Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency

Dan Klein
Computer Science Department
Stanford University
Stanford, CA 94305-9040
klein@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305-9040
manning@cs.stanford.edu

Abstract

We present a generative model for the *unsupervised* learning of dependency structures. We also describe the multiplicative combination of this dependency model with a model of linear constituency. The product model outperforms both components on their respective evaluation metrics, giving the best published figures for unsupervised dependency parsing *and* unsupervised constituency parsing. We also demonstrate that the combined model works and is robust cross-linguistically, being able to exploit either attachment or distributional regularities that are salient in the data.

1 Introduction

The task of statistically inducing hierarchical syntactic structure over unannotated sentences of natural language has received a great deal of attention (Carroll and Charniak, 1992; Pereira and Schabes, 1992; Brill, 1993; Stolcke and Omohundro, 1994). Researchers have explored this problem for a variety of reasons: to argue empirically against the poverty of the stimulus (Clark, 2001), to use induction systems as a first stage in constructing large treebanks (van Zaanen, 2000), to build better language models (Baker, 1979; Chen, 1995), and to examine cognitive issues in language learning (Solan et al., 2003). An important distinction should be drawn between work primarily interested in the weak generative capacity of models, where modeling hierarchical structure is only useful insofar as it leads to improved models over observed structures (Baker, 1979; Chen, 1995), and work interested in the strong generative capacity of models, where the unobserved structure itself is evaluated (van Zaanen, 2000; Clark, 2001; Klein and Manning, 2002). This paper falls into the latter category; we will be inducing models of linguistic constituency and dependency with the goal of recovering linguistically plausible structures. We make no claims as to the cognitive plausibility of the induction mechanisms we present here; however, the ability of these systems to recover substantial linguistic patterns from surface yields alone does speak to the strength of support for these patterns in the data, and hence undermines arguments based on "the poverty of the stimulus" (Chomsky, 1965).

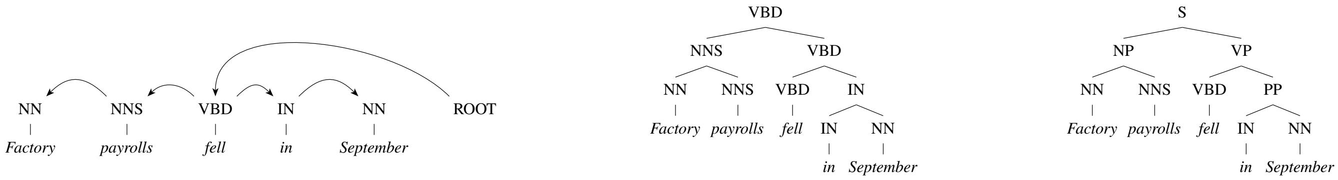
2 Unsupervised Dependency Parsing

Most recent progress in unsupervised parsing has come from tree or phrase-structure grammar based models (Clark, 2001; Klein and Manning, 2002), but there are compelling reasons to reconsider unsupervised *dependency* parsing. First, most state-of-the-art *supervised* parsers make use of specific lexical information in addition to word-class level information – perhaps lexical information could be a useful source of information for unsupervised methods. Second, a central motivation for using tree structures in computational linguistics is to enable the extraction of dependencies – function-argument and modification structures – and it might be more advantageous to induce such structures directly. Third, as we show below, for languages such as Chinese, which have few function words, and for which the definition of lexical categories is much less clear, dependency structures may be easier to detect.

2.1 Representation and Evaluation

An example dependency representation of a short sentence is shown in figure 1(a), where, following the traditional dependency grammar notation, the regent or head of a dependency is marked with the tail of the dependency arrow, and the dependent is marked with the arrowhead (Mel′čuk, 1988). It will be important in what follows to see that such a representation is isomorphic (in terms of strong generative capacity) to a restricted form of phrase structure grammar, where the set of terminals and nonterminals is identical, and every rule is of the form X → X Y or X → Y X (Miller, 1999), giving the isomorphic representation of figure 1(a) shown in figure 1(b).[1] Depending on the model, part-of-

[1]Strictly, such phrase structure trees are isomorphic not to flat dependency structures, but to specific derivations of those



(a) Classical Dependency Structure (b) Dependency Structure as CF Tree (c) CFG Structure

Figure 1: Three kinds of parse structures.

speech categories may be included in the dependency representation, as shown here, or dependencies may be directly between words. Below, we will assume an additonal reserved nonterminal ROOT, whose sole dependent is the head of the sentence. This simplifies the notation, math, and the evaluation metric.

