Three Generative, Lexicalised Models for Statistical Parsing

Michael Collins*
Dept. of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, 19104, U.S.A.
mcollins@gradient.cis.upenn.edu

Abstract

In this paper we first propose a new statistical parsing model, which is a generative model of lexicalised context-free grammar. We then extend the model to include a probabilistic treatment of both subcategorisation and wh-movement. Results on Wall Street Journal text show that the parser performs at 88.1/87.5% constituent precision/recall, an average improvement of 2.3% over (Collins 96).

1 Introduction

Generative models of syntax have been central in linguistics since they were introduced in (Chomsky 57). Each sentence-tree pair (S,T) in a language has an associated top-down derivation consisting of a sequence of rule applications of a grammar. These models can be extended to be statistical by defining probability distributions at points of non-determinism in the derivations, thereby assigning a probability $\mathcal{P}(S,T)$ to each (S,T) pair. Probabilistic context free grammar (Booth and Thompson 73) was an early example of a statistical grammar. A PCFG can be lexicalised by associating a head-word with each non-terminal in a parse tree; thus far, (Magerman 95; Jelinek et al. 94) and (Collins 96), which both make heavy use of lexical information, have reported the best statistical parsing performance on Wall Street Journal text. Neither of these models is generative, instead they both estimate $\mathcal{P}(T \mid S)$ directly.

This paper proposes three new parsing models. **Model 1** is essentially a generative version of the model described in (Collins 96). In **Model 2**, we extend the parser to make the complement/adjunct distinction by adding probabilities over subcategorisation frames for head-words. In **Model 3** we give a probabilistic treatment of wh-movement, which is derived from the analysis given in Generalized Phrase Structure Grammar (Gazdar et al. 95). The work makes two advances over previous models: First, Model 1 performs significantly better than (Collins 96), and Models 2 and 3 give further improvements — our final results are 88.1/87.5% constituent precision/recall, an average improvement of 2.3% over (Collins 96). Second, the parsers in (Collins 96) and (Magerman 95; Jelinek et al. 94) produce trees without information about wh-movement or subcategorisation. Most NLP applications will need this information to extract predicate-argument structure from parse trees.

In the remainder of this paper we describe the 3 models in section 2, discuss practical issues in section 3, give results in section 4, and give conclusions in section 5.

2 The Three Parsing Models

2.1 Model 1

In general, a statistical parsing model defines the conditional probability, $\mathcal{P}(T \mid S)$, for each candidate parse tree T for a sentence S. The parser itself is an algorithm which searches for the tree, T_{best}, that maximises $\mathcal{P}(T \mid S)$. A generative model uses the observation that maximising $\mathcal{P}(T,S)$ is equivalent to maximising $\mathcal{P}(T \mid S)$: [1]

$$\begin{aligned} T_{best} &= \arg\max_T \mathcal{P}(T \mid S) = \arg\max_T \frac{\mathcal{P}(T,S)}{\mathcal{P}(S)} \\ &= \arg\max_T \mathcal{P}(T,S) \qquad (1) \end{aligned}$$

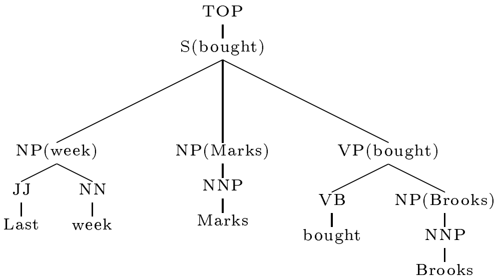
$\mathcal{P}(T,S)$ is then estimated by attaching probabilities to a top-down derivation of the tree. In a PCFG, for a tree derived by n applications of context-free re-write rules $LHS_i \Rightarrow RHS_i$, $1 \le i \le n$,

$$\mathcal{P}(T,S) = \prod_{i=1..n} \mathcal{P}(RHS_i \mid LHS_i) \qquad (2)$$

The re-write rules are either internal to the tree, where LHS is a non-terminal and RHS is a string

*This research was supported by ARPA Grant N6600194-C6043.

[1] $\mathcal{P}(S)$ is constant, hence maximising $\frac{\mathcal{P}(T,S)}{\mathcal{P}(S)}$ is equivalent to maximising $\mathcal{P}(T,S)$.




```
TOP          ->   S(bought)
S(bought)    ->   NP(week)     NP(Marks)    VP(bought)
NP(week)     ->   JJ(Last)     NN(week)
NP(Marks)    ->   NNP(Marks)
VP(bought)   ->   VB(bought)   NP(Brooks)
NP(Brooks)   ->   NNP(Brooks)
```

Figure 1: A lexicalised parse tree, and a list of the rules it contains. For brevity we omit the POS tag associated with each word.

of one or more non-terminals; or lexical, where LHS is a part of speech tag and RHS is a word.

