now state our most general iterative combiner (IC) algorithm: Start with a solution pool $p = \{C^*_i\}^n_{i=1}$. Next, construct p' by adding $C^*_+ = L(\sum^n_{i=1} C^*_i)$ 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 1), lowering cross-entropy (down from 6.96 to 6.93bpt). We propose a distinguished notation for the ICs:

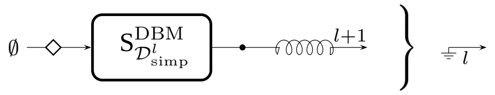


(15)

7.2 A Grammar Transformer (GT)

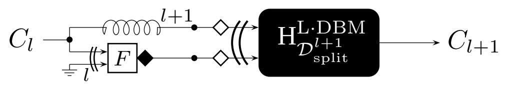
The levels of our systems' performance at grammar induction thus far suggest that the space of possible networks (say, with up to k components) may itself be worth exploring more thoroughly. We leave this exercise to future work, ending with two relatively straight-forward extensions for grounded systems.

Our static bootstrapping mechanism ("ground" of GIFJ) can be improved by pretraining with simple sentences first — as in the curriculum for learning DBM-1 (Spitkovsky et al., 2012b, §7.1), but now with a variable length cut-off l (much lower than the original 45) — instead of starting from $\emptyset$ directly:



(16)

The output of this subnetwork can then be refined, by reconciling it with a previous dynamic solution. We perform a mini-join of a new ground's counts with C_l, using the filter transform (single steps of *lexicalized* Viterbi training on clean, simple data), ahead of the main join (over more training data):



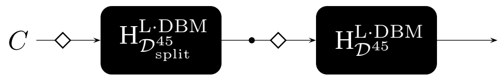
C_l
$l{+}1$
l
F
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{l+1}_{\mathrm{split}}}$
C_{l+1}

(17)

This template can be unrolled, as before, to obtain our last network (GT), which achieves 72.9% accuracy and 6.83bpt cross-entropy (slightly less accurate with basic combiners, at 72.3% — see Table 1).

8 Full Training and System Combination

All systems that we described so far stop training at $\mathcal{D}^{15}_{\mathrm{split}}$. We will use a two-stage adaptor network to transition their grammars to a full data set, $\mathcal{D}^{45}$:

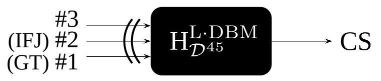


C
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{45}_{\mathrm{split}}}$
$\mathrm{H}^{\mathrm{L\cdot DBM}}_{\mathcal{D}^{45}}$

(18)

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 $l = 45$).[8]

After full training, our 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 2). To test the generalized IC algorithm, we merged our implementations of these three strong grammar induction pipelines into a combined system (CS). It scored highest: 64.4%.



(19)

The quality of bracketings corresponding to (non-trivial) spans derived by heads of our dependency structures 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

[8]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	DDA	(@10)
		(Gimpel and Smith, 2012)	53.1	(64.3)
		(Gillenwater et al., 2010)	53.3	(64.3)
		(Bisk and Hockenmaier, 2012)	53.3	(71.5)
		(Blunsom and Cohn, 2010)	55.7	(67.7)
		(Tu and Honavar, 2012)	57.0	(71.4)
		(Spitkovsky et al., 2011b)	58.4	(71.4)
		(Spitkovsky et al., 2011c)	59.1	(71.4)
#3		(Spitkovsky et al., 2012a)	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 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.

System		F_1	P	R
Binary-Branching	Upper Bound	85.7		
Left-Branching	Baseline	12.0		
CCM (Klein and Manning, 2002)		33.7		
Right-Branching	Baseline	40.7		
F-CCM	(Huang et al., 2012)	45.1		
HMM	(Ponvert et al., 2011)	46.3		
LLCCM	(Golland et al., 2012)	47.6		
CCL	(Seginer, 2007)	52.8	54.6	51.1
PRLG	(Ponvert et al., 2011)	**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 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 our combined system (CS), recent state-of-the-art and the baselines.

PRLG (Ponvert et al., 2011), which is the strongest system of which we are aware (see Table 3).[9]

9 Multi-Lingual Evaluation

Last, we checked how our algorithms generalize outside English WSJ, by testing in 23 more set-ups: all 2006/7 CoNLL test sets (Buchholz and Marsi, 2006; Nivre et al., 2007), spanning 19 languages. Most recent work evaluates against this multi-lingual data, with the unrealistic assumption of part-of-speech tags. But since inducing high quality word clusters for many languages would be beyond the scope of our paper, here we too plugged in gold tags for word categories (instead of unsupervised tags, as in §3–8).