A dependency analysis will always consist of exactly as many dependencies as there are words in the sentence. For example, in the dependency structure of figure 1(b), the dependencies are {(ROOT, *fell*), (*fell, payrolls*), (*fell, in*), (*in, September*), (*payrolls, Factory*)}. The quality of a hypothesized dependency structure can hence be evaluated by accuracy as compared to a gold-standard dependency structure, by reporting the percentage of dependencies shared between the two analyses.

In the next section, we discuss several models of dependency structure, and throughout this paper we report the accuracy of various methods at recovering gold-standard dependency parses from various corpora, detailed here. WSJ is the entire Penn English Treebank WSJ portion. WSJ10 is the subset of sentences which contained 10 words or less after the removal of punctuation. CTB10 is the sentences of the same length from the Penn Chinese treebank (v3). NEGRA10 is the same, for the German NEGRA corpus, based on the supplied conversion of the NEGRA corpus into Penn treebank format. In most of the present experiments, the provided parts-of-speech were used as the input alphabet, though we also present limited experimentation with synthetic parts-of-speech.

It is important to note that the Penn treebanks do *not* include dependency annotations; however, the automatic dependency rules from (Collins, 1999) are sufficiently accurate to be a good benchmark for unsupervised systems for the time being (though see below for specific issues). Similar head-finding rules were used for Chinese experiments. The NEGRA corpus, however, does supply hand-annotated dependency structures.

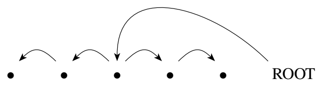


Figure 2: Dependency graph with skeleton chosen, but words not populated.

Where possible, we report an accuracy figure for both directed and undirected dependencies. Reporting undirected numbers has two advantages: first, it facilitates comparison with earlier work, and, more importantly, it allows one to partially obscure the effects of alternate analyses, such as the systematic choice between a modal and a main verb for the head of a sentence (in either case, the two verbs would be linked, but the direction would vary).

2.2 Dependency Models

The dependency induction task has received relatively little attention; the best known work is Carroll and Charniak (1992), Yuret (1998), and Paskin (2002). All systems that we are aware of operate under the assumption that the probability of a dependency structure is the product of the scores of the dependencies (attachments) in that structure. Dependencies are seen as ordered (head, dependent) pairs of words, but the score of a dependency can optionally condition on other characteristics of the structure, most often the direction of the dependency (whether the arrow points left or right).

Some notation before we present specific models: a dependency d is a pair $\langle h, a\rangle$ of a head and argument, which are words in a sentence s, in a corpus S. For uniformity of notation with section 4, words in s are specified as size-one spans of s: for example the first word would be ${}_0s_1$. A dependency structure D over a sentence is a set of dependencies (arcs) which form a planar, acyclic graph rooted at the special symbol ROOT, and in which each word in s appears as an argument exactly once. For a dependency structure D, there is an associated graph G which represents the number of words and arrows between them, without specifying the words themselves (see figure 2). A graph G and sentence s together thus determine a dependency structure. The

structures which specify orders of attachment among multiple dependents which share a common head.

Model	Dir.	Undir.
English (WSJ)		
Paskin 01		39.7
RANDOM		41.7
Charniak and Carroll 92-inspired		44.7
ADJACENT		53.2
DMV		54.4
English (WSJ10)		
RANDOM	30.1	45.6
ADJACENT	33.6	56.7
DMV	43.2	63.7
German (NEGRA10)		
RANDOM	21.8	41.5
ADJACENT	32.6	51.2
DMV	36.3	55.8
Chinese (CTB10)		
RANDOM	35.9	47.3
ADJACENT	30.2	47.3
DMV	42.5	54.2

Figure 3: Parsing performance (directed and undirected dependency accuracy) of various dependency models on various treebanks, along with baselines.

dependency structure is the object generated by all of the models that follow; the steps in the derivations vary from model to model.