A PCFG can be lexicalised[2] by associating a word w and a part-of-speech (POS) tag t with each non-terminal X in the tree. Thus we write a non-terminal as $X(x)$, where $x = \langle w, t \rangle$, and X is a constituent label. Each rule now has the form[3]:

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m) \quad (3)$$

H is the head-child of the phrase, which inherits the head-word h from its parent P. $L_1...L_n$ and $R_1...R_m$ are left and right modifiers of H. Either n or m may be zero, and $n = m = 0$ for unary rules. Figure 1 shows a tree which will be used as an example throughout this paper.

The addition of lexical heads leads to an enormous number of potential rules, making direct estimation of $\mathcal{P}(RHS \mid LHS)$ infeasible because of sparse data problems. We decompose the generation of the `RHS` of a rule such as (3), given the `LHS`, into three steps — first generating the head, then making the independence assumptions that the left and right modifiers are generated by separate 0^{th}-order markov processes[4]:

1. Generate the head constituent label of the phrase, with probability $\mathcal{P}_H(H \mid P, h)$.
2. Generate modifiers to the right of the head with probability $\prod_{i=1..m+1} \mathcal{P}_R(R_i(r_i) \mid P, h, H)$. $R_{m+1}(r_{m+1})$ is defined as $STOP$ — the $STOP$ symbol is added to the vocabulary of non-terminals, and the model stops generating right modifiers when it is generated.
3. Generate modifiers to the left of the head with probability $\prod_{i=1..n+1} \mathcal{P}_L(L_i(l_i) \mid P, h, H)$, where $L_{n+1}(l_{n+1}) = STOP$.

For example, the probability of the rule `S(bought) -> NP(week) NP(Marks) VP(bought)` would be estimated as

$$\mathcal{P}_h(\texttt{VP} \mid \texttt{S,bought}) \times \mathcal{P}_l(\texttt{NP(Marks)} \mid \texttt{S,VP,bought}) \times \mathcal{P}_l(\texttt{NP(week)} \mid \texttt{S,VP,bought}) \times \mathcal{P}_l(\texttt{STOP} \mid \texttt{S,VP,bought}) \times \mathcal{P}_r(\texttt{STOP} \mid \texttt{S,VP,bought})$$

We have made the 0^{th} order markov assumptions

$$\mathcal{P}_l(L_i(l_i) \mid H, P, h, L_1(l_1)...L_{i-1}(l_{i-1})) = \mathcal{P}_l(L_i(l_i) \mid H, P, h) \quad (4)$$
$$\mathcal{P}_r(R_i(r_i) \mid H, P, h, R_1(r_1)...R_{i-1}(r_{i-1})) = \mathcal{P}_r(R_i(r_i) \mid H, P, h) \quad (5)$$

but in general the probabilities could be conditioned on any of the preceding modifiers. In fact, if the derivation order is fixed to be depth-first — that is, each modifier recursively generates the sub-tree below it before the next modifier is generated — then the model can also condition on any structure *below* the preceding modifiers. For the moment we exploit this by making the approximations

$$\mathcal{P}_l(L_i(l_i) \mid H, P, h, L_1(l_1)...L_{i-1}(l_{i-1})) = \mathcal{P}_l(L_i(l_i) \mid H, P, h, distance_l(i-1)) \quad (6)$$
$$\mathcal{P}_r(R_i(r_i) \mid H, P, h, R_1(r_1)...R_{i-1}(r_{i-1})) = \mathcal{P}_r(R_i(r_i) \mid H, P, h, distance_r(i-1)) \quad (7)$$

where $distance_l$ and $distance_r$ are functions of the surface string from the head word to the edge of the constituent (see figure 2). The distance measure is the same as in (Collins 96), a vector with the following 3 elements: (1) is the string of zero length? (Allowing the model to learn a preference for right-branching structures); (2) does the string contain a

[2] We find lexical heads in Penn treebank data using rules which are similar to those used by (Magerman 95; Jelinek et al. 94).

[3] With the exception of the top rule in the tree, which has the form `TOP` $\rightarrow$ `H(h)`.

[4] An exception is the first rule in the tree, `TOP` $\rightarrow$ `H(h)`, which has probability $P_{TOP}(H, h|TOP)$

verb? (Allowing the model to learn a preference for modification of the most recent verb). (3) Does the string contain 0, 1, 2 or $>$ 2 commas? (where a comma is anything tagged as "," or ":").