We compared to the two strongest systems we knew:[10] MZ (Mareček and Žabokrtský, 2012) and SAJ (Spitkovsky et al., 2012b), which report average accuracies of 40.0 and 42.9% for CoNLL data (see Table 4). Our fully-trained IFJ and GT systems score 40.0 and 47.6%. As before, combining these networks with our own implementation of the best previous state-of-the-art system (SAJ) yields a further improvement, increasing final accuracy to 48.6%.

[9]These numbers differ from Ponvert et al.'s (2011, 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 (2012, Figure 1) for CCM and LLCCM; Huang et al.'s (2012, Table 2) for the rest.

[10]During review, another strong system (Mareček and Straka, 2013, scoring 48.7%) of possible interest to the reader came out, exploiting prior knowledge of stopping probabilities (estimated from large POS-tagged corpora, via reducibility principles).

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 our (and previous) results (see Table 4).

Nevertheless, if we look at the more stable average accuracies, we see a positive trend as we move 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.

11 Related Work

The surest way to avoid local optima is to craft an objective that doesn't have them. For example, Wang et al. (2008) demonstrated a convex training method for semi-supervised dependency parsing; Lashkari and Golland (2008) introduced a convex reformulation of likelihood functions for clustering tasks; and Corlett and Penn (2010) designed

		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 4: Blind evaluation on 2006/7 CoNLL test sets (all sentences) for our full networks (IFJ and GT), previous state-of-the-art systems of Spitkovsky et al. (2012b) and Mareček and Žabokrtský (2012), and three-way combination with SAJ (CS, including results up to length ten).

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 (Eisner, 2012; Gormley and Eisner, 2013) may be a good fit for the smaller and simpler, earlier stages of our iterative networks.

Multi-start methods (Solis and Wets, 1981) 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 (Hu et al., 1994). This approach is rarely emphasized in NLP literature. For instance, Moore and Quirk (2008) demonstrated consistent, substantial gains from random restarts in statistical machine translation (but also suggested better and faster replacements — see below); Ravi and Knight (2009, §5, Figure 8) found random restarts for EM to be crucial in parts-of-speech disambiguation. However, other reviews are few and generally negative (Kim and Mooney, 2010; Martin-Brualla et al., 2010).

Iterated local search methods (Hoos and Stützle, 2004; Johnson et al., 1988, *inter alia*) escape local basins of attraction by perturbing candidate solutions, without undoing all previous work. "Large-step" moves can come from jittering (Hinton and Roweis, 2003), dithering (Price et al., 2005, Ch. 2) or smoothing (Bhargava and Kondrak, 2009). Non-improving "sideways" moves offer substantial help with hard satisfiability problems (Selman et al., 1992); and injecting non-random noise (Selman et al., 1994), by introducing "uphill" moves via mixtures of random walks and greedy search strategies, does better than random noise alone or simulated annealing (Kirkpatrick et al., 1983). In NLP, Moore and Quirk's (2008) random walks from previous local optima were faster than uniform sampling and also increased BLEU scores; Elsner and Schudy (2009) showed that local search can outperform greedy solutions for document clustering and chat disentanglement tasks; and Mei et al. (2001) incorporated tabu search (Glover, 1989; Glover and Laguna, 1993, Ch. 3) into HMM training for ASR.

Genetic algorithms are a fusion of what's best in local search and multi-start methods (Houck et al., 1996), exploiting a problem's structure to combine valid parts of any partial solutions (Holland, 1975; Goldberg, 1989). Evolutionary heuristics proved useful in the induction of phonotactics (Belz, 1998), text planning (Mellish et al., 1998), factored modeling of morphologically-rich languages (Duh and Kirchhoff, 2004) and plot induction for story generation (McIntyre and Lapata, 2010). Multi-objective genetic algorithms (Fonseca and Fleming, 1993) can handle problems with equally important but conflicting criteria (Stadler, 1988), 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 (Ke et al., 2003). Our 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.

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 (Liang and Klein, 2009). Elidan et al. (2002) 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 (Elman, 1993), beginning from simpler or more certain cases. This family of techniques has met with success in semi-supervised named entity classification (Collins and Singer, 1999; Yarowsky, 1995),[11] parts-of-speech induction (Clark, 2000; 2003), and language modeling (Krueger and Dayan, 2009; Bengio et al., 2009), in addition to unsupervised parsing (Spitkovsky et al., 2009; Tu and Honavar, 2011; Cohn et al., 2011).