Existing generative dependency models intended for unsupervised learning have chosen to first generate a word-free graph G, then populate the sentence s conditioned on G. For instance, the model of Paskin (2002), which is broadly similar to the semi-probabilistic model in Yuret (1998), first chooses a graph G uniformly at random (such as figure 2), then fills in the words, starting with a fixed root symbol (assumed to be at the rightmost end), and working down G until an entire dependency structure D is filled in (figure 1a). The corresponding probabilistic model is

$$\begin{aligned} \mathrm{P}(D) &= \mathrm{P}(s, G) \\ &= \mathrm{P}(G)\mathrm{P}(s|G) \\ &= \mathrm{P}(G) \prod_{(i,j,dir)\in G} \mathrm{P}({}_{i-1}s_i|{}_{j-1}s_j, dir)\,. \end{aligned}$$

In Paskin (2002), the distribution $\mathrm{P}(G)$ is fixed to be uniform, so the only model parameters are the conditional multinomial distributions $\mathrm{P}(a|h, dir)$ that encode which head words take which other words as arguments. The parameters for left and right arguments of a single head are completely independent, while the parameters for first and subsequent arguments in the same direction are identified.

In those experiments, the model above was trained on over 30M words of raw newswire, using EM in an entirely unsupervised fashion, and at great computational cost. However, as shown in figure 3, the resulting parser predicted dependencies at below chance level (measured by choosing a random dependency structure). This below-random performance seems to be because the model links word pairs which have high mutual information (such as occurrences of *congress* and *bill*) regardless of whether they are plausibly syntactically related. In practice, high mutual information between words is often stronger between two topically similar nouns than between, say, a preposition and its object.

One might hope that the problem with this model is that the actual lexical items are too semantically charged to represent workable units of syntactic structure. If one were to apply the Paskin (2002) model to dependency structures parameterized simply on the word-classes, the result would be isomorphic to the "dependency PCFG" models described in Carroll and Charniak (1992). In these models, Carroll and Charniak considered PCFGs with precisely the productions (discussed above) that make them isomorphic to dependency grammars, with the terminal alphabet being simply parts-of-speech. Here, the rule probabilities are equivalent to P(Y|X, *right*) and P(Y|X, *left*) respectively.[2] The actual experiments in Carroll and Charniak (1992) do not report accuracies that we can compare to, but they suggest that the learned grammars were of extremely poor quality. With hindsight, however, the main issue in their experiments appears to be not their model, but that they randomly initialized the production (attachment) probabilities. As a result, their learned grammars were of very poor quality and had high variance. However, one nice property of their structural constraint, which all dependency models share, is that the symbols in the grammar are not symmetric. Even with a grammar in which the productions are initially uniform, a symbol X can only possibly have non-zero posterior likelihood over spans which contain a matching terminal X. Therefore, one can start with uniform rewrites and let the interaction between the data and the model structure break the initial symmetry. If one recasts their experiments in this way, they achieve an accuracy of 44.7% on the Penn treebank, which is higher than choosing a random dependency structure, but lower than simply linking all adjacent words into a left-headed (and right-branching) structure (53.2%).

A huge limitation of both of the above models is that they are incapable of encoding even first-order valence facts. For example, the latter model learns that nouns to the left of the verb (usually subjects)

[2]There is another, subtle distinction: in the Paskin work, a canonical ordering of multiple attachments was fixed, while in the Carroll and Charniak work all attachment orders are considered, giving a numerical bias towards structures where heads take more than one argument.

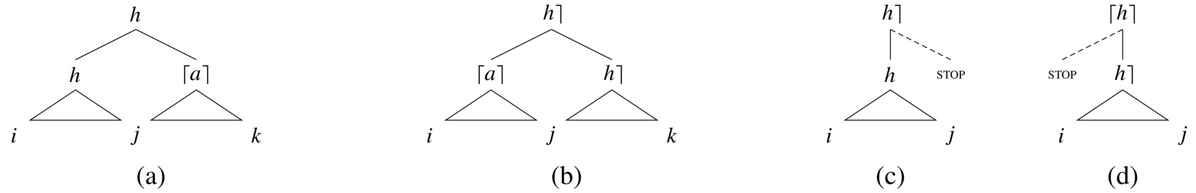


Figure 4: Dependency configurations in a lexicalized tree: (a) right attachment, (b) left attachment, (c) right stop, (d) left stop. h and a are head and argument words, respectively, while i, j, and k are positions between words.

attach to the verb. But then, given a NOUN NOUN VERB sequence, both nouns will attach to the verb – there is no way that the model can learn that verbs have exactly one subject. We now turn to an improved dependency model that addresses this problem.

