

Abstract of “Unsupervised Bayesian Lexicalized Dependency Grammar Induction” by William Perry Headden III, Ph.D., Brown University, May 2012.

This dissertation investigates learning dependency grammars for statistical natural language parsing from corpora without parse tree annotations. Most successful work in unsupervised dependency grammar induction has assumed that the input consists of sequences of parts-of-speech, ignoring words and using extremely simple probabilistic models. However, supervised parsing has long shown the value of more sophisticated models which use lexical features. These more sophisticated models however require probability distributions with complex conditioning information, which must be smoothed to avoid sparsity issues.

In this work we explore several dependency grammars that use smoothing, and lexical features. We explore a variety of different smoothing regimens, and find that smoothing is helpful for even unlexicalized models such as the Dependency Model with Valence. Furthermore, adding lexical features yields the highest accuracy dependency induction on the Penn Treebank WJS10 corpus to date. In sum, this dissertation extends unsupervised grammar induction by incorporating lexical conditional information, by investigating smoothing in an unsupervised framework.

Unsupervised Bayesian Lexicalized Dependency Grammar Induction

by

William Perry Headden III

B. S., Georgetown University, 2003

Sc. M., Brown University, 2006

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
the Department of Computer Science at Brown University.

Providence, Rhode Island

May 2012

© Copyright 2012 by William Perry Headden III

This dissertation by William Perry Headden III is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____

Mark Johnson, Adviser

Recommended to the Graduate Council

Date _____

Eugene Charniak, Reader

Date _____

Jason Eisner, Reader

Date _____

Stuart Geman, Reader

Approved by the Graduate Council

Date _____

Peter Weber

Dean of the Graduate School

Acknowledgements

In memory of my mother.

I would like to express my appreciation to everyone who contributed their support during my time at Brown which culminated in this dissertation.

Let begin with my advisors: Mark Johnson, who kept me focused and whose enthusiasm and insight are much appreciated; and Eugene Charniak, whose door was always open for valuable discussion and advice. Their patience and expertise were invaluable, and I learned a great deal from each of them. Thanks also to the rest of my committee, Jason Eisner and Stuart Geman, for their helpful comments and insights.

I am indebted also to many other professors, especially Tom Griffiths for introducing me to the virtues of Bayesian Statistics, as well as Mark Maloof and Sharon Caraballo, for sparking my interest in machine learning and natural language processing while I was an undergraduate at Georgetown.

Thanks also to members of the Brown Laboratory for Linguistic Information Processing, especially David McClosky, Jenine Turner-Trauring, Matthew Lease, Micha Elsner, Sharon Goldwater, Justin Palmer, Stu Black, Engin Ural, Bevan Jones, and Stella Frank; other friends and fellow students at while I was at Brown: Melissa Chase, Gregory Cooper,

Mira Belenkiy, Naomi Feldman, Dan Grollman, Serdar Kadioglu, Tomer Moscovich, Olga Karpenko, Maggie Benthall; Matthew Krause for making me glad my research did not require zookeeping duties; as well as many others.

Finally I would like to thank my family: my parents Jeff and Ginny, and my brother Andrew, whose love, support, and occasional badgering were essential to completing this dissertation.

Contents

List of Tables	ix
List of Figures	xiv
1 Introduction	1
1.1 Syntactic Dependencies	3
1.2 A Short History of Unsupervised Dependency Grammar Induction	3
1.2.1 Dependency Model with Valence	4
1.2.2 Other Estimation techniques for DMV	7
1.2.3 Unsupervised Search	9
1.2.4 IBM-style alignment models	9
1.2.5 Variational Bayesian Techniques	9
1.2.6 Extensions of Valence	12
1.3 Overview of remainder of thesis	14
2 Learning Framework	15
2.1 Background	15
2.1.1 Split Bilexical CFGs	16
2.1.2 Dependency Model with Valence	20
2.1.3 Extended Valence Grammar	22

2.2	Experimental Setup	23
2.3	Tied Probabilistic Context Free Grammars	25
2.4	Estimation	28
2.4.1	Initialization and Search	31
2.4.2	Analysis: Harmonic vs. Randomized Initialization	35
2.5	Conclusion	36
3	Smoothing PCFGs	42
3.1	Introduction	42
3.2	Estimation from Fully Observed Events	43
3.3	Unsupervised Smoothing for PCFGs	45
3.4	Estimation from Held-out data	48
3.5	Bayesian Estimation with Linear Interpolation Smoothing	51
3.5.1	Priors Preferring Backoff distribution	53
3.6	Analysis	59
3.7	Conclusions	69
4	Lexicalization	72
4.1	Introduction	72
4.2	Lexicalized Grammar Framework	73
4.3	Experimental Setup	74
4.4	Models	76
4.4.1	Argument Distribution: Model 1	76
4.4.2	Stop Distribution	79
4.4.3	Generating the Dependent word.	81
4.5	Discussion	83
4.5.1	Lexicalizing DMV	85

4.6	Analysis: Lexicalized 1b-100 vs EVG	87
4.7	Conclusions	93
5	Final Matters	95
5.0.1	Other Languages	97
5.1	Another look at Randomized Initialization	97
5.2	Learning from Induced tags	102
5.2.1	Learning Hidden State Models	104
5.3	Conclusions	108
A	Summary of Key Notation	110
B	Split Bilexical PCFGs of Dependency Models	112
C	Calculating Confidence Bounds using Bootstrap Sampling	116
D	Smoothing with Collapsed Interpolation	118
D.0.1	DMV and EVG with Collapsed Interpolation	120
	Bibliography	122

List of Tables

2.1	CFG Schema for Dependency Model with Valence.	21
2.2	CFG Schema for the Extended Valence Grammar.	24
2.3	Results of Randomized vs Harmonic initialization for Variational Bayes . . .	35
2.4	Change in correct dependencies by child part-of-speech across 10 runs for DMV-Randomized and DMV-Harmonic. Left indicates child is a left dependent of its head in system output.	37
2.5	Change in correct dependencies by head part-of-speech across 10 runs for DMV-Randomized and DMV-Harmonic. Left indicates child is a left dependent of its head in system output.	38
2.6	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for DMV-Randomized less DMV-Harmonic across 10 runs. Bold: $\text{abs}(\text{difference}) > 1000$. Italics: $\text{abs}(\text{difference}) > 100$	39
2.7	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for DMV-Randomized less DMV-Harmonic across 10 runs. Bold: $\text{abs}(\text{difference}) > 1000$. Italics: $\text{abs}(\text{difference}) > 100$	40
3.1	Results smoothed DMV using held-out estimation for Linear Interpolation using EM. All results use randomized initialization	50

3.2	Results for DMV, EVG smoothed with Linear Interpolation trained using Variational Bayes.	53
3.3	DMV and EVG smoothed using Linear Interpolation, varying strength of interpolation hyperparameters β_1, β_2	54
3.4	DMV and EVG smoothed using Linear Interpolation, varying relative strengths of β_1, β_2	55
3.5	Smoothed DMV with prefer-backoff prior 1 on λ_s	57
3.6	Smoothed DMV and EVG with prefer-backoff prior 2 on λ_s	58
3.7	Change in correct dependencies by child part-of-speech across 10 runs for DMV with and without Linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.	60
3.8	Change in correct dependencies by head part-of-speech across 10 runs for DMV with and without Linear Interpolation smoothing. Left indicates child is a left dependent of its head in system output.	61
3.9	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for DMV with Linear Interpolation less DMV across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.	62
3.10	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for DMV with Linear Interpolation less DMV across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.	63
3.11	Change in correct dependencies by child part-of-speech across 10 runs for EVG and DMV with linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.	64
3.12	Change in correct dependencies by head part-of-speech across 10 runs for EVG and DMV with linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.	65

3.13	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for EVG less DMV (with linear interpolation) across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.	66
3.14	Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for EVG less DMV (with linear interpolation) across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.	67
4.1	Schema for Lexicalized EVG	75
4.2	Lexicalized Model 1a backoff chains. The dependent part-of-speech distribution is lexicalized. The λ s associated with interpolating between $P(A Hhvd)$ and $P(A Hvd)$ are conditioned on the events $hHvd$	78
4.3	Lexicalized Model 1b backoff chains. The dependent part-of-speech distribution is lexicalized. It differs from Lexicalized model 1a in that the λ s associated with interpolating between $P(A Hhvd)$ and $P(A Hvd)$ are conditioned on the events Hvd rather than $hHvd$	78
4.4	Results of Lexicalized models 1 and 1b	79
4.5	Lexicalized model 2 backoff chains. The stop distribution is lexicalized, while the dependent part-of-speech distribution is not. Equivalent to smoothed EVG, with a lexicalized stop distribution.	80
4.6	Lexicalized model 3 backoff chains. Both the stop and dependent part-of-speech distributions are lexicalized. Equivalent to Lexicalized Model 1b with the addition of a lexicalized stop distribution.	80
4.7	Results of Lexicalized models 2 and 3.	80
4.8	Lexicalized model 4 backoff chains	82
4.9	Lexicalized model 5 backoff chains	82
4.10	Results of Lexicalized models 4 and 5	82

4.11	Effect of training linear interpolated EVG with different amounts of training data. We can see that adding increase amounts of NYT degrades accuracy on WSJ.	83
4.12	Effect of training linear interpolated EVG when training from automatically tagged WSJ and NYT Gigaword corpora, controlling for corpus size. Note that since we do not have gold standard trees for Gigaword we cannot present parsing results on the training set.	84
4.13	Effect of training Lexicalized models 1b and 3 on WSJ10, setting the unknown word cutoff to 100 times.	85
4.14	Lexicalized DMV backoff chains. The dependent part-of-speech distribution is lexicalized, and smoothed with the basic DMV dependent part-of-speech distribution.	86
4.15	Effect of training Lexicalized DMV on WSJ10, setting the unknown word cutoff to 100 times. We compare it to DMV, to show that adding just lexicalization to DMV improves performance even without the gains from Chapter 3.	86
4.16	Change in correct dependencies by child part-of-speech across 10 runs. Left indicates child is a left dependent of its head in system output.	88
4.17	Change in correct dependencies by head part-of-speech across 10 runs. Left indicates child is a left dependent of its head in system output.	89
4.18	Correct: Child POS(columns) vs Head POS(rows)(Part 1)	90
4.19	Correct: Child POS(columns) vs Head POS(rows)(Part 2)	91
5.1	Results on WSJ 10 test set (section 23). * Prior work.	96
5.2	Results on WSJ 20 development set (section 24), i.e.	96
5.3	Training set directed accuracy across several languages.	98

5.4	Training set Undirected Accuracy across languages.	99
5.5	DMV on tags induced using bitag HMM, accuracy on the training set. . . .	103
5.6	DMV with randomized Harmonic Initializer	105
5.7	DMV with annotated parts-of-speech and smoothing backoff chains. The annotated dependent part-of-speech distribution is conditioned on the annotated head part-of-speech, and is smoothed with a distribution conditioned on the unannotated head. The dependent word is conditioned on the annotated part-of-speech.	107
5.8	DMV with hidden states on HMM-induced part-of-speech tags, WSJ10 sections 2-21.	108
B.1	Basic Schema for EVG	113
B.2	Schema for EVG after fold-unfold transform	114
D.1	Results from Smoothing DMV, EVG using Collapsed Interpolation	121

List of Figures

1.1	A Dependency Tree	1
1.2	Dependency Model with Valence Generative Story - Example	5
2.1	Example dependency parse.	16
2.2	Simple split-head bilexical CFG parse of “The big dog barks.”	18
2.3	An example of moving from Simple Split-Head Grammar 1 to Simple Split-Head Grammar 2, which separates the decision of whether to generate another argument (L_H) from what that argument should be (L_H^1).	20
2.4	An example of moving from Simple Split-Head Grammar 2 to DMV. DMV distinguishes between the deciding if there will be a first argument, and deciding when to stop generating subsequent arguments. Notice the lower two L_{dog} nonterminals in the left tree becomes L'_{dog} in the second.	20
2.5	An example of moving from DMV to EVG. The key difference is that EVG distinguishes between the distributions over the argument nearest the head (<i>big</i>) from arguments farther away (<i>The</i>).	22
2.6	Directed Accuracy vs Log Likelihood, 100 runs of Expectation Maximization trained DMV.	32
2.7	Directed Accuracy vs Lower Bound, 100 runs of Variational Bayes trained DMV, each with 20 random restarts.	34