12 Conclusion

We 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 (Xiao et al., 2010), ours do not require incoming models to be of similar quality to make improvements. We exploited these properties of the combiners to reconcile grammars induced by different views of data (Blum and Mitchell, 1998). 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 (Koller and Friedman, 2009, §6.2.2), we managed to specify relatively "deep" learning architectures without sacrificing (too much) clarity or simplicity. On a still more speculative note, we see 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 base-rate fallacy (Bar-Hillel, 1980; Kahneman and Tversky, 1982) and the availability heuristic (Chapman, 1967; Tversky and Kahneman, 1973), since people are notoriously bad at probability (Attneave, 1953; Kahneman and Tversky, 1972; Kahneman and Tversky, 1973). 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 (Craik and Bialystok, 2006; Low and Cheng, 2006). "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 (Neal and Hinton, 1999), or "dropout training" (Hinton et al., 2012; Wang and Manning, 2013), which is 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 (Carreras and Màrquez, 2005) and relation extraction (Sun et al., 2011), 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 (Surdeanu and Manning, 2010; Petrov, 2010) and automating the operator discovery process. Furthermore, we are 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 (Geyer, 1991), 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.

[11]The so-called Yarowsky-*cautious* modification of the original algorithm for unsupervised word-sense disambiguation.

Acknowledgments

We thank Yun-Hsuan Sung, for early-stage discussions on ways of extending "baby steps," Elias Ponvert, for sharing all of the relevant experimental results and evaluation scripts from his work with Jason Baldridge and Katrin Erk, and the anonymous reviewers, for their helpful comments on the draft version of this paper. Funded, in part, by Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program, under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Once again, the first author thanks Moofus.

References

H. Alshawi. 1996. Head automata for speech translation. In *ICSLP*.

F. Attneave. 1953. Psychological probability as a function of experienced frequency. *Experimental Psychology*, 46.

M. Bar-Hillel. 1980. The base-rate fallacy in probability judgments. *Acta Psychologica*, 44.

A. Belz. 1998. Discovering phonotactic finite-state automata by genetic search. In *COLING-ACL*.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *ICML*.

A. Bhargava and G. Kondrak. 2009. Multiple word alignment with profile hidden Markov models. In *NAACL-HLT: Student Research and Doctoral Consortium*.

Y. Bisk and J. Hockenmaier. 2012. Simple robust grammar induction with combinatory categorial grammars. In *AAAI*.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

P. Blunsom and T. Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *CoNLL*.

L. J. Chapman. 1967. Illusory correlation in observational report. *Verbal Learning and Verbal Behavior*, 6.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL-LLL*.

A. Clark. 2003. Combining distributional and morphological information for part of speech induction. In *EACL*.

S. B. Cohen and N. A. Smith. 2010. Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *ACL*.

T. Cohn, P. Blunsom, and S. Goldwater. 2011. Inducing tree-substitution grammars. *JMLR*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

E. Corlett and G. Penn. 2010. An exact A* method for deciphering letter-substitution ciphers. In *ACL*.

F. I. M. Craik and E. Bialystok. 2006. Cognition through the lifespan: mechanisms of change. *TRENDS in Cognitive Sciences*, 10.

C. de Marcken. 1995. Lexical heads, phrase structure and the induction of grammar. In *WVLC*.

K. Duh and K. Kirchhoff. 2004. Automatic learning of language model structure. In *COLING*.

J. Eisner. 2012. Grammar induction: Beyond local search. In *ICGI*.

G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. 2002. Data perturbation for escaping local maxima in learning. In *AAAI*.

J. L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48.

M. Elsner and W. Schudy. 2009. Bounding and comparing methods for correlation clustering beyond ILP. In *NAACL-HLT: Integer Linear Programming for NLP*.

C. M. Fonseca and P. J. Fleming. 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *ICGA*.

C. J. Geyer. 1991. Markov chain Monte Carlo maximum likelihood. In *Interface Symposium*.

J. Gillenwater, K. Ganchev, J. Graça, F. Pereira, and B. Taskar. 2010. Posterior sparsity in unsupervised dependency parsing. Technical report, University of Pennsylvania.

K. Gimpel and N. A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *NAACL-HLT*.

F. Glover and M. Laguna. 1993. Tabu search. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications.

F. Glover. 1989. Tabu search — Part I. *ORSA Journal on Computing*, 1.

D. E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.

D. Golland, J. DeNero, and J. Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *EMNLP-CoNLL*.

M. R. Gormley and J. Eisner. 2013. Nonconvex global optimization for latent-variable models. In *ACL*.

W. P. Headden, III, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

G. Hinton and S. Roweis. 2003. Stochastic neighbor embedding. In *NIPS*.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. In *ArXiv*.

J. H. Holland. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press.

H. H. Hoos and T. Stützle. 2004. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

C. R. Houck, J. A. Joines, and M. G. Kay. 1996. Comparison of genetic algorithms, random restart, and two-opt switching for solving large location-allocation problems. *Computers & Operations Research*, 23.