3 An Improved Dependency Model

The dependency models discussed above are distinct from dependency models used inside high-performance supervised probabilistic parsers in several ways. First, in supervised models, a head outward process is modeled (Eisner, 1996; Collins, 1999). In such processes, heads generate a sequence of arguments outward to the left or right, conditioning on not only the identity of the head and direction of the attachment, but also on some notion of distance or valence. Moreover, in a head-outward model, it is natural to model stop steps, where the final argument on each side of a head is always the special symbol STOP. Models like Paskin (2002) avoid modeling STOP by generating the graph skeleton G first, uniformly at random, then populating the words of s conditioned on G. Previous work (Collins, 1999) has stressed the importance of including termination probabilities, which allows the graph structure to be generated jointly with the terminal words, precisely because it does allow the modeling of required dependents.

We propose a simple head-outward dependency model over word classes which includes a model of valence, which we call *DMV* (for *dependency model with valence*). We begin at the ROOT. In the standard way, each head generates a series of non-STOP arguments to one side, then a STOP argument to that side, then non-STOP arguments to the other side, then a second STOP.

For example, in the dependency structure in figure 1, we first generate a single child of ROOT, here *fell*. Then we recurse to the subtree under *fell*. This subtree begins with generating the right argument *in*. We then recurse to the subtree under *in* (generating *September* to the right, a right STOP, and a left STOP). Since there are no more right arguments after *in*, its right STOP is generated, and the process moves on to the left arguments of *fell*.

In this process, there are two kinds of derivation events, whose local probability factors constitute the model's parameters. First, there is the decision at any point whether to terminate (generate STOP) or not: $\mathrm{P}_{\text{STOP}}(\text{STOP}|h, dir, adj)$. This is a binary decision conditioned on three things: the head h, the direction (generating to the left or right of the head), and the adjacency (whether or not an argument has been generated yet in the current direction, a binary variable). The stopping decision is estimated directly, with no smoothing. If a stop is generated, no more arguments are generated for the current head to the current side. If the current head's argument generation does not stop, another argument is chosen using: $\mathrm{P}_{\text{CHOOSE}}(a|h, dir)$. Here, the argument is picked conditionally on the identity of the head (which, recall, is a word class) and the direction. This term, also, is not smoothed in any way. Adjacency has no effect on the identity of the argument, only on the likelihood of termination. After an argument is generated, its subtree in the dependency structure is recursively generated.

Formally, for a dependency structure D, let each word h have left dependents $deps_D(h, l)$ and right dependents $deps_D(h, r)$. The following recursion defines the probability of the fragment $D(h)$ of the dependency tree rooted at h:

$$\mathrm{P}(D(h)) = \prod_{dir \in \{l,r\}} \prod_{a \in deps_D(h,dir)} \mathrm{P}_{\text{STOP}}(\neg\text{STOP}|h, dir, adj) \\ \mathrm{P}_{\text{CHOOSE}}(a|h, dir)\mathrm{P}(D(a)) \\ \mathrm{P}_{\text{STOP}}(\text{STOP}|h, dir, adj)$$

One can view a structure generated by this derivational process as a "lexicalized" tree composed of the local binary and unary context-free configurations shown in figure 4.[3] Each configuration equivalently represents either a head-outward derivation step or a context-free rewrite rule. There are four such configurations. Figure 4(a) shows a head h

[3]It is lexicalized in the sense that the labels in the tree are derived from terminal symbols, but in our experiments the terminals were word classes, not individual lexical items.

taking a right argument a. The tree headed by h contains h itself, possibly some right arguments of h, but no left arguments of h (they attach after all the right arguments). The tree headed by a contains a itself, along with all of its left and right children. Figure 4(b) shows a head h taking a left argument a – the tree headed by h must have already generated its right stop to do so. Figure 4(c) and figure 4(d) show the *sealing* operations, where STOP derivation steps are generated. The left and right marks on node labels represent left and right STOPs that have been generated.[4]