2.8	Undirected Accuracy vs Lower Bound, 100 runs of Variational Bayes trained DMV, each with 20 random restarts.	35
3.1	Using linear interpolation to smooth $L_{dog}^1 \rightarrow Y_{big} dog_L$: The first component represents the distribution fully conditioned on head <i>dog</i> , while the second component represents the distribution ignoring the head conditioning event. This later is accomplished by tying the rule $L_{dog}^{1b2} \rightarrow Y_{big} dog_L$ to, for instance, $L_{cat}^{1b2} \rightarrow Y_{big} cat_L$, $L_{fish}^{1b2} \rightarrow Y_{big} fish_L$ etc.	49
3.2	Plot of $p(\lambda_1 \beta)$ vs. λ_1 for various β hyperparameters.	56
3.3	Beta Prior favoring Backoff distribution, version 2	58
3.4	Directed Accuracy vs Lower Bound, 300 runs of DMV/EVG with linear interpolation. Run initialized from gold standard dependencies marked with x.	70
3.5	Undirected Accuracy vs Lower Bound, 300 runs of DMV/EVG with linear interpolation. Run initialized from gold standard dependencies marked with x.	71
4.1	A subtree from an unlexicalized EVG parse, and the corresponding subtree in lexicalized EVG, which adds word annotations.	74
4.2	Dependency tree for sentence with head “says.” Notice that the right argument of “says” is a VBZ, which would be unlikely as the right argument of the VBZ “gives”.	77
4.3	Directed Accuracy vs Lower Bound, 300 runs of Lexicalized Model1b-100.	92
4.4	Undirected Accuracy vs Lower Bound, 300 runs of Lexicalized Model1b-100.	93

5.1 % of runs with Directed Accuracy exceeding the threshold given on the x -axis. Each run is the result of has a beam of 20 random-restarts of Variational Bayes, and running the best of these to convergence. Each lexicalized model run however has a beam of 20 smoothed EVG models, from which one is initialized and run to convergence. 100

5.2 % of runs with Undirected Accuracy exceeding the threshold given on the x -axis, using the same setup as Figure 5.1. 101

D.1 Using collapsed interpolation to smooth $L_{dog}^1 \rightarrow Y_{big} L_{dog}^0$: note that from L_{dog}^1 there are two paths to the children $Y_{big} L_{dog}^0$: (1) directly and (2) indirectly through L_{dog}^{1b} . The distribution over different arguments given L_{dog}^{1b} is tied 120

Chapter 1

Introduction

The last decade has seen great strides in statistical natural language parsing. Supervised and semi-supervised methods provide highly accurate parsers, but require training from corpora hand-annotated with parse trees. Unfortunately, manually annotating corpora with parse trees is expensive, so for languages and domains with minimal resources it is valuable to find methods to parse without requiring annotated sentences. The topic of this dissertation is unsupervised methods of learning syntactic models. In particular, our focus will be on unsupervised dependency parsing. Informally, a dependency parse tree is a directed graph structure whose nodes are words in a sentence and whose edges point from some word to a syntactic dependent of that word. We can see an example in Figure 1.1. In this instance, “dog” and “home” are dependents of the main verb “ran” and “big” modifies “dog”. The edge from the word “root” to “ran” indicates that “ran” is the root of the dependency tree.

Most supervised and unsupervised systems use parts-of-speech such as nouns, verbs,



Figure 1.1: A Dependency Tree

and adjectives as a coarse proxy of the words in the sentence in determining their syntactic relations. This is useful both for generalization, as well as when an individual word has only been seen infrequently. However, the words themselves are quite useful in determining whether one is a dependency of the other or not. For instance, “pancake” is a far more likely dependent for the verb “eat” than “juice” would be, even though both are nouns. Supervised parsers have shown time and time again that word-to-word relations are extremely valuable (see for instance (Charniak, 2000; Collins, 1999; McDonald et al., 2005)). The use of these features is called *lexicalization*.

Unfortunately, to date unsupervised dependency parsing has not successfully incorporated lexical information. Most of the recent work has focused on variants of the Dependency Model with Valence (DMV) of Klein and Manning (2004), which we describe below in Section 2.1.2. DMV treats the parts-of-speech of a sentence as the observed variables and learns the relations between pairs of parts-of-speech.

The main goals of this dissertation are twofold. The first is to investigate smoothing in unsupervised dependency grammar induction, which will be required in order to model more sophisticated conditioning events. We propose several frameworks for doing so, and show how they can be estimated. We also present a model that successfully uses one of the frameworks. The second goal is to incorporate lexical features, which have been so successful in supervised parsing. We present several models that use lexical features.

This dissertation presents and expands upon the models we described in our 2009 NAACL paper (III et al., 2009).

In the remainder of this chapter we will first introduce syntactic dependencies and dependency grammars. We then discuss the state of the literature regarding unsupervised dependency induction. Finally we will end with an overview of the rest of the thesis.

1.1 Syntactic Dependencies

In this thesis, the observed variables will generally be a corpus of n sentences of text s , where each s_i is a sequence of m_i words $s_{i1} \dots s_{im_i}$. Each word s_{ij} is associated with a part-of-speech τ_{ij} , which for most of the thesis will also be treated as observed.

The hidden variables will be dependency parse trees $t = \{t_i\}_{i=1,n}$. A dependency tree t_i is a directed acyclic graph whose nodes are the words in s_i . The graph has a single incoming edge for each word s_{ij} , except one word, denoted as the *root* of the graph. An edge from word s_{ij} to $s_{ij'}$ means that $s_{ij'}$ is a *dependent* of s_{ij} , or alternatively, s_{ij} is the *head* of $s_{ij'}$. Note that each word token may be the dependent of only one head, but a head may have several arguments.

For instance, in Figure 1.1, “barks” is the root of the graph, “dog” is a dependent of barks, and “The” and “big” are both dependents of “dog”. In turn, “dog” is the head of “The” and “big,” and “barks” is the head of “dog.”

1.2 A Short History of Unsupervised Dependency Grammar Induction

Carroll and Charniak (1992) investigate a dependency grammar which generates all of the argument parts-of-speech of a given head part-of-speech, using EM and a probabilistic context free grammar. They report poor results, but note importantly that local maxima in the likelihood function are a major problem in this space: each of their randomly initialized grammars ended up in a different maxima.

Paskin (2001) presents a dependency model trained from words in which, given sentence length l , first a dependency tree t is generated uniformly of those with l edges. Then, starting with the root, each edge is assigned an argument, given its head word (or ROOT)

and the direction of the edge. The probability of the parsed sentence s, t , given the sentence length l is:

$$P(s, t|l) = P(t|l)P(s|t) = P(t|l)P_{ROOT}(s_r) \prod_{i \rightarrow j \in t} P(s_j|s_i, d)$$

This is trained using EM on 67 million words of untagged newswire text and evaluated on the Penn Treebank Wall Street Journal. He reports 39.1 percent undirected accuracy on unseen WSJ test data, which is below the baseline of attaching each word to the word next to it.


Yuret (1998) describes a similar model, which is trained through a heuristic bootstrapping method in which each sentence is parsed using the statistics of the previous sentences, and then this single parse is used to update the word pair statistics.

1.2.1 Dependency Model with Valence

Klein and Manning (2004) introduced the Dependency Model with Valence, which due to its success has been the focus of most of the dependency grammar induction work ever since. Whereas in Paskin's model the dependency tree structures are considered equally likely, DMV conditions the number of arguments a particular head has in a particular direction on both the head, and whether or not this head has generated any arguments in this direction previously. Additionally, DMV uses the parts-of-speech, rather than the words, to determine these relations.

The DMV is a generative model of dependencies, in which the dependency trees are generated according to the following generative process:

- Generate Root POS
- For each generated POS H , direction $d \in \{L, R\}$ generate H 's arguments A_{dH} :



The big dog barks

- Generate root "barks"
- Decide to generate 1 left argument to "barks"
- Decide to generate 0 right arguments to "barks"
- Generate "dog" as left argument of "barks"
- Decide to generate 2 left arguments to "dog"
- Decide to generate 0 right arguments to "dog"
- Generate "bad" as left argument to "dog"
- Generate "the" as left argument to "dog"
- Decide to generate 0 left arguments to "bad"
- Decide to generate 0 right arguments to "bad"
- Decide to generate 0 left arguments to "the"
- Decide to generate 0 right arguments to "the"

Figure 1.2: Dependency Model with Valence Generative Story - Example

- Decide whether to generate any arguments in direction d .
- Decide the number of arguments k_{Hd} greater than zero in direction d by repeatedly deciding whether to continue generating arguments in this direction, given H and the fact that there is at least one argument.
- Generate k_{Hd} parts-of-speech arguments $A_1, \dots, A_{k_{Hd}}$ given d, H .

Let t_H be the subtree rooted at H , and let A_{dH} be a vector of dependencies of H in direction d . Let A_{dHi} be the i th element of A_{dH} . A_{dH0} is considered the nearest argument to H . This yields the probability of a dependency tree as

$$P(s, t) = P_{ROOT}(s_r)P(t_{s_r}|s_r)$$

where

$$P(t_H|H) = \prod_{d \in \{L, R\}} \prod_{v=0}^{k_{hd}-1} [P_{stop}(\text{continue}|H, d, \min(v, 1)) P_{arg}(A_{dHi}|H, d)P(t_{A_{dHi}}|A_i)] P_{stop}(\text{stop}|H, d, \min(k_{hd}, 1))$$

Klein and Manning formulate this as a PCFG, supposing that right attachments are made before left. This is then trained using the inside-outside algorithm (Baker, 1979), a variant of the Expectation Maximization algorithm for PCFGs. This yields a model with 43.2 % directed accuracy on the Penn Treebank Wall Street Journal corpus, stripped of punctuation, sentences with 10 words or fewer (henceforth WSJ10), training and testing on the whole corpus.

Key to their performance is their initialization: they start with an E-step in which the posterior probability that the j th word is an argument of the i th is proportional to $\frac{1}{|i-j|}$,

and each word is equally likely to be the root. By combining DMV with their Constituent-Context Model for constituent bracketing induction, they can improve this performance to 47.5%.

1.2.2 Other Estimation techniques for DMV

Smith and Eisner (2005) and Smith (2006) investigate using their Contrastive Estimation technique to estimate DMV. Contrastive Estimation maximizes the conditional probability of the observed sentences s given some similar sequences that were not seen. These similar sequences are defined by a Neighborhood function, which defines for each sentence s_i a set $\mathcal{N}(s_i)$, consisting of all such sequences that can be made from some simple transformation of s_i . Examples of such transformations include transposing a pair of words, or deleting a word. The intuition is that for every observed s_i , there are many sequences involving words with the same semantic content, but which were not generated. These are treated as “implicit negative examples”, and Contrastive Estimation places probability mass on s_i at the expense of them. The results of this approach vary widely based on regularization and neighborhood, but many of the solutions found are quite a bit better than when using EM.

Smith (2006) also investigates two techniques for maximizing likelihood, while incorporating the locality bias encoded in the Klein and Manning initializer for DMV. One, skewed deterministic annealing, ameliorates the local maximum problem by raising to some positive power $b \leq 1$ the posterior distribution over trees in the E-step of EM, flattening it, while interpolating in log space with the Klein and Manning initializer. The degree to which the latter distribution is included is decreased with time, eventually a local maximum in the likelihood.

The second technique is structural annealing, introduced in Smith and Eisner (2006) and explored more deeply by Smith (2006), which penalizes long dependencies at first.

This penalty is gradually weakened during estimation. This is inspired by a notion that short dependencies are easier to learn than long dependencies. If hand-annotated dependencies on a held-out set are available for parameter selection, this performs far better than EM; however performing parameter selection on a held-out set without the use of gold dependencies does not perform as well.

The Baby Steps technique of Spitzkovsky et al. (2009; Spitzkovsky et al. (2010a) exploits a similar idea that short sentences are easier to learn from than long sentences. This technique starts by training DMV using EM on sentences of length one, then incrementally including sentences of length one longer in the training data, initializing with the previous model.

Spitzkovsky et al. (Spitzkovsky et al., 2009; Spitzkovsky et al., 2010a) note a tradeoff between the addition of more training data by including longer sentences and the resulting complexity of the sentences in question. They report that training on sentences up to length 15 using an initializer similar to that of Klein and Manning (2004) works better when evaluated on test sets with varying sentence lengths. They additionally combine this idea with the incremental aspect of Baby Steps to form what they call Leapfrog, (Spitzkovsky et al., 2010a) which initializes the model using a mixture of the outputs of this model and Baby Steps on sentences of length 1 to 15. It then trains EM to convergence, and then repeat this process on training sets with length cutoffs of 30 and 45, each time initializing with the result of the previous model.