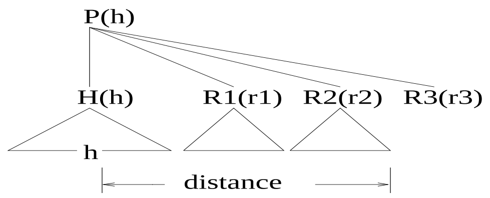


Figure 2: The next child, $R_3(r_3)$, is generated with probability $\mathcal{P}(R_3(r_3) \mid P, H, h, distance_r(2))$. The *distance* is a function of the surface string from the word after h to the last word of R_2, inclusive. In principle the model could condition on any structure dominated by H, R_1 or R_2.

2.2 Model 2: The complement/adjunct distinction and subcategorisation

The tree in figure 1 is an example of the importance of the complement/adjunct distinction. It would be useful to identify "Marks" as a subject, and "Last week" as an adjunct (temporal modifier), but this distinction is not made in the tree, as both NPs are in the same position[5] (sisters to a VP under an S node). From here on we will identify complements by attaching a "-C" suffix to non-terminals — figure 3 gives an example tree.

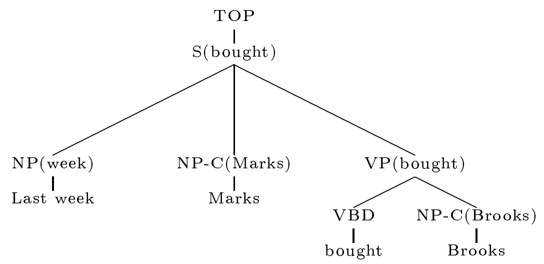


Figure 3: A tree with the "-C" suffix used to identify complements. "Marks" and "Brooks" are in subject and object position respectively. "Last week" is an adjunct.

A post-processing stage could add this detail to the parser output, but we give two reasons for making the distinction while parsing: First, identifying complements is complex enough to warrant a probabilistic treatment. Lexical information is needed — for example, knowledge that "week" is likely to be a temporal modifier. Knowledge about subcategorisation preferences — for example that a verb takes exactly one subject — is also required. These problems are not restricted to NPs, compare "The spokeswoman said (SBAR that the asbestos was dangerous)" vs. "Bonds beat short-term investments (SBAR because the market is down)", where an SBAR headed by "that" is a complement, but an SBAR headed by "because" is an adjunct.

The second reason for making the complement/adjunct distinction while parsing is that it may help parsing accuracy. The assumption that complements are generated independently of each other often leads to incorrect parses — see figure 4 for further explanation.

[5]Except "Marks" is closer to the VP, but note that "Marks" is also the subject in "Marks last week bought Brooks".

2.2.1 Identifying Complements and Adjuncts in the Penn Treebank

We add the "-C" suffix to all non-terminals in training data which satisfy the following conditions:

1. The non-terminal must be: (1) an NP, SBAR, or S whose parent is an S; (2) an NP, SBAR, S, or VP whose parent is a VP; or (3) an S whose parent is an SBAR.

2. The non-terminal must *not* have one of the following semantic tags: ADV, VOC, BNF, DIR, EXT, LOC, MNR, TMP, CLR or PRP. See (Marcus et al. 94) for an explanation of what these tags signify. For example, the NP "Last week" in figure 1 would have the TMP (temporal) tag; and the SBAR in "(SBAR because the market is down)", would have the ADV (adverbial) tag.

In addition, the first child following the head of a prepositional phrase is marked as a complement.

2.2.2 Probabilities over Subcategorisation Frames

The model could be retrained on training data with the enhanced set of non-terminals, and it might learn the lexical properties which distinguish complements and adjuncts ("Marks" vs "week", or "that" vs. "because"). However, it would still suffer from the bad independence assumptions illustrated in figure 4. To solve these kinds of problems, the generative process is extended to include a probabilistic choice of left and right subcategorisation frames:

1. Choose a head H with probability $\mathcal{P}_H(H \mid P, h)$.

2. Choose left and right subcat frames, LC and RC, with probabilities $\mathcal{P}_{lc}(LC \mid P, H, h)$ and

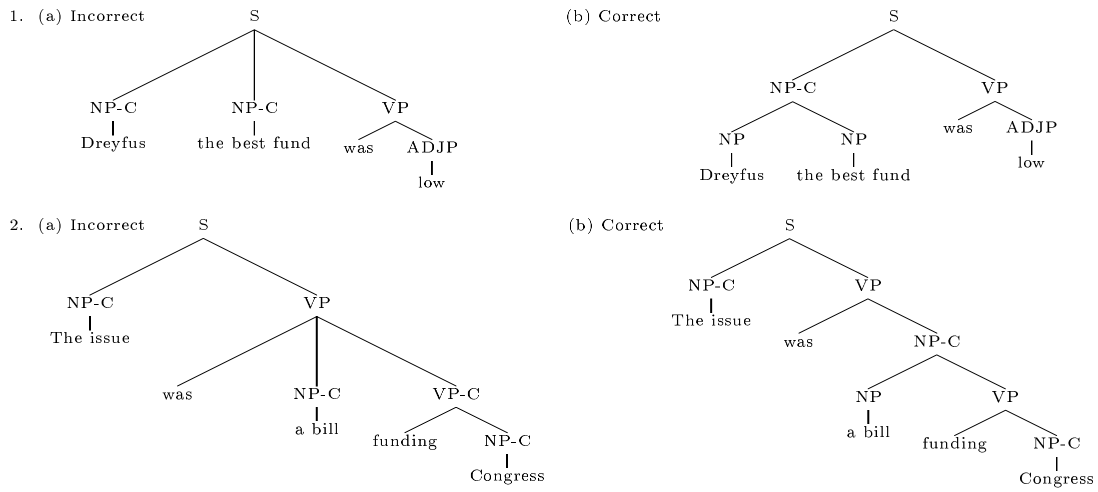


Figure 4: Two examples where the assumption that modifiers are generated independently of each other leads to errors. In (1) the probability of generating both "Dreyfus" and "fund" as subjects, $\mathcal{P}(\texttt{NP-C(Dreyfus)} \mid \texttt{S,VP,was}) * \mathcal{P}(\texttt{NP-C(fund)} \mid \texttt{S,VP,was})$ is unreasonably high. (2) is similar: $\mathcal{P}(\texttt{NP-C(bill),VP-C(funding)}|\texttt{VP,VB,was}) = \mathcal{P}(\texttt{NP-C(bill)}|\texttt{VP,VB,was}) * \mathcal{P}(\texttt{VP-C(funding)}|\texttt{VP,VB,was})$ is a bad independence assumption.

$\mathcal{P}_{rc}(RC \mid P, H, h)$. Each subcat frame is a multiset[6] specifying the complements which the head requires in its left or right modifiers.

3. Generate the left and right modifiers with probabilities $\mathcal{P}_l(L_i, l_i \mid H, P, h, distance_l(i-1), LC)$ and $\mathcal{P}_r(R_i, r_i \mid H, P, h, distance_r(i-1), RC)$ respectively. Thus the subcat requirements are added to the conditioning context. As complements are generated they are removed from the appropriate subcat multiset. Most importantly, the probability of generating the $STOP$ symbol will be 0 when the subcat frame is *non-empty*, and the probability of generating a complement will be 0 when it is not in the subcat frame; thus all and only the required complements will be generated.

The probability of the phrase `S(bought) -> NP(week) NP-C(Marks) VP(bought)` is now:

$\mathcal{P}_h(\texttt{VP} \mid \texttt{S,bought}) \times$
$\mathcal{P}_{lc}(\{\texttt{NP-C}\} \mid \texttt{S,VP,bought}) \times \mathcal{P}_{rc}(\{\} \mid \texttt{S,VP,bought}) \times$
$\mathcal{P}_l(\texttt{NP-C(Marks)} \mid \texttt{S,VP,bought}, \{\texttt{NP-C}\}) \times$
$\mathcal{P}_l(\texttt{NP(week)} \mid \texttt{S,VP,bought}, \{\}) \times$
$\mathcal{P}_l(\texttt{STOP} \mid \texttt{S,VP,bought}, \{\}) \times$
$\mathcal{P}_r(\texttt{STOP} \mid \texttt{S,VP,bought}, \{\})$

Here the head initially decides to take a single `NP-C` (subject) to its left, and no complements to its right. `NP-C(Marks)` is immediately generated as the required subject, and `NP-C` is removed from LC, leaving it empty when the next modifier, `NP(week)` is generated. The incorrect structures in figure 4 should now have low probability because $\mathcal{P}_{lc}(\{\texttt{NP-C,NP-C}\} \mid \texttt{S,VP,bought})$ and $\mathcal{P}_{rc}(\{\texttt{NP-C,VP-C}\} \mid \texttt{VP,VB,was})$ are small.

2.3 Model 3: Traces and Wh-Movement

Another obstacle to extracting predicate-argument structure from parse trees is wh-movement. This section describes a probabilistic treatment of extraction from relative clauses. Noun phrases are most often extracted from subject position, object position, or from within PPs:

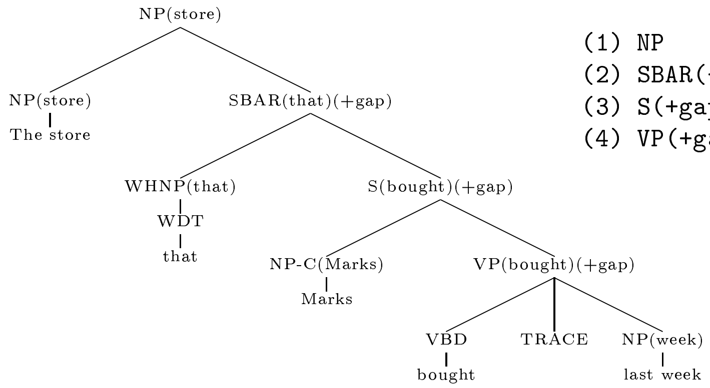
Example 1 *The store (SBAR which TRACE bought Brooks Brothers)*