The basic inside-outside algorithm (Baker, 1979) can be used for re-estimation. For each sentence $s \in S$, it gives us $c_s(x : i, j)$, the expected fraction of parses of s with a node labeled x extending from position i to position j. The model can be re-estimated from these counts. For example, to re-estimate an entry of $P_{\text{STOP}}(\text{STOP}|h, \textit{left}, \textit{non-adj})$ according to a current model Θ, we calculate two quantities.[5] The first is the (expected) number of trees headed by $h\rceil$ whose rightmost edge i is strictly left of h. The second is the number of trees headed by $\lceil h \rceil$ with rightmost edge i strictly left of h. The ratio is the MLE of that local probability factor:

$$P_{\text{STOP}}(\text{STOP}|h, \textit{left}, \textit{non-adj}) = \frac{\sum_{s\in S}\sum_{i<loc(h)}\sum_k c(h\rceil : i, k)}{\sum_{s\in S}\sum_{i<loc(h)}\sum_k c(\lceil h\rceil : i, k)}$$

This can be intuitively thought of as the relative number of times a tree headed by h had already taken at least one argument to the left, had an opportunity to take another, but didn't.[6]

Initialization is important to the success of any local search procedure. We chose to initialize EM not with an initial model, but with an initial guess at posterior distributions over dependency structures (completions). For the first-round, we constructed a somewhat ad-hoc "harmonic" completion where all non-ROOT words took the same number of arguments, and each took other words as arguments in inverse proportion to (a constant plus) the distance between them. The ROOT always had a single argument and took each word with equal probability. This structure had two advantages: first, when testing multiple models, it is easier to start them all off in a common way by beginning with an M-step, and, second, it allowed us to point the model in the vague general direction of what linguistic dependency structures should look like.

On the WSJ10 corpus, the DMV model recovers a substantial fraction of the broad dependency trends: 43.2% of guessed directed dependencies were correct (63.7% ignoring direction). To our knowledge, this is the first published result to break the adjacent-word heuristic (at 33.6% for this corpus). Verbs are the sentence heads, prepositions take following noun phrases as arguments, adverbs attach to verbs, and so on. The most common source of discrepancy between the test dependencies and the model's guesses is a result of the model systematically choosing determiners as the heads of noun phrases, while the test trees have the rightmost noun as the head. The model's choice is supported by a good deal of linguistic research (Abney, 1987), and is sufficiently systematic that we also report the scores where the NP headship rule is changed to percolate determiners when present. On this adjusted metric, the score jumps hugely to 55.7% directed (and 67.9% undirected).

This model also works on German and Chinese at above-baseline levels (55.8% and 54.2% undirected, respectively), with no modifications whatsoever. In German, the largest source of errors is also the systematic postulation of determiner-headed noun-phrases. In Chinese, the primary mismatch is that subjects are considered to be the heads of sentences rather than verbs.

This dependency induction model is reasonably successful. However, our intuition is still that the model can be improved by paying more attention to syntactic constituency. To this end, after briefly recapping the model of Klein and Manning (2002), we present a combined model that exploits dependencies and constituencies. As we will see, this combined model finds correct dependencies more successfully than the model above, and finds constituents more successfully than the model of Klein and Manning (2002).

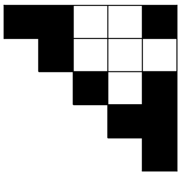
4 Distributional Constituency Induction

In linear distributional clustering, items (e.g., words or word sequences) are represented by characteristic distributions over their linear contexts (e.g., multinomial models over the preceding and following words, see figure 5). These context distributions are then clustered in some way, often using standard

[4] Note that the asymmetry of the attachment rules enforces the right-before-left attachment convention. This is harmless and arbitrary as far as dependency evaluations go, but imposes an x-bar-like structure on the constituency assertions made by this model. This bias/constraint is dealt with in section 5.

[5] To simplify notation, we assume each word h occurs at most one time in a given sentence, between indexes $loc(h)$ and $loc(h) + 1$).

[6] As a final note, in addition to enforcing the right-argument-first convention, we constrained ROOT to have at most a single dependent, by a similar device.



(a)

Span	Label	Constituent	Context
⟨0,5⟩	S	NN NNS VBD IN NN	◇ – ◇
⟨0,2⟩	NP	NN NNS	◇ – VBD
⟨2,5⟩	VP	VBD IN NN	NNS – ◇
⟨3,5⟩	PP	IN NN	VBD – ◇
⟨0,1⟩	NN	NN	◇ – NNS
⟨1,2⟩	NNS	NNS	NN – VBD
⟨2,3⟩	VBD	VBD	NNS – IN
⟨3,4⟩	IN	IN	VBD – NN
⟨4,5⟩	NN	NNS	IN – ◇

(b)

Figure 5: The CCM model's generative process for the sentence in figure 1. (a) A binary tree-equivalent bracketing is chosen at random. (b) *Each* span generates its yield and context (empty spans not shown here). Derivations which are not coherent are given mass zero.

data clustering methods. In the most common case, the items are words, and one uses distributions over adjacent words to induce word classes. Previous work has shown that even this quite simple representation allows the induction of quite high quality word classes, largely corresponding to traditional parts of speech (Finch, 1993; Schütze, 1995; Clark, 2000). A typical pattern would be that *stocks* and *treasuries* both frequently occur before the words *fell* and *rose*, and might therefore be put into the same class.