Spitzkovsky et al. (2010b) investigate using Viterbi EM instead of traditional EM to estimate DMV. Viterbi EM replaces the expected counts of the E-step with counts extracted from the Viterbi parse of the current model. They find that this works better than traditional EM when training on longer sentences.

Berg-Kirkpatrick et al. (2010) present a method for using EM with locally normalized log-linear models and extend this by showing how the gradient optimization in the M-step

can be changed to directly climb the gradient of the log marginal likelihood. They use this to experiment with adding broader part-of-speech knowledge to the model, such as adding features that something is a noun or a verb.

1.2.3 Unsupervised Searn

Daumé III (2009) applies an unsupervised version of his Searn structured prediction approach to unsupervised dependency parsing. Searn is a framework for applying classifiers to structured prediction problems, given some loss function. Unsupervised Searn works by considering loss functions which only depend on the observed data. Daumé applies this to a shift-reduce parser, which includes both lexical and unlexical features.

1.2.4 IBM-style alignment models

Brody (2010) explores using the IBM word alignment models, used for statistical machine translation, and reformulating them for dependency grammar induction. The approach considers an alignment between two copies of the same sentence, where words are prevented from aligning to themselves. An alignment is considered the analogue to a dependency in this model. The approach does not perform as well as the DMV, though this is possibly due to the fact that the model does not force the alignment to have a tree-structure.

1.2.5 Variational Bayesian Techniques

Cohen et al. (2008) explores using two Bayesian priors in conjunction with the Dependency Model with Valence: a sparse symmetric Dirichlet Prior and a Logistic Normal Prior. The sparse Dirichlet places a bias towards distributions where each nonterminal expands to only a few of its possible right-hand-sides (Johnson et al., 2007).

The Logistic Normal Prior allows for prior distributions in which the probabilities of right-hand sides of a rule can covary. Under this prior, the probability vector for a particular nonterminal with K possible righthand-sides is generated by first drawing from a K -dimensional multivariate Gaussian, exponentiating the result and normalizing to form a legitimate probability distribution. The result is a distribution where more of the covariance between rules of a particular nonterminal is captured in the prior. Cohen et al. (2008) use an Empirical Bayes approach to learn this model, which gives the best previous performance on this task, 59.1% directed accuracy on WSJ10. Additionally, if knowledge of the meanings of the part-of-speech tags is allowed, they get further improvement (59.4%) by initializing the covariance matrices so that tags in the same family (all nouns in one family, all verbs in another, etc) covary positively.

Concurrently to this work, Cohen and Smith (2009) explore an extension of the Logistic Normal Prior called the Shared Logistic Normal Prior. The Shared Logistic Normal Prior replaces the K -dimensional Multivariate Gaussian random variable with the average over several multivariate Gaussian random variables (called experts). Each expert may be shared across different nonterminals, so that rule probabilities can correlate. Cohen et al. use this model to explore tying tags in the same family together: nouns, verbs and adjectives. They find that this improves performance beyond the simple Logistic Normal Prior for English, and sometimes helps, and sometimes hurts for Chinese. They also explore a bilingual learning setting, where part-of-speech families are given a common expert across an English and Chinese, giving further improvement.

Gillenwater et al. (2011) apply the Posterior Regularization (PR) framework of Graça et al. (2008) to the problem of dependency grammar induction. Posterior Regularization allows the inclusion of soft constraints on the learned posterior distribution which might be difficult to encode in a prior. Gillenwater et al. (2011) explore encouraging sparsity on the total number of types of parent/child relations (by part-of-speech) in the grammar.

This differs from the form of sparsity coming from the sparse Dirichlet Prior, which only encourages each individual probability distribution in the model to be sparse (e.g. for DMV encouraging each parent/direction type to have a few types of children).

Naseem et al. (2010) use universal linguistic rules as a soft constraint using posterior regularization, on top of a Hierarchical Dirichlet Process-grammar refinement version of DMV. Their model also includes an extended notion of valence, conditioning the dependent on whether it is the first, second, or third-or-greater child of its parent. (see Subsection 1.2.6). They find that including these rules can improve performance over even the models presented in this paper; however, when the rule constraints are excluded, the performance drops precipitously.

Cohen et al. (2010) apply the Adaptor Grammar framework (Johnson et al., 2006) to DMV. Adaptor grammars are a nonparametric Bayesian extension to PCFGs, which allow for whole previously generated subtrees to be memoized and reused without repaying the probabilistic cost of generating it anew. Cohen et al. apply this memoization to noun constituents with the DMV, utilizing a novel variational estimation technique. They find this brings a modest but significant improvement over the baseline DMV with Dirichlet priors.

Blunsom and Cohn (2010) present a Tree Substitution Grammar (TSG) version of lexicalized DMV (see Section 4.5.1). A TSG represents parse trees as made up of a combination of tree fragments that might be larger than those of a simple CFG parse. Their model uses hierarchical Pitman-Yor processes to do the smoothing. While the model doesn't explicitly treat arguments differently by their valence position (See Section 1.2.6), by modeling larger tree fragments it is able to incorporate that information.

1.2.6 Extensions of Valence

Valence has long been known to be valuable for supervised parsing (Eisner, 1996; Eisner, 2000; Collins, 1997; Charniak, 2000). McClosky (2008) presents two variations of DMV which extend its notion of valence. McClosky notes that across languages, few heads have three or more arguments in a given direction. *Restricted Valence Grammar* (RVG) presumes that each head has a maximal number of positions K in each direction to fill (McClosky considers $K \in \{1, 2, 3\}$). Its generative process is:

- Generate Root POS
- For each generated POS H , direction $d \in \{L, R\}$ generate H 's arguments $D_d(H)$:
 - Decide how many arguments $k_{Hd} \in \{0, 1, \dots, K\}$ in direction d to generate.
 - Generate k_{Hd} parts-of-speech arguments $A_1, \dots, A_{k_{Hd}}$ given d, H, v , where v is the valence slot.

The probability of a dependency tree under RVG is then:

$$P(s, t) = P_{ROOT}(s_r)P(t_{s_r}|s_r)$$

where

$$P(t_H|H) = \prod_{d \in \{L, R\}} P(k_{Hd}|H, d) \prod_{i=0}^{k_{dH}-1} [P_{arg}(A_{dHi}|H, d, i)P(t_{A_{dHi}}|A_i)]$$

In contrast *Unrestricted Valence Grammar* (UVG), which we use as the basis for our models, does not place a hard limit on the number of arguments; but merely models arguments K or more positions away from the head according to the same distribution. It can

be thought of as the same as DMV, except that argument A_{Hdi} is modeled conditioned on $H, d, \min(i, K)$, not merely on H, d as in DMV.

- Decide whether to generate any arguments in direction d .
- Decide the number of arguments k_{Hd} greater than zero in direction d by repeatedly deciding whether to continue generating arguments in this direction, given H and the fact that there is at least one argument.
- Generate k_{Hd} parts-of-speech arguments $A_1, \dots, A_{k_{Hd}}$ given d, H, v , where v is the valence slot. If $v \geq K$ the arguments are drawn from a common distribution.

Under UVG the probability of a dependency tree as

$$P(s, t) = P_{ROOT}(s_r)P(t_{s_r}|s_r)$$

where

$$P(t_H|H) = \prod_{d \in \{L, R\}} \prod_{v=0}^{k_{Hd}-1} [P_{stop}(\text{continue}|H, d, \min(v, K)) P_{arg}(A_{dHi}|H, d) P(t_{A_{dHi}}|A_v)] P_{stop}(\text{stop}|H, d, \min(k_{Hd}, K))$$

In the remainder of the thesis, we will refer to this model as *Extended Valence Grammar* (EVG), which is what we called it in our 2009 NAACL paper (III et al., 2009).

McClosky reports the RVG with valence 2 gets the best performance of these models on English Wall Street Journal words length 10 or less (56.5 directed, 69.7 undirected). Although this is the case, our models are extensions on the EVG framework with $K = 2$, since we are reluctant to eliminate potentially useful attachments a priori. Pilot Experiments with RVG and EVG with backoff indicated they work roughly equally well.

1.3 Overview of remainder of thesis

In Chapter 2 we will go over the basic learning framework we will use in the remainder of the thesis. This will include representing dependency models as split bilexical PCFGs, define PCFGs with parameters tied in a particular way, as well as the unsupervised estimation of these PCFGs using various techniques. It will additionally go into the details of the Dependency Model with Valence and Extended Valence Grammar which will form the basis of the models discussed in the remainder of the dissertation.

Chapter 3 will cover a series of different smoothing techniques for PCFGs which can be estimated in an unsupervised fashion. These will involve augmenting the PCFG in particular ways, and making use of the tied-PCFG framework. It will look at smoothed versions of DMV and EVG, and see how smoothing can improve the performance of these models.

Chapter 4 will explore lexicalized models. We will look at integrating lexical features to the DMV and EVG models in various ways. We will utilize the lessons regarding smoothing learned in Chapter 3 to effectively learn these models.

Chapter 5 will look at several related questions that arose in the course of this work, and conclude.

Chapter 2

Learning Framework

2.1 Background

This chapter will describe the general learning framework we shall employ for the rest of the thesis. We shall begin by describing the sort of syntactic structure we are interested in learning. Next we shall discuss the split-head bilexical context-free grammar framework for describing dependency grammars, and describe the models of previous work as probabilistic context free grammars of this variety. We will next discuss various unsupervised estimation procedures for PCFGs. Finally we shall close by describing a variety of PCFG that will be useful in the next Chapter, which will allow us to make additional independency assumptions about the model.

In this thesis, the observed variables will generally be a corpus of n sentences of text s , where each s_i is a sequence of m_i words $s_{i1} \dots s_{im_i}$. Each word s_{ij} is associated with a part-of-speech τ_{ij} , which will also be treated as observed. The (here finite) set of all possible words is denoted as V_w , and the set of possible parts-of-speech is denoted V_τ .

The hidden variables will be dependency parse trees $\mathbf{t} = \{t_i\}_{i=1,n}$. A dependency tree t_i is a directed acyclic graph whose nodes are the words in s_i . The graph has a single



Figure 2.1: Example dependency parse.

incoming edge for each word s_{ij} , except one word, denoted as the *root* of the graph. An edge from word s_{ij} to $s_{ij'}$ means that $s_{ij'}$ is an *argument* of s_{ij} , or alternatively, s_{ij} is the *head* of $s_{ij'}$. Note that each word token may be the argument of only one head, but a head may have several arguments.

If t_i can be drawn on a plane above the sentence with no crossing edges, it is called *projective*. Otherwise it is *nonprojective*. While there are languages whose dependency structures have crossing edges, and there are supervised algorithms for learning and parsing nonprojective dependency structures, no such algorithms currently are known in the unsupervised setting. The algorithms we consider here only examine projective structure.

2.1.1 Split Bilexical CFGs

In order to efficiently estimate the dependency models in this thesis, we will need to devise CFGs that factor their parsing decisions in the same way. In the sections that follow, we frame various dependency models as a particular variety known as split bilexical CFGs (Eisner and Satta, 1999). These will allow us to use the much faster Eisner-Satta (Eisner and Satta, 1999) parsing algorithm to compute our dynamic programming steps in $O(|s_i|^3)$ time (Eisner and Blatz, 2007; Johnson, 2007). (Efficiently parsable versions of split bilexical CFGs for the models described in this dissertation can be derived using the fold-unfold grammar transform (Eisner and Blatz, 2007; Johnson, 2007)).

In the split-head bilexical CFG framework each nonterminal in the grammar is annotated with a terminal symbol. We will define grammars in this framework in terms of rule schemas and nonterminal schemas. For instance we might have a series of rules

$X_H \rightarrow Y_{H'}Z_H$, for all pairs $H, H' \in T$, where X, Y, Z are nonterminals with the annotations removed. This one schema would provide $|T|^2$ rules. For dependency grammars, these annotations correspond to words and/or parts-of-speech. The second important property of split-head bilexical CFGs is that each observed symbol τ_{ij} in a sentence is represented in a split form, consisting of a left part τ_{ijL} and a right part τ_{ijR} (McAllester, 1999). These parts become the terminal symbols of the grammar. This split-head property relates to a particular type of dependency grammar, in which the left and right dependents of a head are generated independently.

Note that split-head bilexical CFGs can be made probabilistic in the same way as standard PCFGs.