Example 2 *The store (SBAR which Marks bought TRACE)*

Example 3 *The store (SBAR which Marks bought Brooks Brothers from TRACE)*

It might be possible to write rule-based patterns which identify traces in a parse tree. However, we argue again that this task is best integrated into the parser: the task is complex enough to warrant a probabilistic treatment, and integration may help parsing accuracy. A couple of complexities are that modification by an SBAR does not always involve extraction (e.g., "the fact (SBAR that besoboru is

[6] A multiset, or bag, is a set which may contain duplicate non-terminal labels.




```
(1) NP           ->  NP     SBAR(+gap)
(2) SBAR(+gap)   ->  WHNP   S-C(+gap)
(3) S(+gap)      ->  NP-C   VP(+gap)
(4) VP(+gap)     ->  VB     TRACE NP
```

Figure 5: A *+gap* feature can be added to non-terminals to describe NP extraction. The top-level NP initially generates an SBAR modifier, but specifies that it must contain an NP trace by adding the *+gap* feature. The gap is then passed down through the tree, until it is discharged as a $TRACE$ complement to the right of *bought*.

played with a ball and a bat)"), and it is not uncommon for extraction to occur through several constituents, (e.g., "The changes (SBAR that he said the government was prepared to make TRACE)").

The second reason for an integrated treatment of traces is to improve the parameterisation of the model. In particular, the subcategorisation probabilities are smeared by extraction. In examples 1, 2 and 3 above 'bought' is a transitive verb, but without knowledge of traces example 2 in training data will contribute to the probability of 'bought' being an intransitive verb.

Formalisms similar to GPSG (Gazdar et al. 95) handle NP extraction by adding a *gap* feature to each non-terminal in the tree, and propagating gaps through the tree until they are finally discharged as a trace complement (see figure 5). In extraction cases the Penn treebank annotation co-indexes a TRACE with the WHNP head of the SBAR, so it is straightforward to add this information to trees in training data.

Given that the LHS of the rule has a gap, there are 3 ways that the gap can be passed down to the RHS:

Head The gap is passed to the head of the phrase, as in rule (3) in figure 5.

Left, Right The gap is passed on recursively to one of the left or right modifiers of the head, or is discharged as a *trace* argument to the left/right of the head. In rule (2) it is passed on to a right modifier, the S complement. In rule (4) a *trace* is generated to the right of the head VB.

We specify a parameter $\mathcal{P}_G(G \mid P, h, H)$ where G is either **Head**, **Left** or **Right**. The generative process is extended to choose between these cases after generating the head of the phrase. The rest of the phrase is then generated in different ways depending on how the gap is propagated: In the **Head** case the left and right modifiers are generated as normal. In the **Left, Right** cases a *gap* requirement is added to either the left or right SUBCAT variable. This requirement is fulfilled (and removed from the subcat list) when a trace or a modifier non-terminal which has the *+gap* feature is generated. For example, Rule (2), `SBAR(that)(+gap) -> WHNP(that) S-C(bought)(+gap)`, has probability

$$\mathcal{P}_h(\texttt{WHNP} \mid \texttt{SBAR,that}) \times \mathcal{P}_G(\texttt{Right} \mid \texttt{SBAR,WHNP,that}) \times$$
$$\mathcal{P}_{LC}(\{\} \mid \texttt{SBAR,WHNP,that}) \times$$
$$\mathcal{P}_{RC}(\{\texttt{S-C}\} \mid \texttt{SBAR,WHNP,that}) \times$$
$$\mathcal{P}_R(\texttt{S-C(bought)(+gap)} \mid \texttt{SBAR,WHNP,that}, \{\texttt{S-C,+gap}\}) \times$$
$$\mathcal{P}_R(\texttt{STOP} \mid \texttt{SBAR,WHNP,that}, \{\}) \times$$
$$\mathcal{P}_L(\texttt{STOP} \mid \texttt{SBAR,WHNP,that}, \{\})$$