X. Hu, R. Shonkwiler, and M. C. Spruill. 1994. Random restarts in global optimization. Technical report, GT.

Y. Huang, M. Zhang, and C. L. Tan. 2012. Improved constituent context model with features. In *PACLIC*.

F. Jelinek and R. L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*.

D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. 1988. How easy is local search? *Journal of Computer and System Sciences*, 37.

D. Kahneman and A. Tversky. 1972. Subjective probability: A judgment of representativeness. *Cognitive Psychology*, 3.

D. Kahneman and A. Tversky. 1973. On the psychology of prediction. *Psychological Review*, 80.

D. Kahneman and A. Tversky. 1982. Evidential impact of base rates. In D. Kahneman, P. Slovic, and A. Tversky, editors, *Judgment under uncertainty: Heuristics and biases*. Cambridge University Press.

J. Ke, M. Ogura, and W. S.-Y. Wang. 2003. Optimization models of sound systems using genetic algorithms. *Computational Linguistics*, 29.

J. Kim and R. J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *COLING*.

S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL*.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

K. A. Krueger and P. Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110.

D. Lashkari and P. Golland. 2008. Convex clustering with exemplar-based models. In *NIPS*.

P. Liang and D. Klein. 2009. Online EM for unsupervised models. In *NAACL-HLT*.

L. K. Low and H.-J. Cheng. 2006. Axon pruning: an essential step underlying the developmental plasticity of neuronal connections. *Royal Society of London Philosophical Transactions Series B*, 361.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

D. Mareček and M. Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL*.

D. Mareček and Z. Žabokrtský. 2011. Gibbs sampling with treeness constraint in unsupervised dependency parsing. In *ROBUS*.

D. Mareček and Z. Žabokrtský. 2012. Exploiting reducibility in unsupervised dependency parsing. In *EMNLP-CoNLL*.

R. Martin-Brualla, E. Alfonseca, M. Pasca, K. Hall, E. Robledo-Arnuncio, and M. Ciaramita. 2010. Instance sense induction from attribute sets. In *COLING*.

N. McIntyre and M. Lapata. 2010. Plot induction and evolutionary search for story generation. In *ACL*.

X.-d. Mei, S.-h. Sun, J.-s. Pan, and T.-Y. Chen. 2001. Optimization of HMM by the tabu search algorithm. In *ROCLING*.

C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. 1998. Experiments using stochastic search for text planning. In *INLG*.

R. C. Moore and C. Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *COLING*.

T. Naseem and R. Barzilay. 2011. Using semantic cues to learn syntax. In *AAAI*.

R. M. Neal and G. E. Hinton. 1999. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

M. A. Paskin. 2001a. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report, UCB.

M. A. Paskin. 2001b. Grammatical bigrams. In *NIPS*.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL*.

S. Petrov. 2010. Products of random latent variable grammars. In *NAACL-HLT*.

E. Ponvert, J. Baldridge, and K. Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *ACL-HLT*.

K. V. Price, R. M. Storn, and J. A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Springer.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *ACL-IJCNLP*.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression and related optmization problems. *Proceedings of the IEEE*, 86.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *ACL*.

B. Selman, H. Levesque, and D. Mitchell. 1992. A new method for solving hard satisfiability problems. In *AAAI*.

B. Selman, H. A. Kautz, and B. Cohen. 1994. Noise strategies for improving local search. In *AAAI*.

F. J. Solis and R. J.-B. Wets. 1981. Minimization by random search techniques. *Mathematics of Operations Research*, 6.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2009. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *GRLL*.

V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *CoNLL*.

V. I. Spitkovsky, A. X. Chang, H. Alshawi, and D. Jurafsky. 2011c. Unsupervised dependency parsing without gold part-of-speech tags. In *EMNLP*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2012a. Bootstrapping dependency grammar inducers from incomplete sentence fragments via austere models. In *ICGI*.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. 2012b. Three dependency-and-boundary models for grammar induction. In *EMNLP-CoNLL*.

W. Stadler, editor. 1988. *Multicriteria Optimization in Engineering and in the Sciences*. Plenum Press.

A. Sun, R. Grishman, and S. Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *ACL*.

M. Surdeanu and C. D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *NAACL-HLT*.

K. Tu and V. Honavar. 2011. On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI*.

K. Tu and V. Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *EMNLP-CoNLL*.

A. Tversky and D. Kahneman. 1973. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5.

S. I. Wang and C. D. Manning. 2013. Fast dropout training. In *ICML*.

Q. I. Wang, D. Schuurmans, and D. Lin. 2008. Semi-supervised convex training for dependency parsing. In *HLT-ACL*.

T. Xiao, J. Zhu, M. Zhu, and H. Wang. 2010. Boosting-based system combination for machine translation. In *ACL*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*.