A simple example of a split-head bilexical CFG (denoted Simple Split-Head Grammar 1) for dependency parsing is:

Simple Split-Head Grammar 1	
Rule	Description
$S \rightarrow Y_H$	select H as root
$Y_H \rightarrow L_H R_H$	Move to split-head representation
$L_H \rightarrow H_L$	no more arguments to left of H
$L_H \rightarrow Y_A L_H$	argument = A to left of head = H
$R_H \rightarrow H_R$	no more arguments to right of H
$R_H \rightarrow R_H Y_A$	argument = A to right of head = H

Here H_L, H_R are the terminals of the grammar; there is an H_L and an H_R for each H in

the vocabulary V_w . The expansion of L_H encodes the decision of whether there is another left argument of H , and if so, what it should be. Likewise R_H is a nonterminal encoding the decision of whether there is another right argument of H and what that argument should be. An example parse of “The big dog barks” is given in 2.2.

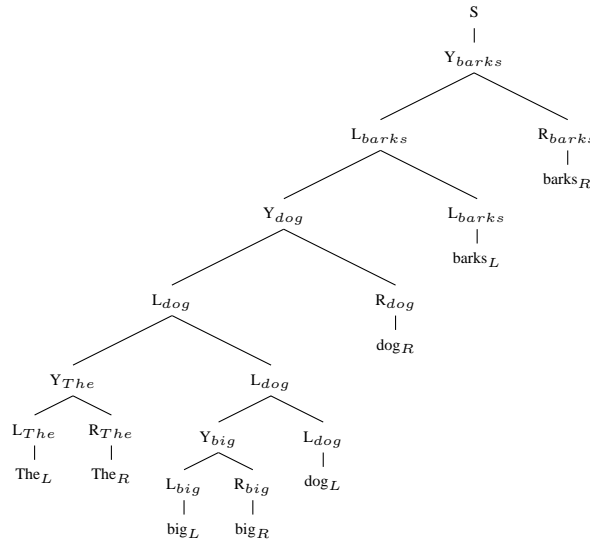


Figure 2.2: Simple split-head bilexical CFG parse of “The big dog barks.”

Note that this model combines the decision about whether to generate another argument (a stopping decision) with the decision about what that argument should be. For instance, in Figure 2.2, the rule $L_{barks} \rightarrow Y_{dog} L_{barks}$ combines deciding that “barks” should have another left argument, and selecting “dog” as that argument. Separating these will simplify the derivation of our later models –specifically the introduction of valence. They can be separated performing the following transformations to the grammar. First we add the nonterminals L_H^1 and R_H^1 for each $H \in V_w$, which signifies a state in which we know we will generate at least one argument to the left and right respectively, and must decide which argument to generate. Second, we replace the rule $L_H \rightarrow Y_A L_H$ with the pair of rules $L_H \rightarrow L_H^1$ and $L_H^1 \rightarrow Y_A L_H$ (and likewise replace $R_H \rightarrow R_H Y_A$ with $R_H \rightarrow R_H^1$ and

$R_H^1 \rightarrow R_H Y_A$). Now for instance L_H indicates a decision about whether to stop generating arguments to the left, or to generate at least one more argument. An example of the difference between the old and new grammars are given in Figure 2.3. The transformations yield this new grammar (Simple Split-Head Grammar 2):

Simple Split-Head Grammar 2	
Rule	Description
$S \rightarrow Y_H$	select H as root
$Y_H \rightarrow L_H R_H$	Move to split-head representation
$L_H \rightarrow L_H^0$	stop generating arguments left,head= H
$L_H \rightarrow L_H^1$	continue generating arguments left,head= H
$L_H^1 \rightarrow Y_A L_H$	argument= A left,head= H
$L_H^0 \rightarrow H_L$	
$R_H \rightarrow R_H^0$	stop generating arguments right,head= H
$R_H \rightarrow R_H^1$	continue generating arguments right,head= H
$R_H^1 \rightarrow R_H Y_A$	argument= A right,head= H
$R_H^0 \rightarrow H_R$	

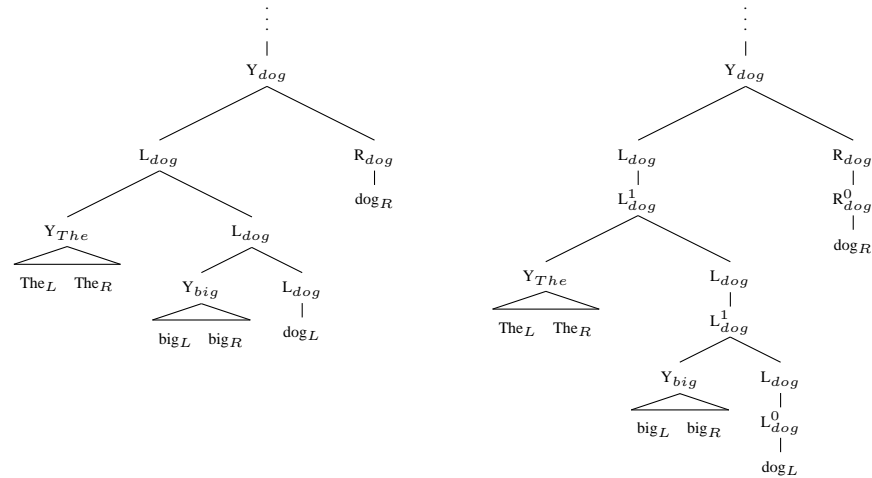


Figure 2.3: An example of moving from Simple Split-Head Grammar 1 to Simple Split-Head Grammar 2, which separates the decision of whether to generate another argument (L_H) from what that argument should be (L_H^1).

2.1.2 Dependency Model with Valence

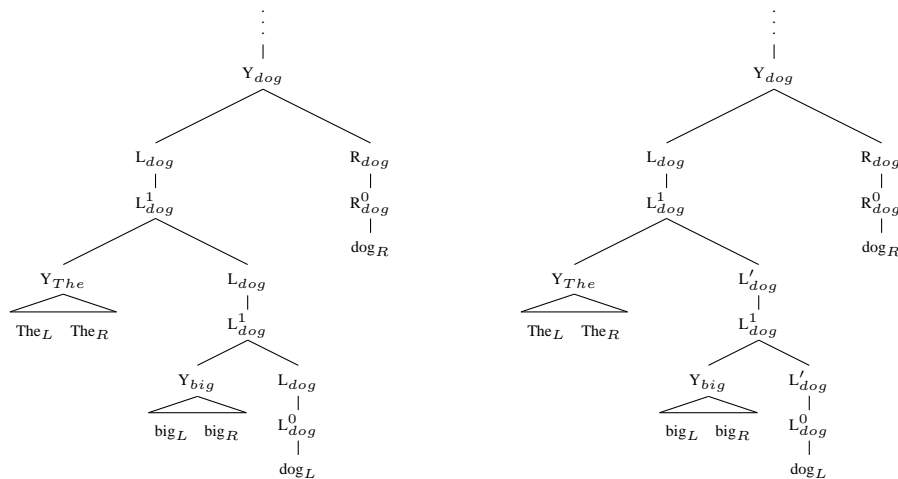


Figure 2.4: An example of moving from Simple Split-Head Grammar 2 to DMV. DMV distinguishes between the deciding if there will be a first argument, and deciding when to stop generating subsequent arguments. Notice the lower two L_{dog} nonterminals in the left tree becomes L'_{dog} in the second.

The most successful recent work on dependency induction has focused on the Dependency Model with Valence (DMV) of Klein and Manning (Klein and Manning, 2004).

Dependency Model with Valence (DMV)	
Rule	Description
$S \rightarrow Y_H$	select H as root word
$Y_H \rightarrow L_H R_H$	Move to split-head representation
$L_H \rightarrow L_H^0$	stop generating arguments left,head= H , no arguments
$L_H \rightarrow L_H^1$	continue generating arguments left,head= H no arguments
$L'_H \rightarrow L_H^0$	stop generating arguments left,head= H , one or more arguments
$L'_H \rightarrow L_H^1$	continue generating arguments left,head= H , one or more arguments
$L_H^1 \rightarrow Y_A L'_H$	argument= A left,head= H
$L_H^0 \rightarrow H_L$	
$R_H \rightarrow R_H^0$	stop generating arguments right,head= H , no arguments
$R_H \rightarrow R_H^1$	continue generating arguments right,head= H no arguments
$R'_H \rightarrow R_H^0$	stop generating arguments right,head= H , one or more arguments
$R'_H \rightarrow R_H^1$	continue generating arguments right,head= H , one or more arguments
$R_H^1 \rightarrow R'_H Y_A$	argument= A right, head= H
$R_H^0 \rightarrow H_R$	

Table 2.1: CFG Schema for Dependency Model with Valence.

The main difference between DMV and the Simple Split-Head Grammar 2 is that DMV distinguishes the probability of the decision to generate the first argument in a particular direction from the probability of deciding to generate subsequent arguments. This is the sense in which it models valence. We can incorporate this in a split-head bilexical CFG by splitting the L_H nonterminal into L_H and L'_H (likewise R_H becomes R_H and R'_H). The rule $L_H^1 \rightarrow Y_A L_H$ becomes $L_H^1 \rightarrow Y_A L'_H$ (likewise $R_H^1 \rightarrow R_H Y_A$ becomes $R_H^1 \rightarrow R'_H Y_A$). L_H now indicates a decision of whether to generate the first argument, and L'_H a decision of whether to generate subsequent arguments. This results in a grammar given in Table 2.1.

An example of the difference between grammar 2 and DMV is shown in figure 2.4.

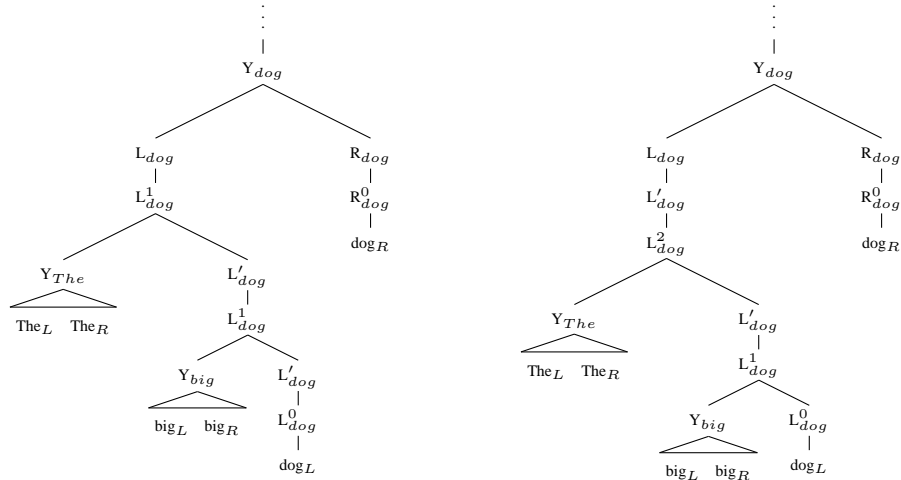


Figure 2.5: An example of moving from DMV to EVG. The key difference is that EVG distinguishes between the distributions over the argument nearest the head (*big*) from arguments farther away (*The*).

(In fact all work implementing DMV has replaced the words with their parts of speech. However for expositional clarity the example shows words). In this example the lower two L_{dog} nonterminals in the grammar 2 tree become, L'_{dog} in DMV; however the top one remains as L_{dog} , distinguishing the first argument decision from subsequent ones.

2.1.3 Extended Valence Grammar

Among the models we discuss in this dissertation are some derived from a variant of DMV presented by McClosky (McClosky, 2008), called the *Unrestricted Valence Grammar* or *Extended Valence Grammar* (EVG). The main insight is that for a given head, different valence positions in a given direction should have different distributions over arguments. In particular, EVG modifies DMV by distinguishing the distribution over the argument nearest the head from the distribution of subsequent arguments. For example, in the phrase “the big hungry dog”, the distribution over “hungry” as the closest left argument of “dog” would be different than the distribution over “the” and over “big”.

Consider the following changes to the DMV grammar. First, we will introduce the

nonterminals L_H^2 and R_H^2 and rules $L_H^2 \rightarrow Y_A L'_H$ and $R_H^2 \rightarrow R'_H Y_A$ to denote the decision of what argument to generate for positions not next to the head. Next instead of expanding $L'_H \rightarrow L_H^0 | L_H^1$ we will expand it as $L'_H \rightarrow L_H^1 | L_H^2$. L'_H still represents the decision of whether to keep generating arguments to the left, given there is at least one. However, L_H^1 now indicates that there is exactly one left argument remaining (that nearest the head), and so the rule $L_H^1 \rightarrow Y_A L'_H$ must become $L_H^1 \rightarrow Y_A L_H^0$ (i.e. generate left argument A , and no more). See Figure 2.5 for an example. These transformations yield the grammar in Table 2.2.

It is important to note that in previous work, as with DMV, EVG has only been estimated using parts-of-speech ignoring the words.