Clark (2001) and Klein and Manning (2002) show that this approach can be successfully used for discovering syntactic constituents as well. However, as one might expect, it is easier to cluster word sequences (or word class sequences) than to tell how to put them together into trees. In particular, if one is given all contiguous subsequences (*subspans*) from a corpus of sentences, most natural clusters will not represent valid constituents (to the extent that constituency of a non-situated sequence is even a well-formed notion). For example, it is easy enough to discover that DET N and DET ADJ N are similar and that V PREP DET and V PREP DET ADJ are similar, but it is much less clear how to discover that the former pair are generally constituents while the latter pair are generally not. In Klein and Manning (2002), we proposed a *constituent-context model* (CCM) which solves this problem by building constituency decisions directly into the distributional model, by earmarking a single cluster d for non-constituents. During the calculation of cluster assignments, only a non-crossing subset of the observed word sequences can be assigned to other, constituent clusters. This integrated approach is empirically successful.

The CCM works as follows. Sentences are given as sequences s of word classes (parts-of-speech or otherwise). One imagines each sentence as a list of the $O(n^2)$ index pairs $\langle i, j\rangle$, each followed by the corresponding subspan ${}_is_j$ and linear context ${}_{i-1}s_i \sim {}_js_{j+1}$ (see figure 5). The model generates all constituent-context pairs, span by span.

The first stage is to choose a *bracketing* B for the sentence, which is a maximal non-crossing subset of the spans (equivalent to a binary tree). In the basic model, $\mathrm{P}(B)$ is uniform over binary trees. Then, for each $\langle i, j\rangle$, the subspan and context pair $({}_is_j,\ {}_{i-1}s_i \sim {}_js_{j+1})$ is generated via a class-conditional independence model:

$$\mathrm{P}(s, B) = \mathrm{P}(B) \prod_{\langle i,j\rangle} \mathrm{P}({}_is_j|b_{ij})\mathrm{P}({}_{i-1}s_i \sim {}_js_{j+1}|b_{ij})$$

That is, all spans guess their sequences and contexts given only a constituency decision b.[7]

This is a model $\mathrm{P}(s, B)$ over hidden bracketings and observed sentences, and it is estimated via EM to maximize the sentence likelihoods $\mathrm{P}(s)$ over the training corpus. Figure 6 shows the accuracy of the CCM model not only on English but for the Chinese and German corpora discussed above.[8] Results are reported at convergence; for the English case, F_1 is monotonic during training, while for the others, there is an earlier peak.

Also shown is an upper bound (the target trees are not all binary and so any all-binary system will over-propose constituents). Klein and Manning (2002) gives comparative numbers showing that the basic CCM outperforms other recent systems on the ATIS corpus (which many other constituency induction systems have reported on). While absolute numbers are hard to compare across corpora, all the systems compared to in Klein and Manning (2002) parsed below a right-branching baseline, while the CCM is substantially above it.

5 A Combined Model

The two models described above have some common ground. Both can be seen as models over lexicalized trees composed of the configurations in figure 4. For the DMV, it is already a model over these structures. At the "attachment" rewrite for the CCM

[7]As is typical of distributional clustering, positions in the corpus can get generated multiple times. Since derivations need not be consistent, the entire model is mass deficient when viewed as a model over sentences.