Rule (4), `VP(bought)(+gap) -> VB(bought) TRACE NP(week)`, has probability

$$\mathcal{P}_h(\texttt{VB} \mid \texttt{VP,bought}) \times \mathcal{P}_G(\texttt{Right} \mid \texttt{VP,bought,VB}) \times$$
$$\mathcal{P}_{LC}(\{\} \mid \texttt{VP,bought,VB}) \times \mathcal{P}_{RC}(\{\texttt{NP-C}\} \mid \texttt{VP,bought,VB}) \times$$
$$\mathcal{P}_R(\texttt{TRACE} \mid \texttt{VP,bought,VB}, \{\texttt{NP-C, +gap}\}) \times$$
$$\mathcal{P}_R(\texttt{NP(week)} \mid \texttt{VP,bought,VB}, \{\}) \times$$
$$\mathcal{P}_L(\texttt{STOP} \mid \texttt{VP,bought,VB}, \{\}) \times$$
$$\mathcal{P}_R(\texttt{STOP} \mid \texttt{VP,bought,VB}, \{\})$$

In rule (2) `Right` is chosen, so the *+gap* requirement is added to RC. Generation of `S-C(bought)(+gap)`

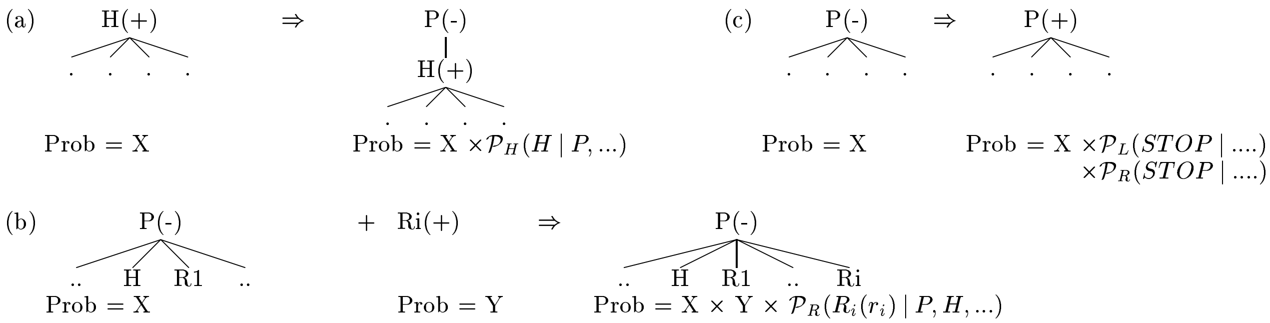


Figure 6: The life of a constituent in the chart. (+) means a constituent is complete (i.e. it includes the stop probabilities), (−) means a constituent is incomplete. (a) a new constituent is started by projecting a complete rule upwards; (b) the constituent then takes left and right modifiers (or none if it is unary). (c) finally, $STOP$ probabilities are added to complete the constituent.

Back-off Level	$\mathcal{P}_H(H \mid ...)$	$\mathcal{P}_G(G \mid ...)$ $\mathcal{P}_{LC}(LC \mid ...)$ $\mathcal{P}_{RC}(RC \mid ...)$	$\mathcal{P}_{L1}(L_i(lt_i) \mid ...)$ $\mathcal{P}_{R1}(R_i(rt_i) \mid ...)$	$\mathcal{P}_{L2}(lw_i \mid ...)$ $\mathcal{P}_{R2}(rw_i \mid ...)$
1	P, w, t	P, H, w, t	P, H, w, t, Δ, LC	L_i, lt_i, P, H, w, t, Δ, LC
2	P, t	P, H, t	P, H, t, Δ, LC	L_i, lt_i, P, H, t, Δ, LC
3	P	P, H	P, H, Δ, LC	L_i, lt_i
4	—	—	—	lt_i

Table 1: The conditioning variables for each level of back-off. For example, $\mathcal{P}_H$ estimation interpolates $e_1 = \mathcal{P}_H(H \mid P, w, t)$, $e_2 = \mathcal{P}_H(H \mid P, t)$, and $e_3 = \mathcal{P}_H(H \mid P)$. Δ is the distance measure.

fulfills both the `S-C` and $+gap$ requirements in RC. In rule (4) `Right` is chosen again. Note that generation of *trace* satisfies both the `NP-C` and $+gap$ subcat requirements.

3 Practical Issues

3.1 Smoothing and Unknown Words

Table 1 shows the various levels of back-off for each type of parameter in the model. Note that we decompose $\mathcal{P}_L(L_i(lw_i, lt_i) \mid P, H, w, t, \Delta, LC)$ (where lw_i and lt_i are the word and POS tag generated with non-terminal L_i, Δ is the distance measure) into the product $\mathcal{P}_{L1}(L_i(lt_i) \mid P, H, w, t, \Delta, LC) \times \mathcal{P}_{L2}(lw_i \mid L_i, lt_i, P, H, w, t, \Delta, LC)$, and then smooth these two probabilities separately (Jason Eisner, p.c.). In each case[7] the final estimate is

$$e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2) e_3)$$

where e_1, e_2 and e_3 are maximum likelihood estimates with the context at levels 1, 2 and 3 in the table, and λ_1, λ_2 and λ_3 are smoothing parameters where $0 \leq \lambda_i \leq 1$. All words occurring less than 5 times in training data, and words in test data which have never been seen in training, are replaced with the "UNKNOWN" token. This allows the model to robustly handle the statistics for rare or new words.