2.2 Experimental Setup

Dependency parses are typically evaluated against an annotated data set (the “gold-standard”). The standard metric is *directed accuracy*, which is the percent of directed edges proposed by the parser that match those present in the gold standard.

The experiments presented here use the Penn Treebank Wall Street Journal corpus (Marcus et al., 1993). We follow the now standard practice by Klein and Manning 2002 (Klein and Manning, 2002) of deleting punctuation, and using only sentences with 10 words or fewer. This corpus will henceforth be referred to as WSJ10. The dependencies are extracted from the phrase structure trees using the rules by Yamada and Matsumoto 2003 (Yamada and Matsumoto, 2003). We train on sections 2-21, use the likelihood of section 22 to evaluate convergence, use section 24 for development, and section 23 as our final test set.

We will present results on both the training set, and on the development/final-test sections. Typically in unsupervised learning problems we would only have one data-set (akin

Extended Valence Grammar (EVG)	
Rule	Description
$S \rightarrow Y_H$	select H as root
$Y_H \rightarrow L_H R_H$	Move to split-head representation
$L_H \rightarrow L_H^0$	stop generating arguments left,head = H , no arguments
$L_H \rightarrow L'_H$	continue generating arguments left,head = H no arguments
$L'_H \rightarrow L_H^1$	stop generating arguments left,head = H , one or more arguments
$L'_H \rightarrow L_H^2$	continue generating arguments left,head = H , one or more arguments
$L_H^2 \rightarrow Y_A L'_H$	argument = A left,head = H , argument is not nearest to head
$L_H^1 \rightarrow Y_A L_H^0$	argument = A left,head = H , argument is nearest to head
$L_H^0 \rightarrow H_L$	
$R_H \rightarrow R_H^0$	stop generating arguments right,head = H , no arguments
$R_H \rightarrow R'_H$	continue generating arguments right,head = H no arguments
$R'_H \rightarrow R_H^1$	stop generating arguments right,head = H , one or more arguments
$R'_H \rightarrow R_H^2$	continue generating arguments right,head = H , one or more arguments
$R_H^2 \rightarrow R'_H Y_A$	argument = A right,head = H , argument is not nearest to head
$R_H^1 \rightarrow R_H^0 Y_A$	argument = A right,head = H , argument is not nearest to head
$R_H^0 \rightarrow H_R$	

Table 2.2: CFG Schema for the Extended Valence Grammar.

to the training set), and we would evaluate how well we represent the underlying structure of that set. Here we will present results on both sets, to give a sense for now only how well the learned model represents the given data, but also how well the model generalizes to novel sentences. This also allows us to compare to previous work such as (Cohen et al., 2008).

To evaluate models learned using Expectation Maximization, we examine the Viterbi parse under the learned parameter vector θ . To evaluate models learned using Variational Bayes, we follow Cohen et al. 2008 (Cohen et al., 2008) in using the mean of the variational posterior Dirichlets as a point estimate.

2.3 Tied Probabilistic Context Free Grammars

In order to perform smoothing in PCFGs in Chapter 3, we will find useful a class of PCFGs in which the probabilities of certain rules are required to be the same. This will allow us to make independence assumptions for smoothing purposes without losing information, by giving analogous rules the same probability. For instance, we might have a grammar rule $L_H^2 \rightarrow Y_A L'_H$ and might want the probability of the rule to not depend on the head part-of-speech H , but want the fact that the head is H to propagate to the righthand side nonterminal L'_H , and so be available to condition on farther down the tree.

Let tuple $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{R})$ be a Context Free Grammar (CFG) with nonterminal symbols \mathcal{N} , terminal symbols \mathcal{T} , start symbol $S \in \mathcal{N}$, and set of rewrite productions \mathcal{R} of the form $N \rightarrow \beta$, for $N \in \mathcal{N}, \beta \in (\mathcal{N} \cup \mathcal{T})^*$. Let \mathcal{R}_N indicate the subset of \mathcal{R} whose left-hand sides are N .

(G, θ) defines a Probabilistic Context Free Grammar (PCFG), where θ is a vector of length $|\mathcal{R}|$, indexed by productions $N \rightarrow \beta \in \mathcal{R}$. $\theta_{N \rightarrow \beta}$ specifies the probability that N rewrites to β . Hence $\sum_{N \rightarrow \beta \in \mathcal{R}_N} \theta_{N \rightarrow \beta} = 1$. We will let θ_N denote the subvector of θ

corresponding to rules in \mathcal{R}_N .

A tied PCFG specifies that certain nonterminals have common distributions over the indexes of their rules. For instance, if we have $N \rightarrow \beta_1|\beta_2$ and $N' \rightarrow \beta'_1|\beta'_2$, we want to be able to specify that, e.g. $\theta_{N \rightarrow \beta_1} = \theta_{N' \rightarrow \beta'_1}$ and $\theta_{N \rightarrow \beta_2} = \theta_{N' \rightarrow \beta'_2}$.

We define a tied PCFG $H = (G, \theta, \overset{H_{\mathcal{R}}}{\equiv})$, where $\overset{H_{\mathcal{R}}}{\equiv}$ is an equivalence relation on \mathcal{R} that satisfies the following properties:

1.	$\forall r_1 r_2 \in \mathcal{R}$, if $r_1 \overset{H_{\mathcal{R}}}{\equiv} r_2$ then $\theta_{r_1} = \theta_{r_2}$	Tied rules have the same probability.
2.	$\forall r_1, r_2 \in \mathcal{R}_N$, $r_1 \neq r_2$ $r_1 \not\overset{H_{\mathcal{R}}}{\equiv} r_2$	Distinct rules expanding the same nonterminal are never tied.
3.	$\forall N_1 \neq N_2, \forall r_1 \in \mathcal{R}_{N_1}, r_2 \in \mathcal{R}_{N_2}$ if $r_1 \overset{H_{\mathcal{R}}}{\equiv} r_2$ then for all $\forall r'_1 \in \mathcal{R}_{N_1}$: $\exists r'_2 \in \mathcal{R}_{N_2}$ such that $r'_1 \overset{H_{\mathcal{R}}}{\equiv} r'_2$	If any rule in \mathcal{R}_{N_1} is tied to a rule in \mathcal{R}_{N_2} then each rule in \mathcal{R}_{N_1} is tied to a rule in \mathcal{R}_{N_2} and vice versa.

We call $\overset{H_{\mathcal{R}}}{\equiv}$ the tying relation. If $N_1 \rightarrow \beta_1$ and $N_2 \rightarrow \beta_2$ are tied then the tying relation defines a one-to-one mapping between rules in \mathcal{R}_{N_1} and \mathcal{R}_{N_2} . This can be seen from the following: (3) says that each rule in \mathcal{R}_{N_1} is tied to a rule in \mathcal{R}_{N_2} and vice versa. (2) plus the transitivity of an equivalence relation indicates that each rule in \mathcal{R}_{N_1} is tied to a unique rule in \mathcal{R}_{N_2} (and vice versa). Hence the relation is one-to-one.

Clearly, the tying relation also defines an equivalence class over nonterminals, and we say that N_1 and N_2 are tied nonterminals if there is an $r_1 \in \mathcal{R}_{N_1}$ and $r_2 \in \mathcal{R}_{N_2}$ where $r_1 \overset{H_{\mathcal{R}}}{\equiv} r_2$. The tying relation allows us to formulate the distributions over trees in terms of rule equivalence classes and nonterminal equivalence classes. Suppose $\bar{\mathcal{R}}$ is the set

of rule equivalence classes and $\bar{\mathcal{N}}$ is the set of nonterminal equivalence classes. Since all rules in an equivalence class \bar{r} have the same probability (condition 1), and since all the nonterminals in an equivalence class $\bar{N} \in \bar{\mathcal{N}}$ have the same distribution over rule equivalence classes (condition 1 and 3), we can define the set of rule equivalence classes $\bar{\mathcal{R}}_{\bar{N}}$ associated with a nonterminal equivalence class \bar{N} , and a vector $\bar{\theta}$ of probabilities, indexed by rule equivalence classes $\bar{r} \in \bar{\mathcal{R}}$. $\bar{\theta}_{\bar{N}}$ refers to the subvector of $\bar{\theta}$ associated with nonterminal equivalence class \bar{N} , indexed by $\bar{r} \in \bar{\mathcal{R}}_{\bar{N}}$. Since rules in the same equivalence class have the same probability, we have that for each $r \in \bar{r}$, $\theta_r = \bar{\theta}_{\bar{r}}$.

Let $f(\mathbf{t}, r)$ denote the number of times rule r appears in tree \mathbf{t} , and let $f(\mathbf{t}, \bar{r}) = \sum_{r \in \bar{r}} f(\mathbf{t}, r)$. We see that the complete data likelihood is

$$P(\mathbf{s}, \mathbf{t} | \theta) = \prod_{\bar{r} \in \bar{\mathcal{R}}} \prod_{r \in \bar{r}} \theta_r^{f(\mathbf{t}, r)} = \prod_{\bar{r} \in \bar{\mathcal{R}}} \bar{\theta}_{\bar{r}}^{f(\mathbf{t}, \bar{r})}$$

Let $\bar{\theta}_{\bar{N}}$ be the subvector of $\bar{\theta}$ indexed by $\bar{r} \in \bar{\mathcal{R}}_{\bar{N}}$. $\bar{\theta}_{\bar{N}}$ is a multinomial parameter vector (i.e., $\sum_{\bar{r} \in \bar{\mathcal{R}}_{\bar{N}}} \bar{\theta}_{\bar{r}} = 1$), and so $p(\mathbf{t} | \bar{\theta})$ is a product of multinomials, one for each $\bar{N} \in \bar{\mathcal{N}}$:

$$p(\mathbf{t} | \bar{\theta}) = \prod_{\bar{r} \in \bar{\mathcal{R}}} \bar{\theta}_{\bar{r}}^{f(\mathbf{t}, \bar{r})} = \prod_{\bar{N} \in \bar{\mathcal{N}}} \prod_{\bar{r} \in \bar{\mathcal{R}}_{\bar{N}}} \bar{\theta}_{\bar{r}}^{f(\mathbf{t}, \bar{r})}$$

That is, the likelihood is a product of multinomials, one for each nonterminal equivalence class, and there are no constraints placed on the parameters of these multinomials besides being positive and summing to one. This means that all the standard estimation methods (e.g. Expectation Maximization, Variational Bayes, sampling), in particular the efficient dynamic programming algorithms for estimating sufficient statistics over PCFGs, extend directly to tied PCFGs.

2.4 Estimation

We have so far specified a framework by which one can model simple probabilistic dependency grammars. The next issue is to discuss briefly how we might determine the probabilities of each rule or rule equivalence class in the grammar. We will touch first upon techniques that will be of use in the remainder of the dissertation; some other estimation techniques in the literature will be discussed at the end of the section.

Perhaps the most straightforward approach is to assign probabilities to maximize the likelihood of the observed sentences:

$$\begin{aligned}\theta_{MLE} &= \operatorname{argmax}_{\theta} p(\mathbf{s}|\theta) \\ &= \operatorname{argmax}_{\theta} \sum_{\mathbf{t}} p(\mathbf{s}, \mathbf{t}|\theta)\end{aligned}$$

The standard technique for performing maximum likelihood estimation when the likelihood is stated in terms of hidden variables such as the parse trees \mathbf{t} is the Expectation Maximization algorithm (EM). Expectation Maximization is an iterative technique for finding a local maximum of the observed data likelihood, which alternates between two steps. First some initial setting for the parameters is chosen for $\theta^{(0)}$. In the l th iteration, the *E-Step* calculates for each sentence s_i and possible tree t_i $p(t_i|s_i, \theta^{(l-1)})$. The *M-Step* assigns $\theta^{(l)}$ to maximize $E_{p(\mathbf{t}|\mathbf{s}, \theta^{(l-1)})} \log p(\mathbf{s}, \mathbf{t}|\theta^{(l-1)})$.

For PCFGs, the M-Step is straightforward to implement given the expected count $f(\mathbf{t}, r)$ of each rule equivalence class r under the distribution $p(\mathbf{t}|\mathbf{s}, \theta^{(l-1)})$ calculated in the E-Step:

$$\theta_r^{(l)} \propto E_{p(\mathbf{t}|\mathbf{s}, \theta^{(l-1)})} f(\mathbf{t}, r)$$

To process the E-Step, there is an efficient dynamic programming algorithm (Baker,

1979) for PCFGs, which calculates exactly these expected counts. Together these are known as the Inside-Outside algorithm.