[8]In Klein and Manning (2002), we reported results using unlabeled bracketing statistics which gave no credit for brackets which spanned the entire sentence (raising the scores) but macro-averaged over sentences (lowering the scores). The numbers here hew more closely to the standard methods used for evaluating supervised parsers, by being micro-averaged and including full-span brackets. However, the scores are, overall, approximately the same.

in (a/b), we assign the quantity:

$$\frac{\mathrm{P}({}_i s_k | true)\mathrm{P}({}_{i-1}s_i \sim {}_k s_{k+1} | true)}{\mathrm{P}({}_i s_k | false)\mathrm{P}({}_{i-1}s_i \sim {}_k s_{k+1} | false)}$$

which is the odds ratio of generating the subsequence and context for span $\langle i, k\rangle$ as a constituent as opposed to a non-constituent. If we multiply all trees' attachment scores by

$$\prod_{\langle i,j\rangle} \mathrm{P}({}_i s_j | false)\mathrm{P}({}_{i-1}s_i \sim {}_j s_{j+1} | false)$$

the denominators of the odds ratios cancel, and we are left with each tree being assigned the probability it would have received under the CCM.[9]

In this way, both models can be seen as generating either constituency or dependency structures. Of course, the CCM will generate fairly random dependency structures (constrained only by bracketings). Getting constituency structures from the DMV is also problematic, because the choice of which side to first attach arguments on has ramifications on constituency – it forces x-bar-like structures – even though it is an arbitrary convention as far as dependency evaluations are concerned. For example, if we attach right arguments first, then a verb with a left subject and a right object will attach the object first, giving traditional VPs, while the other attachment order gives subject-verb groups. To avoid this bias, we alter the DMV in the following ways. When using the dependency model alone, we allow each word to have even probability for either generation order (but in each actual head derivation, only one order occurs). When using the models together, better performance was obtained by releasing the one-side-attaching-first requirement entirely.

In figure 6, we give the behavior of the CCM constituency model and the DMV dependency model on both constituency and dependency induction. Unsurprisingly, their strengths are complementary. The CCM is better at recovering constituency, and the dependency model is better at recovering dependency structures. It is reasonable to hope that a combination model might exhibit the best of both. In the supervised parsing domain, for example, scoring a lexicalized tree with the product of a simple lexical dependency model and a PCFG model can outperform each factor on its respective metric (Klein and Manning, 2003).

[9]This scoring function as described is not a generative model over lexicalized trees, because it has no generation step at which nodes' lexical heads are chosen. This can be corrected by multiplying in a "head choice" factor of $1/(k-j)$ at each final "sealing" configuration (d). In practice, this correction factor was harmful for the model combination, since it duplicated a strength of the dependency model, badly.

Model	UP	UR	UF$_1$	Dir	Undir
English (WSJ10 – 7422 Sentences)					
LBRANCH/RHEAD	25.6	32.6	28.7	33.6	56.7
RANDOM	31.0	39.4	34.7	30.1	45.6
RBRANCH/LHEAD	55.1	70.0	61.7	24.0	55.9
DMV	46.6	59.2	52.1	43.2	62.7
CCM	64.2	81.6	71.9	23.8	43.3
DMV+CCM (POS)	**69.3**	**88.0**	**77.6**	**47.5**	**64.5**
DMV+CCM (DISTR.)	65.2	82.8	72.9	42.3	60.4
UBOUND	78.8	100.0	88.1	100.0	100.0
German (NEGRA10 – 2175 Sentences)					
LBRANCH/RHEAD	27.4	48.8	35.1	32.6	51.2
RANDOM	27.9	49.6	35.7	21.8	41.5
RBRANCH/LHEAD	33.8	60.1	43.3	21.0	49.9
DMV	38.4	69.5	49.5	40.0	57.8
CCM	48.1	85.5	61.6	25.5	44.9
DMV+CCM	**49.6**	**89.7**	**63.9**	**50.6**	**64.7**
UBOUND	56.3	100.0	72.1	100.0	100.0
Chinese (CTB10 – 2437 Sentences)					
LBRANCH/RHEAD	26.3	48.8	34.2	30.2	43.9
RANDOM	27.3	50.7	35.5	35.9	47.3
RBRANCH/LHEAD	29.0	53.9	37.8	14.2	41.5
DMV	**35.9**	**66.7**	**46.7**	42.5	54.2
CCM	34.6	64.3	45.0	23.8	40.5
DMV+CCM	33.3	62.0	43.3	**55.2**	**60.3**
UBOUND	53.9	100.0	70.1	100.0	100.0

Figure 6: Parsing performance of the combined model on various treebanks, along with baselines.