[7]Except cases L_2 and R_2, which have 4 levels, so that $e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2)(\lambda_3 e_3 + (1 - \lambda_3) e_4))$.

3.2 Part of Speech Tagging and Parsing

Part of speech tags are generated along with the words in this model. When parsing, the POS tags allowed for each word are limited to those which have been seen in training data for that word. For unknown words, the output from the tagger described in (Ratnaparkhi 96) is used as the single possible tag for that word. A CKY style dynamic programming chart parser is used to find the maximum probability tree for each sentence (see figure 6).

4 Results

The parser was trained on sections 02 - 21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al. 93) (approximately 40,000 sentences), and tested on section 23 (2,416 sentences). We use the PARSEVAL measures (Black et al. 91) to compare performance:

Labeled Precision $= \dfrac{number\ of\ correct\ constituents\ in\ proposed\ parse}{number\ of\ constituents\ in\ proposed\ parse}$

MODEL	$\leq$ 40 Words (2245 sentences)					$\leq$ 100 Words (2416 sentences)				
	LR	LP	CBs	0 CBs	$\leq$ 2 CBs	LR	LP	CBs	0 CBs	$\leq$ 2 CBs
(Magerman 95)	84.6%	84.9%	1.26	56.6%	81.4%	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 96)	85.8%	86.3%	1.14	59.9%	83.6%	85.3%	85.7%	1.32	57.2%	80.8%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%	86.8%	87.6%	1.11	63.1%	84.1%
Model 2	88.1%	88.6%	0.91	66.5%	86.9%	87.5%	88.1%	1.07	63.9%	84.6%
Model 3	88.1%	88.6%	0.91	66.4%	86.9%	87.5%	88.1%	1.07	63.9%	84.6%

Table 2: Results on Section 23 of the WSJ Treebank. **LR/LP** = labeled recall/precision. **CBs** is the average number of crossing brackets per sentence. **0 CBs,** $\leq$ **2 CBs** are the percentage of sentences with 0 or ≤ 2 crossing brackets respectively.

Labeled Recall = $\dfrac{number\ of\ correct\ constituents\ in\ proposed\ parse}{number\ of\ constituents\ in\ treebank\ parse}$

Crossing Brackets = number of constituents which violate constituent boundaries with a constituent in the treebank parse.

For a constituent to be 'correct' it must span the same set of words (ignoring punctuation, i.e. all tokens tagged as commas, colons or quotes) and have the same label[8] as a constituent in the treebank parse. Table 2 shows the results for Models 1, 2 and 3. The precision/recall of the traces found by Model 3 was 93.3%/90.1% (out of 436 cases in section 23 of the treebank), where three criteria must be met for a trace to be "correct": (1) it must be an argument to the correct head-word; (2) it must be in the correct position in relation to that head word (preceding or following); (3) it must be dominated by the correct non-terminal label. For example, in figure 5 the trace is an argument to **bought**, which it **follows**, and it is dominated by a **VP**. Of the 436 cases, 342 were string-vacuous extraction from subject position, recovered with 97.1%/98.2% precision/recall; and 94 were longer distance cases, recovered with 76%/60.6% precision/recall [9].

4.1 Comparison to previous work

Model 1 is similar in structure to (Collins 96) — the major differences being that the "score" for each bigram dependency is $\mathcal{P}_l(L_i, l_i|H, P, h, distance_l)$ rather than $\mathcal{P}_l(L_i, P, H \mid l_i, h, distance_l)$, and that there are the additional probabilities of generating the head and the $STOP$ symbols for each constituent. However, Model 1 has some advantages which may account for the improved performance. The model in (Collins 96) is deficient, that is for most sentences S, $\sum_T \mathcal{P}(T \mid S) < 1$, because probability mass is lost to dependency structures which violate the hard constraint that no links may cross. For reasons we do not have space to describe here, Model 1 has advantages in its treatment of unary rules and the distance measure. The generative model can condition on any structure that has been previously generated — we exploit this in models 2 and 3 — whereas (Collins 96) is restricted to conditioning on features of the surface string alone.

(Charniak 95) also uses a lexicalised generative model. In our notation, he decomposes $\mathcal{P}(RHS_i \mid LHS_i)$ as $\mathcal{P}(R_n...R_1HL_1..L_m \mid P, h) \times \prod_{i=1..n} \mathcal{P}(r_i \mid P, R_i, h) \times \prod_{i=1..m} \mathcal{P}(l_i \mid P, L_i, h)$. The Penn treebank annotation style leads to a very large number of context-free rules, so that directly estimating $\mathcal{P}(R_n...R_1HL_1..L_m \mid P, h)$ may lead to sparse data problems, or problems with coverage (a rule which has never been seen in training may be required for a test data sentence). The complement/adjunct distinction and traces increase the number of rules, compounding this problem.