Maximum likelihood estimation provides a point estimate of θ . However, often we want to incorporate information about θ by modeling its *prior* distribution, and model uncertainty by estimating a *posterior* distribution. As a prior, for each $N \in \mathcal{N}$ we will specify a Dirichlet distribution over θ_N with hyperparameters α_N . The Dirichlet has the density function:

$$P(\theta_N | \alpha_N) = \frac{\Gamma(\sum_{r \in \mathcal{R}_N} \alpha_r)}{\prod_{r \in \mathcal{R}_N} \Gamma(\alpha_r)} \prod_{r \in \mathcal{R}_N} \theta_r^{\alpha_r - 1},$$

Thus the prior over θ is a product of Dirichlets, which is *conjugate* to the PCFG likelihood function (Johnson et al., 2007). That is, the posterior $P(\theta | \mathbf{s}, \mathbf{t}, \alpha)$ is also a product of Dirichlets, also factoring into a Dirichlet for each nonterminal N , where the parameters α_r are augmented by the number of times rule r is observed in tree \mathbf{t} :

$$\begin{aligned} P(\theta | \mathbf{s}, \mathbf{t}, \alpha) &\propto P(\mathbf{s}, \mathbf{t} | \theta) P(\theta | \alpha) \\ &\propto \prod_{r \in \mathcal{R}} \theta_r^{f(\mathbf{t}, r) + \alpha_r - 1} \end{aligned}$$

We can see that α_r acts as a pseudocount of the number of times r is observed prior to \mathbf{t} .

To make use of this prior, we use the Variational Bayes (VB) technique for PCFGs with Dirichlet Priors presented by (Kurihara and Sato, 2004). VB estimates a distribution over θ . In contrast, Expectation Maximization estimates merely a point estimate of θ . In VB, one estimates $Q(\mathbf{t}, \theta)$, called the variational distribution or variational posterior, which approximates the posterior distribution $P(\mathbf{t}, \theta | \mathbf{s}, \alpha)$ by minimizing the KL divergence of P from Q . Minimizing the KL divergence, it turns out, is equivalent to maximizing a lower

bound \mathcal{F} of the log marginal likelihood $\log P(\mathbf{s}|\alpha)$.

$$\log P(\mathbf{s}|\alpha) \geq \sum_{\mathbf{t}} \int_{\theta} Q(\mathbf{t}, \theta) \log \frac{P(\mathbf{s}, \mathbf{t}, \theta|\alpha)}{Q(\mathbf{t}, \theta)} = \mathcal{F}$$

The negative of the lower bound, $-\mathcal{F}$, is sometimes called the *free energy*.

As is typical in variational approaches, Kurihara and Sato (2004) make the “mean field” assumption, in which the hidden variables and parameters in the variational posterior are independent. They assume a factorization:

$$Q(\mathbf{t}, \theta) = Q(\mathbf{t})Q(\theta) = \prod_{i=1}^n Q_i(t_i|s_i) \prod_{N \in \mathcal{N}} Q(\theta_N)$$

The goal is to recover $Q(\theta)$, the estimate of the posterior distribution over parameters and $Q(t)$, the estimate of the posterior distribution over trees. Finding a local maximum of \mathcal{F} is done via an alternating maximization of $Q(\theta)$ and $Q(\mathbf{t})$. Kurihara and Sato (2004) show that each $Q(\theta_N)$ is a Dirichlet distribution with parameters $\hat{\alpha}_r = \alpha_r + E_{Q(\mathbf{t})}f(\mathbf{t}, r)$, and that

$$Q(\mathbf{t}) = \prod_{s \in \mathbf{s}} Q(t|s) = \frac{\prod_{r \in \mathcal{R}} \pi_r^{f(t,r)}}{\sum_{t'} \prod_{r \in \mathcal{R}} \pi_r^{f(t',r)}}$$

where for $r \in \mathcal{R}_N$:

$$\pi_r = \exp \left(\psi(\hat{\alpha}_r) - \psi \left(\sum_{r' \in \mathcal{R}_N} \hat{\alpha}_{r'} \right) \right)$$

Estimation of $Q(\mathbf{t})$ and $E_{Q(\mathbf{t})}f(\mathbf{t}, r)$ is performed using a variation of the inside-outside algorithm, replacing each rule probability θ_r with π_r .

Another option for performing Bayesian inference is a sampling approach using Markov

Chain Monte Carlo (MCMC). Johnson et al. (2007) present two approaches: a Gibbs sampler to draw samples from $P(\mathbf{t}, \theta | \mathbf{s}, \alpha)$, and a Hastings sampler for drawing samples from $P(\mathbf{t} | \mathbf{s}, \alpha)$. Our preliminary experiments found that these experienced the same severe local maxima problems experienced by Expectation Maximization and Variational Bayes approaches we describe in the next section. However, since the aim of a sampling approach is to produce samples distributed according to a certain distribution, the random restart and select the maximum approach we advocate for those is not so principled for sampling.

2.4.1 Initialization and Search

Both the Expectation Maximization and Variational Bayes approach locally maximize their respective objective functions. In practice for probabilistic dependency grammars there exist many local maxima, and most of the correspond to qualitatively different dependency grammars (Carroll and Charniak, 1992). We can observe this property in Figure 2.6 which shows a graph of dependency accuracy vs log likelihood of DMV, for 100 randomly initialized, Expectation Maximization trained models. Each point represents a local maxima in the log likelihood, and one can see that they are quite spread out. Here and subsequently we randomly initialize by assigning each θ_N as a sample from a symmetric Dirichlet with parameter 1.

Faced with this fact about the likelihood space, several approaches have arisen in the literature. One approach is an initial parameter setting which is likely to be close to a good portion of the space. Klein and Manning (2004) propose their “harmonic” initializer, which incorporate the linguistic intuition that shorter dependencies are preferable to longer, in the hope that portions of the parameter space that are near this are probably better. This initializer is used in much of the subsequent literature, including (Smith, 2006; Cohen et

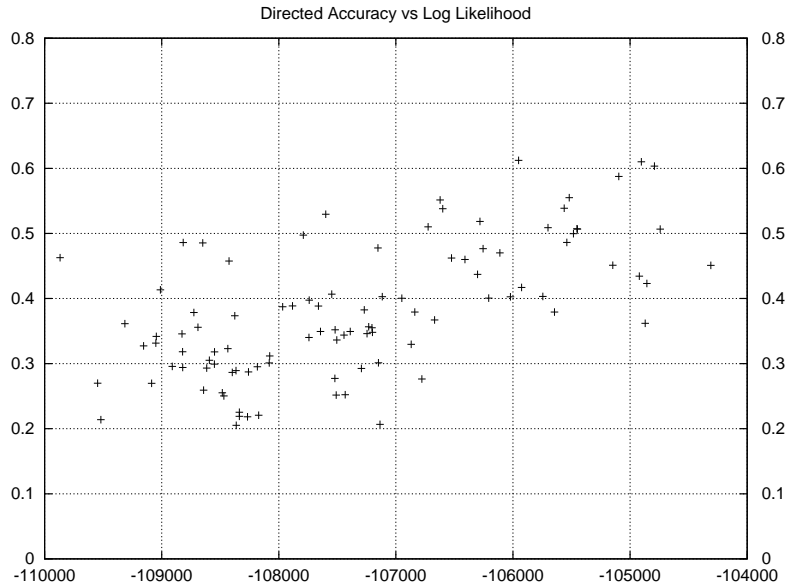


Figure 2.6: Directed Accuracy vs Log Likelihood, 100 runs of Expectation Maximization trained DMV.

al., 2008; Cohen and Smith, 2009; Daumé III, 2009)¹.

Smith (2006) proposes several techniques for dealing with the local maxima problem. The Skewed Deterministic Annealing and Structural Annealing techniques attempt to bias the initial parameter settings to reflect this intuition that short dependencies are better, slowly removing the bias over the course of learning. Deterministic Annealing attempts to flatten the likelihood surface in the hopes of finding maxima in the likelihood surface with higher likelihood.

It does so by essentially making the E-step distribution over \mathbf{t} initially similar to uniform, and slowly removing this bias. This is equivalent to setting each rule probability $\theta_{N \rightarrow \beta} = \frac{1}{|\mathcal{R}_N|}$. While the likelihoods it finds are indeed higher than straightforward Expectation Maximization, the accuracies are poor. Skewed Deterministic Annealing improves

¹Thanks to Noah Smith for providing his implementation of the harmonic initializer

upon this by starting with a bias towards a different initializer (such as the Klein and Manning Harmonic Initializer).

In this thesis our view is that both the likelihood and the variational lower bound should provide a sufficient signal for reasonable dependency learning. In particular we will instead use the technique of random restart to explore many local maxima, selecting the one with the best value of the objective function of interest. The main idea of this scheme is that we would like to wind up in a “good” part of the space in terms of accuracy, most of the time. Note in Figure 2.6 that there are far more points in the upper right than in the lower right corner.

For some distribution over initial parameters $F_0(\theta)$, there will be a corresponding distribution over the converged states of parameters $F_c(\theta)$, resulting from running Expectation Maximization starting from an initial state drawn from $F_0(\theta)$. (The analogue to this in the Variational Bayes case is a distribution over initial variational hyperparameters $\hat{\alpha}$). Our goal is that the expected accuracy should be high.

Figures 2.7 and 2.8 show the results of 100 runs of estimating DMV using Variational Bayes, where each run is given 20 random restarts. Each restart was run for 40 iterations, and the model with the highest lower bound value was run until convergence. We can see that this compresses the resulting grammars into a higher accuracy portion of the space.

In the experiments in the rest of the dissertation, we will present results where each run has 600 random restarts, run for 40 iterations. The highest lower bound restart is selected from each group of 20 restarts (i.e. 30 in sum), and run until convergence. The best of these 30 is then selected according to lower bound value. We report results averaged over 10 runs. To account for the variance introduced through the randomized initialization, we also report 95% confidence intervals calculated using bootstrap sampling where feasible (See Appendix C).

Algorithm 1 Estimation for Variational Bayes with Randomized Initialization

```

for  $m$  from 1 to  $M$  do
  for  $b$  from 1 to  $B$  do
    Sample  $\tilde{\theta}_N^b \sim \text{DIR}(\alpha_N)$ 
    Let  $Q(\mathbf{t})^{(0)} \leftarrow P(\mathbf{t}|\mathbf{s}, \tilde{\theta}^b)$ 
    Let  $\hat{\alpha}_r \leftarrow \alpha_r + E_{Q(\mathbf{t})^{(0)}} f(\mathbf{t}, r)$ 
    for  $i$  from 1 to 40 do
      Iterate Variational Bayes on model  $b$ .
    end for
  end for
  end for
  Select model  $b^* = \text{argmax}_b \mathcal{F}(Q_b(t), Q_b(\theta))$ 
  while TEST_CONVERGENCE=FALSE do
    Iterate Variational Bayes on model  $Q_{b^*}$ 
  end while
  Let model  $Q_m$  be converged  $Q_{b^*}$ 
end for
return model  $m^* = \text{argmax}_m \mathcal{F}(Q_m(t), Q_m(\theta))$ 

```

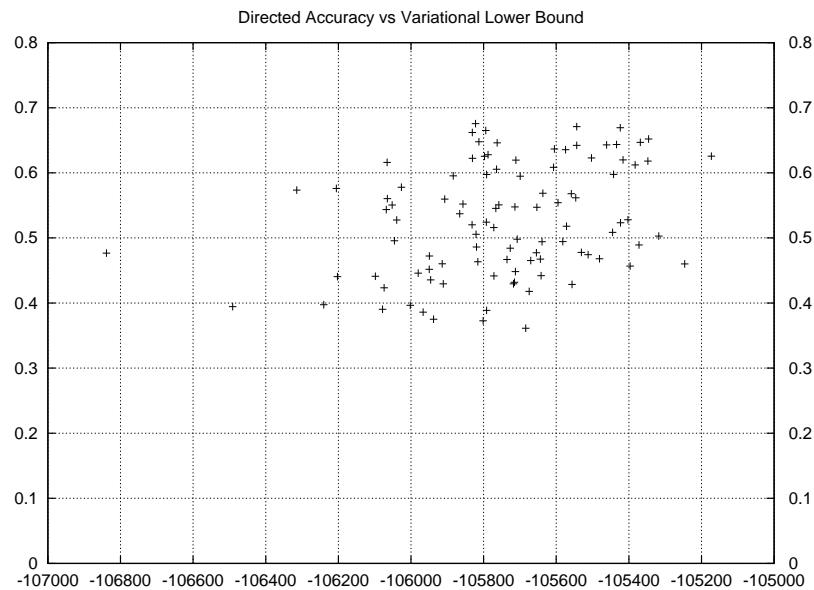


Figure 2.7: Directed Accuracy vs Lower Bound, 100 runs of Variational Bayes trained DMV, each with 20 random restarts.