In the combined model, we score each tree with the product of the probabilities from the individual models above. We use the inside-outside algorithm to sum over all lexicalized trees, similar to the situation in section 3. The tree configurations are shown in figure 4. For each configuration, the relevant scores from each model are multiplied together. For example, consider figure 4(a). From the CCM we generate ${}_i s_k$ as a constituent and its corresponding context. From the dependency model, we pay the cost of h taking a as a right argument ($\mathrm{P}_{\text{CHOOSE}}$), as well as the cost of not stopping (P_{STOP}). The other configurations are similar. We then run the inside-outside algorithm over this product model. From the results, we can extract the statistics needed to re-estimate both individual models.[10]

The models in combination were intitialized in the same way as when they were run individually. Sufficient statistics were separately taken off these individual completions. From then on, the resulting models were used together during re-estimation.

Figure 6 summarizes the results. The combined model beats the CCM on English F_1: 77.6 vs. 71.9. The figure also shows the combination model's score when using word classes which were induced entirely automatically, using the simplest distributional clustering method of Schütze (1995). These classes show some degradation, e.g. 72.9 F_1, but it

[10]The product, like the CCM itself, is mass-deficient.

is worth noting that these totally unsupervised numbers are better than the performance of the CCM model of Klein and Manning (2002) running off of Penn treebank word classes. Again, if we modify the gold standard so as to make determiners the head of NPs, then this model with distributional tags scores 50.6% on directed and 64.8% on undirected dependency accuracy.

On the German data, the combination again outperforms each factor alone, though while the combination was most helpful at boosting constituency quality for English, for German it provided a larger boost to the dependency structures. Finally, on the Chinese data, the combination did substantially boost dependency accuracy over either single factor, but actually suffered a small drop in constituency.[11] Overall, the combination is able to combine the individual factors in an effective way.

6 Conclusion

We have presented a successful new dependency-based model for the unsupervised induction of syntactic structure, which picks up the key ideas that have made dependency models successful in supervised statistical parsing work. We proceeded to show that it works cross-linguistically. We then demonstrated how this model could be combined with the previous best constituent-induction model to produce a combination which, in general, substantially outperforms either individual model, on either metric. A key reason that these models are capable of recovering structure more accurately than previous work is that they minimize the amount of hidden structure that must be induced. In particular, neither model attempts to learn intermediate, recursive categories with no direct connection to surface statistics. Our results here are just on the ungrounded induction of syntactic structure. Nonetheless, we see the investigation of what patterns can be recovered from corpora as important, both from a computational perspective and from a philosophical one. It demonstrates that the broad constituent and dependency structure of a language can be recovered quite successfully (individually or, more effectively, jointly) from a very modest amount of training data.

7 Acknowledgements

This work was supported by a Microsoft Graduate Research Fellowship to the first author and by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program. This work also benefited from an enormous amount of useful feedback, from many audiences and individuals.

[11]This seems to be partially due to the large number of unanalyzed fragments in the Chinese gold standard, which leave a very large fraction of the posited bracketings completely unjudged.

References

Stephen P. Abney. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, MIT.

James K. Baker. 1979. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Eric Brill. 1993. Automatic grammar induction and parsing free text: A transformation-based approach. In *ACL 31*, pages 259–265.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In Carl Weir, Stephen Abney, Ralph Grishman, and Ralph Weischedel, editors, *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13. AAAI Press, Menlo Park, CA.

Stanley F. Chen. 1995. Bayesian grammar induction for language modeling. In *ACL 33*, pages 228–235.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *The Fourth Conference on Natural Language Learning*.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *The Fifth Conference on Natural Language Learning*.

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

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 16*, pages 340–345.

Steven Paul Finch. 1993. *Finding Structure in Language*. Ph.D. thesis, University of Edinburgh.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL 40*, pages 128–135.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Igor Aleksandrovich Mel′čuk. 1988. *Dependency Syntax: theory and practice*. State University of New York Press, Albany, NY.

Philip H. Miller. 1999. *Strong Generative Capacity*. CSLI Publications, Stanford, CA.

Mark A. Paskin. 2002. Grammatical bigrams. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL 30*, pages 128–135.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *EACL 7*, pages 141–148.

Zach Solan, Eytan Ruppin, David Horn, and Shimon Edelman. 2003. Automatic acquisition and efficient representation of syntactic structures. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

Andreas Stolcke and Stephen M. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications: Proceedings of the Second International Colloquium on Grammatical Inference*. Springer Verlag.

Menno van Zaanen. 2000. ABL: Alignment-based learning. In *COLING 18*, pages 961–967.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, MIT.