(Eisner 96) proposes 3 dependency models, and gives results that show that a generative model similar to Model 1 performs best of the three. However, a pure dependency model omits non-terminal information, which is important. For example, "hope" is likely to generate a `VP(TO)` modifier (e.g., I hope [VP to sleep]) whereas "require" is likely to generate an `S(TO)` modifier (e.g., I require [S Jim to sleep]), but omitting non-terminals conflates these two cases, giving high probability to incorrect structures such as "I hope [Jim to sleep]" or "I require [to sleep]". (Alshawi 96) extends a generative dependency model to include an additional state variable which is equivalent to having non-terminals — his

[8] (Magerman 95) collapses `ADVP` and `PRT` to the same label, for comparison we also removed this distinction when calculating scores.

[9] We exclude infinitival relative clauses from these figures, for example "I called a plumber TRACE to fix the sink" where 'plumber' is co-indexed with the trace subject of the infinitival. The algorithm scored 41%/18% precision/recall on the 60 cases in section 23 — but infinitival relatives are extremely difficult even for human annotators to distinguish from purpose clauses (in this case, the infinitival could be a purpose clause modifying 'called') (Ann Taylor, p.c.)

suggestions may be close to our models 1 and 2, but he does not fully specify the details of his model, and doesn't give results for parsing accuracy. (Miller et al. 96) describe a model where the RHS of a rule is generated by a Markov process, although the process is not head-centered. They increase the set of non-terminals by adding semantic labels rather than by adding lexical head-words.

(Magerman 95; Jelinek et al. 94) describe a history-based approach which uses decision trees to estimate $\mathcal{P}(T|S)$. Our models use much less sophisticated n-gram estimation methods, and might well benefit from methods such as decision-tree estimation which could condition on richer history than just surface distance.

There has recently been interest in using dependency-based parsing models in speech recognition, for example (Stolcke 96). It is interesting to note that Models 1, 2 or 3 could be used as language models. The probability for any sentence can be estimated as $\mathcal{P}(S) = \sum_T \mathcal{P}(T, S)$, or (making a Viterbi approximation for efficiency reasons) as $\mathcal{P}(S) \approx \mathcal{P}(T_{best}, S)$. We intend to perform experiments to compare the perplexity of the various models, and a structurally similar 'pure' PCFG[10].

5 Conclusions

This paper has proposed a generative, lexicalised, probabilistic parsing model. We have shown that linguistically fundamental ideas, namely subcategorisation and wh-movement, can be given a statistical interpretation. This improves parsing performance, and, more importantly, adds useful information to the parser's output.

6 Acknowledgements

I would like to thank Mitch Marcus, Jason Eisner, Dan Melamed and Adwait Ratnaparkhi for many useful discussions, and comments on earlier versions of this paper. This work has also benefited greatly from suggestions and advice from Scott Miller.

References

H. Alshawi. 1996. Head Automata and Bilingual Tiling: Translation with Minimal Representations. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 167-176.

E. Black et al. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

T. L. Booth and R. A. Thompson. 1973. *Applying Probability Measures to Abstract Languages.* IEEE Transactions on Computers, C-22(5), pages 442-450.

E. Charniak. 1995. *Parsing with Context-Free Grammars and Word Statistics.* Technical Report CS-95-28, Dept. of Computer Science, Brown University.

N. Chomsky. 1957. *Syntactic Structures*, Mouton, The Hague.

M. J. Collins. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184-191.

J. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. *Proceedings of COLING-96*, pages 340-345.

G. Gazdar, E.H. Klein, G.K. Pullum, I.A. Sag. 1985. *Generalized Phrase Structure Grammar.* Harvard University Press.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, S. Roukos. 1994. Decision Tree Parsing using a Hidden Derivation Model. *Proceedings of the 1994 Human Language Technology Workshop*, pages 272-277.

D. Magerman. 1995. Statistical Decision-Tree Models for Parsing. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276-283.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313-330.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. *Proceedings of the 1994 Human Language Technology Workshop*, pages 110-115.

S. Miller, D. Stallard and R. Schwartz. 1996. A Fully Statistical Approach to Natural Language Interfaces. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 55-61.

A. Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. *Conference on Empirical Methods in Natural Language Processing.*

A. Stolcke. 1996. Linguistic Dependency Modeling. *Proceedings of ICSLP 96, Fourth International Conference on Spoken Language Processing.*

[10]Thanks to one of the anonymous reviewers for suggesting these experiments.