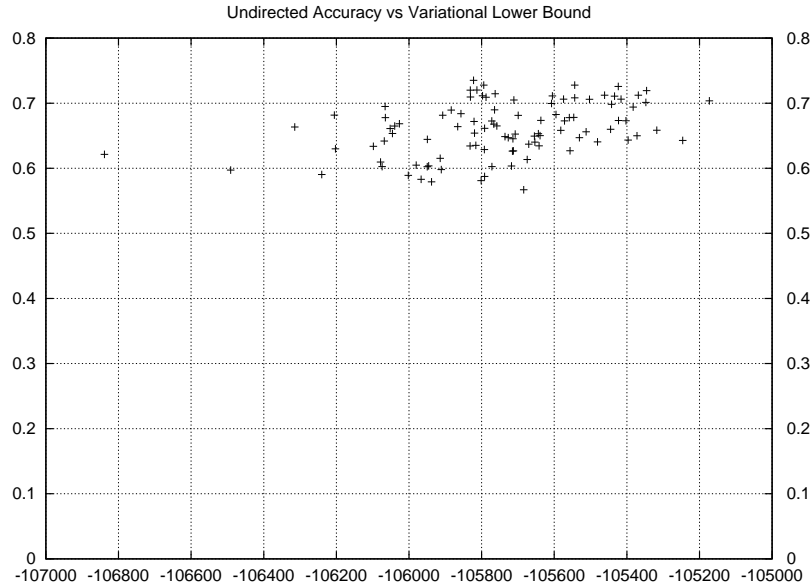


Figure 2.8: Undirected Accuracy vs Lower Bound, 100 runs of Variational Bayes trained DMV, each with 20 random restarts.

		Directed Acc		Undirected Acc.	
Model	Initialization	Train	Dev	Train	Dev
DMV	Harmonic	0.483	0.457	0.651	0.633
DMV	Random	0.583 ($+0.064$ -0.034)	0.549 ($+0.065$ -0.031)	0.689 ($+0.025$ -0.014)	0.668 ($+0.027$ -0.013)

Table 2.3: Results of Randomized vs Harmonic initialization for Variational Bayes

Table 2.3 describes the results of running DMV with both the Klein and Manning harmonic initializer, as well as with the randomized initialization approach. We can see using randomized initialization improves the average dependency accuracy by a great deal.

2.4.2 Analysis: Harmonic vs. Randomized Initialization

To see where our improvements are concentrated, we looked at the difference in correct directed dependencies under both the harmonic initializer and the randomized initializer, broken down by various categories, in Table 2.4 (by child part-of-speech and direction),

Table 2.5 (by head part-of-speech and direction), and Tables 2.6 and 2.7 (by head/child part-of-speech pair). Since the randomized initializer results are presented averaged over 10 runs in the experiments in this paper, the tables incorporate the sum of correct directed dependencies over these 10 runs. To allow comparison to the harmonic initializer, which is only given 1 run we have scaled its results up by 10. In these tables and those like it later in the thesis, we have placed in boldface differences with a magnitude greater than 1000, and italicized those greater than 100 and fewer than 1001, to emphasize the major net gains/losses between models.

In Table 2.4 we see that switching to randomized from harmonic initialization, the greatest net correct dependencies are with DT, noun (NN, NNP, NNS), CD, and PRP\$ children. However randomized initialization has net fewer correct dependencies with JJ, IN, CC, and verb children. Breaking the results down by head part-of-speech in Table 2.5, we see major improvements in dependencies headed by nouns and most verb types, as well as IN and CD. Dependencies headed by ROOT is the largest category in which randomized initialization does worse. Looking at Tables 2.6 and 2.7, we see that the single biggest improvement is with (NN,DT) dependencies, as well as (NNP,NNP). Additionally, major improvements come from NN/NNP children under VBD, VBZ, IN heads, as well as NNS children under VBD, IN and VBP. We see that the net fewer ROOT headed correct dependencies with randomized initialization mostly accrue to VBZ, VBD, VBP, and MD; randomized initialization actually gets more noun-rooted sentences correct.

2.5 Conclusion

In this chapter we described the basic learning framework we will build on in the remainder of the thesis. We saw how DMV and EVG can be described using Split-Head PCFGs, and covered several estimation techniques for PCFGs. We also described Tied-PCFGs, which

Child POS	Correct Harmonic	Correct Random	Difference (Random-Harmonic)	Difference Left	Difference Right
DT	2550	24405	21855	22011	-156
NN	22510	39688	17178	6648	10530
NNP	10800	24856	14056	11089	2967
NNS	15660	21306	5646	3265	2381
JJ	20520	15385	-5135	-5176	41
CD	6580	9712	3132	3463	-331
VBN	8970	5960	-3010	-60	-2950
IN	13270	10821	-2449	-40	-2409
CC	3120	964	-2156	-1920	-236
PRP\$	10	2095	2085	2085	0
VBZ	16040	14758	-1282	-1	-1281
VBD	16430	15440	-990	6	-996
VBP	9010	8051	-959	4	-963
MD	4830	3941	-889	-7	-882
VB	7750	7104	-646	11	-657
NNPS	520	1139	619	343	276
VBG	4550	4207	-343	-6	-337
POS	1850	1611	-239	-235	-4
EX	710	843	133	140	-7
RP	740	630	-110	0	-110
JJR	1010	908	-102	-91	-11
RB	6570	6667	97	-48	145
WP	670	576	-94	31	-125
PRP	14860	14946	86	20	66
JJS	350	265	-85	-78	-7
\$	720	635	-85	116	-201
LS	70	24	-46	-46	0
PDT	40	84	44	39	5
WDT	130	163	33	10	23
FW	60	89	29	27	2
TO	2620	2598	-22	434	-456
WRB	480	465	-15	13	-28
SYM	150	145	-5	5	-10
RBS	110	109	-1	-1	0
RBR	520	519	-1	8	-9
UH	170	170	0	31	-31

Table 2.4: Change in correct dependencies by child part-of-speech across 10 runs for DMV-Randomized and DMV-Harmonic. Left indicates child is a left dependent of its head in system output.

Head POS	Correct Harmonic	Correct Random	Difference (Random-Harmonic)	Difference Left	Difference Right
NN	18440	33387	14947	14980	-33
NNP	2530	10056	7526	8176	-650
IN	15140	21358	6218	40	6178
VBZ	21040	26550	5510	4507	1003
VBD	21850	27195	5345	4561	784
NNS	16140	20158	4018	4094	-76
ROOT	48260	45985	-2275	0	-2275
CD	530	2076	1546	1758	-212
VB	7610	9071	1461	424	1037
VBP	15650	17108	1458	1442	16
VBN	5370	4072	-1298	19	-1317
TO	1960	2855	895	-21	916
MD	8720	9592	872	1240	-368
NNPS	110	758	648	648	0
RB	1150	621	-529	88	-617
VBG	3070	3431	361	-42	403
POS	2550	2312	-238	-238	0
\$	1210	990	-220	37	-257
JJ	2230	2414	184	265	-81
WP	230	125	-105	-3	-102
WRB	360	305	-55	0	-55
JJR	320	375	55	55	0
DT	150	100	-50	-53	3
CC	170	120	-50	0	-50
FW	30	65	35	34	1
UH	0	27	27	27	0
PRP	60	74	14	19	-5
RBR	0	12	12	12	0
JJS	0	12	12	12	0
WDT	0	3	3	3	0
RP	0	3	3	0	3
SYM	70	69	-1	6	-7
RBS	0	0	0	0	0
PRP\$	0	0	0	0	0
PDT	0	0	0	0	0
LS	0	0	0	0	0
EX	0	0	0	0	0

Table 2.5: Change in correct dependencies by head part-of-speech across 10 runs for DMV-Randomized and DMV-Harmonic. Left indicates child is a left dependent of its head in system output.

	NN	NNP	DT	NNS	JJ	IN	RB	VBZ	VBD	CD	PRP	VB	VBP	VBN	TO	CC	VBG	MD	PRP\$
NN	-6	-559	17155	43	-4331	0	2	-9	-1	1462	0	1	0	-17	-9	53	46	0	1423
ROOT	<i>408</i>	1478	0	6	1	-4	-23	-1238	-887	-65	0	-45	-921	-270	0	-25	-27	-838	0
VBD	3786	1560	0	1181	12	72	<i>131</i>	6	-94	-16	18	39	-32	-696	-199	-430	-101	-24	0
VBZ	4856	1921	-2	<i>128</i>	-15	2	<i>141</i>	-31	49	41	23	47	3	-1011	-17	-578	-131	-11	0
NNS	47	-90	3736	28	<i>-943</i>	3	-41	0	0	492	11	0	-1	-49	0	<i>153</i>	93	0	<i>623</i>
IN	4304	<i>967</i>	-86	1056	34	-4	-19	22	-1	-12	5	1	33	6	-1	0	-36	-6	0
VBP	389	87	-24	1951	14	33	<i>171</i>	-1	-1	0	38	-3	-19	-540	-5	<i>-614</i>	-129	-2	0
NNP	-56	7909	<i>657</i>	0	-27	<i>-219</i>	1	0	0	-287	9	-6	6	-14	0	-422	0	0	11
VB	1105	<i>121</i>	-7	365	-10	-38	21	12	0	0	-10	29	-1	-379	<i>409</i>	-30	-19	7	0
MD	<i>761</i>	<i>324</i>	24	<i>339</i>	0	0	<i>307</i>	4	0	10	11	<i>-690</i>	2	0	0	-246	-5	0	0
VBN	<i>502</i>	-3	6	<i>166</i>	<i>125</i>	-2024	78	0	0	29	28	-7	0	4	<i>-165</i>	-37	-23	0	0
JJ	68	-10	5	59	-5	0	-82	0	0	33	-2	0	0	0	2	100	0	0	12
VBG	<i>498</i>	25	1	<i>214</i>	-6	<i>-236</i>	6	-3	0	15	-12	0	0	-38	-42	-39	-1	0	0
POS	-57	<i>-187</i>	16	-12	0	0	0	0	0	-1	0	0	0	0	0	0	-10	0	2
TO	<i>533</i>	59	0	69	-7	-6	0	0	0	<i>348</i>	-10	-4	0	0	0	0	-1	0	0
CD	-24	16	40	11	8	0	<i>-177</i>	0	0	1529	0	0	0	1	5	-19	0	0	0
RB	-7	0	32	0	18	0	<i>-370</i>	0	0	<i>-199</i>	0	1	0	1	0	-2	2	0	0
\$	8	0	0	0	5	0	24	0	0	<i>-258</i>	0	0	0	0	0	1	0	0	0
NNPS	0	<i>431</i>	<i>235</i>	0	-10	0	0	0	0	0	0	0	0	0	0	-11	0	0	0
JJR	47	0	15	26	7	4	-43	0	0	-3	-2	0	0	0	0	-4	0	0	7
WP	0	0	0	0	0	0	-13	-26	-42	0	0	0	-22	0	0	4	0	-6	0
DT	0	0	0	0	0	0	0	-8	1	14	-21	0	-6	0	0	0	-1	0	0
WRB	0	0	0	0	-1	0	0	-10	-14	0	0	-6	-1	-8	0	0	0	-15	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
CC	1	1	0	0	0	-28	-14	0	0	0	0	-3	0	0	0	0	0	6	0
RBR	0	0	0	10	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
JJS	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
PRP	14	6	12	0	0	0	-5	0	0	0	0	0	0	0	0	-13	0	0	0
FW	1	0	32	0	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	4
SYM	-3	0	0	6	0	-4	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.6: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for DMV-Randomized less DMV-Harmonic across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

	POS	JJR	\$	NNPS	RP	WP	RBR	EX	JJS	WRB	WDT	SYM	UH	PDT	RBS	FW	LS
NN	-139	-87	-34	6	0	0	0	0	-55	0	12	2	-18	7	0	0	0
ROOT	0	0	4	<i>195</i>	0	0	0	0	0	-16	0	-10	0	0	0	2	0
VBD	-6	-7	-10	<i>170</i>	-40	-8	3	30	0	-5	-2	0	0	0	0	7	0
VBZ	-3	-5	-1	45	-15	-25	0	71	3	5	4	-8	0	0	0	22	2
NNS	-69	-10	-10	7	0	0	0	0	-30	-2	6	1	0	63	0	0	0
IN	-2	-1	-89	63	0	-23	2	-7	-4	11	0	4	0	1	0	0	0
VBP	0	14	-9	<i>110</i>	-4	-19	0	29	2	-10	0	0	0	0	0	0	0
NNP	-18	0	0	-20	0	0	0	0	0	0	2	0	0	0	0	0	0
VB	-3	0	4	6	-75	-17	0	0	0	-6	1	0	8	0	0	0	-32
MD	0	0	0	9	0	0	0	10	0	14	8	0	-10	0	0	0	0
VBN	0	-1	5	6	27	-4	-4	0	7	0	0	4	0	0	-1	0	-16
JJ	0	0	-9	5	0	0	18	0	-10	0	0	0	0	0	0	0	0
VBG	0	-1	-12	-4	1	-2	4	0	2	0	0	0	-9	0	0	0	0
POS	1	0	0	12	0	0	0	0	0	0	0	0	0	0	0	-2	0
TO	-2	-6	-80	6	0	-4	0	0	0	0	0	0	0	0	0	0	0
CD	4	0	<i>156</i>	0	-4	0	0	0	0	0	0	0	0	0	0	0	0
RB	0	2	0	0	0	7	-24	0	0	0	0	0	6	4	0	0	0
\$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNPS	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
JJR	-2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DT	0	0	0	0	0	1	0	0	0	0	2	0	0	-32	0	0	0
WRB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	-3	0	0	0	0	0	0	0	-6	0	0	-4	0	0	0	0
RBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJS	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
PRP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
SYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.7: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for DMV-Randomized less DMV-Harmonic across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

can also be estimated using these techniques, and will prove useful to us for estimating the smoothed models of Chapters 3 and 4.

We also discussed various issues involved with initialization when estimating dependency grammars using Expectation Maximization and Variational Bayes. We found that there was a great benefit in terms of accuracy to using a randomized initialization scheme rather than the Klein and Manning (2004) harmonic initializer. One disadvantage of course is that this randomized approach requires many times more computing power than the specialized initializer approach. However, using randomized initialization does hopefully indicate that any improvements we find by changing the model will be attributable to changes in the model and not to the initializer.

Chapter 3

Smoothing PCFGs

3.1 Introduction

The models we have discussed so far in Chapter 2 make use of very simple features for unsupervised dependency grammar induction. We would like to integrate more interesting lexical features, which much previous work has shown. This will require using smoothing in an unsupervised setting. In this chapter we will first explore smoothing in a supervised setting, and then look at several ways in which it can be applied to the unsupervised setting. We will see that smoothing can be useful even for the unlexicalized DMV and EVG models. We will look at the linear interpolation technique, and talk about estimation using both Expectation Maximization and Variational Bayes. Linear Interpolation will be the basis for the exploration of techniques using several varieties of Bayesian Prior, as well as using estimation from held-out data. We will see that Bayesian estimation using linear interpolation will be the most effective of these techniques.

3.2 Estimation from Fully Observed Events

Suppose we are interested in smoothing the discrete probability distribution over of a event E with J possibilities conditioned on some conditioning events $C = C_1 \dots C_k$. This corresponds to selecting E given $C = c$ from a categorical distribution with parameter θ_c . In the supervised setting we have a series of observations of $(E_1, C_1) \dots (E_n, C_n)$. The maximum likelihood estimate (MLE) then assigns a value to θ_{ce} :

$$\hat{P}(E = e \mid C = c) = \hat{\theta}_{c,e} = \frac{f_{c,e}}{\sum_{e'} f_{c,e'}}$$

where $f_{e,c} = \sum_{i=1}^n \delta_{E_i=e} \delta_{C_i=c}$ is the number of times e, c was observed. One thing to note is that if c has never been seen, or seen only rarely, then this estimate will be very poor. This will particularly be the case when distributions condition on words.

However, perhaps by making an independence assumption about C , we can get an estimate that is better. For example, suppose we are estimating a language model. One option is a bigram model, in which each word is generated conditioned on the probability of the previous word. Hence, the probability of a sequence of words W_1, \dots, W_n is

$$P(W_1, \dots, W_n) = P(W_1) \prod_{i=2}^n P(W_i \mid W_{i-1})$$

If we estimate this model from a corpus in which some word w in the vocabulary is unobserved, the maximum likelihood estimate will set $\hat{P}(W_i = w \mid W_{i-1} = w') = 0$. If however w occurs in the test set, this will result in assigning the test set a probability estimate of zero. To avoid this, we would like to modify the estimate to place probability mass on unseen events at the expense of observed ones.

One option is to add some amount α to the count each possible word pair (w, w') ,

known as additive or Lidstone smoothing . This results in an estimate:

$$P_{\text{add}}(W_i = w \mid W_{i-1} = w') = \frac{f_{w,w'} + \alpha}{\sum_{v,w'} f_{v,w'} + \alpha}$$

Now if w is unobserved in the test set, it will have positive probability $\propto \alpha$. Now, an additional problem crops up when have to estimate conditional distributions. If in our test set we observe the sequence of words *The cassowary ate fruit* we would need to know an estimate of the probability of $P(w_i = \text{ate} \mid w_{i-1} = \text{cassowary})$. Perhaps *cassowary* is a sufficiently rare word that we never saw it in the training set. If we use additive smoothing, we would have, for example, that:

$$P_{\text{add}}(w_i = \text{ate} \mid w_{i-1} = \text{cassowary}) = P_{\text{add}}(w_i = \text{gormandized} \mid w_{i-1} = \text{cassowary})$$

However, under most corpora *ate* would be seen far more frequently than *gormandized*. The only reason the two are equal in this case is because we have never seen *cassowary*.

What we would really like is an estimate that allows $P(w_i \mid w_{i-1})$ to share some information across conditioning events w_{i-1} , while allowing the conditional distributions to still differ. One way to do this is through linear interpolation (Jelinek, 1997).

$$P_{LI}(w_i \mid w_{i-1}) = \lambda_1 \hat{P}(w_i \mid w_{i-1}) + \lambda_2 \hat{P}(w_i)$$

where $\lambda_1 + \lambda_2 = 1$ and \hat{P} indicates the maximum likelihood estimate. One important question is the setting of the parameters λ_1, λ_2 . Setting them via maximum likelihood would result in $\lambda_1 = 1, \lambda_2 = 0$, resulting in $P_{LI}(w_i \mid w_{i-1}) = \hat{P}(w_i \mid w_{i-1})$. Instead, they are typically set via EM by maximizing the likelihood of some heldout data.

These techniques are not mutually exclusive. For instance, we could linearly interpolate a bigram and unigram that were each additively smoothed.

For instance, the maximum likelihood estimate for $P(A = \text{DT} \mid d = \text{left}, H = \text{NN})$ would be:

$$\hat{P}(A = \text{DT} \mid d = \text{left}, H = \text{NN}) = \hat{\theta}_{L_{1,\text{NN}} \rightarrow Y_A L'_{\text{NN}}} = \frac{f(\mathbf{t}, L_{1,\text{NN}} \rightarrow Y_A L'_{\text{NN}})}{f(\mathbf{t}, L_{1,\text{NN}})}$$

3.3 Unsupervised Smoothing for PCFGs

In our dependency grammar induction problem we will be interested in smoothing PCFG rules. Each rule will correspond to some event E in the model, conditioned on some conditioning events $C = C_1 \dots C_K$ in the dependency model. As is usual in a PCFG these conditioning events are encoded in the nonterminal left-hand side of the rule, and the right-hand side represents some event in the model. For instance, in DMV, $L_{1,\text{NN}}$ encodes two variables: the head part-of-speech $H = \text{NN}$, and the direction $d = \text{left}$.

Suppose we are interested in smoothing to estimate a distribution over $r \in R_N$ for some nonterminal N , which encodes some set of conditioning events $N_{(1)} \dots N_{(k)}$. We will suppose there is a set of nonterminals $\mathcal{B} \subseteq \mathcal{N}$ where $N \in \mathcal{B}$, with \mathcal{B} called a *backoff set*, such that there exists a $j \in \{1, \dots, k\}$ such that for all $\hat{N} \in \mathcal{B}$, $\hat{N}_{(j)} = N_{(j)}$. That is, \mathcal{B} is a set of nonterminals with some conditioning event in common with N . We will smooth the distribution over $r \in R_N$ by combining a distribution conditioned on N with a distribution conditioned only on \mathcal{B} (called the *backoff distribution*). For instance, for the nonterminal $L_{1,\text{NN}}$ in DMV, which rewrites to $Y_A L'_{\text{NN}}$ for some part-of-speech A , the backoff set might group together $L_{1,\text{NN}}$ for all part-of-speech H . The backoff distribution would then give a common distribution over the left arguments, regardless of the head parts-of-speech.

To use linear interpolation to estimate the distribution of some event E conditioned on some set of context information $C = C_1 \dots C_K$ by smoothing it with distributions conditioned on a portion of the conditioning information $C' \subset C$, we would estimate

$P(E|C)$ as a weighted average of two distributions:

$$P(E | C) = \lambda_1 P_1(E|C) + \lambda_2 P_2 P(E|C')$$

where the distribution P_2 makes an independence assumption by dropping some of the conditioning information. In this section we will discuss formulating this variety of smoothing for PCFGs using the Tied-PCFG framework discussed in Chapter 2, as well as several techniques for estimating the λ interpolation parameters.

In a PCFG for each nonterminal N there is an associated distribution over rules in R_N . Nonterminal N encodes the information that the distribution over R_N is conditioned upon. For example, in DMV the nonterminal L_{NN}^1 encodes three separate pieces of conditioning information: the direction $d = left$ and the head part-of-speech $H = NN$. Likewise, a rule $r \in R_N$ encodes an event, the probability of which is conditioned on N . For instance in DMV $L_{NN}^1 \rightarrow Y_{JJ} L'_{NN}$ represents the generation of JJ as a left dependent of NN , and so the probability of rewriting $L_{NN}^1 \rightarrow Y_{JJ} L'_{NN}$ will be $P(A = JJ | H = NN, d = L)$.

Suppose in DMV we are interested in smoothing $P(A | H, d)$ with a component that excludes the head conditioning event. Using linear interpolation, this would be:

$$P(A | d, H) = \lambda_1 P_1(A | d, H) + \lambda_2 P_2(A | d)$$

We will estimate PCFG rules with linearly interpolated probabilities by creating a tied PCFG which is extended by adding rules that select between the main distribution P_1 and the backoff distribution P_2 , and also rules that correspond to selecting an argument from those distributions. In this example, we will start by replacing for each $A \in V_\tau$ the rule $L_{NN}^1 \rightarrow Y_A L'_{NN} \in R_{L_{NN}^1}$ with the following four rules:

$L_{1,NN} \rightarrow L_{1,NN}^1$	select distribution P_1 with probability λ_1
$L_{1,NN} \rightarrow L_{1,NN}^2$	select distribution P_2 with probability λ_2
$L_{1,NN}^1 \rightarrow Y_A L'_{NN}$	generate dependent A according to P_1
$L_{1,NN}^2 \rightarrow Y_A L'_{NN}$	generate dependent A according to P_2

Note that in order to have the conditioning structure we desire – namely that $P_2(A|d)$ is independent of the head H – the rule $L_{1,NN}^2 \rightarrow Y_A L'_{NN}$ needs to be tied together with other rules of the form $L_{1,H}^2 \rightarrow Y_A L'_H$, for all parts-of-speech $H \in V_\tau$.

More formally, given a CFG $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{R})$, let $\mathcal{A} \subseteq \mathcal{N}$ be the set of nonterminals whose rules we want to smooth. We will define an extended CFG $G' = (\mathcal{N}', \mathcal{T}, S, \mathcal{R}')$ in the following way. The set of terminal symbols \mathcal{T} and the start symbol S remain the same. Let the set of nonterminals $\mathcal{N}' = \mathcal{N} \cup \{A^i \mid i \in \{1, 2\}, \exists A \in \mathcal{A}\}$. That is, we have each nonterminal in \mathcal{N} , and additionally nonterminals A^1, A^2 for each nonterminal A whose distribution over rules is to be smoothed. Let the set of rules \mathcal{R}' contain all the unsmoothed rules of \mathcal{R}_A , as well as rules $A \rightarrow A^i$ and $A^i \rightarrow \beta$ when $A \rightarrow \beta$ is a smoothed rule:

$$\begin{aligned} \mathcal{R}' = & \quad \{\mathcal{R}_A \mid A \in \mathcal{N} \setminus \mathcal{A}\} \\ & \cup \{A \rightarrow A^i \mid i \in \{1, 2\}, A \in \mathcal{A}\} \\ & \cup \{A^i \rightarrow \beta \mid i \in \{1, 2\}, A \in \mathcal{A}, A \rightarrow \beta \in \mathcal{R}_A\} \end{aligned}$$

Having defined G' , we now set about tying the A^2 nonterminals. We will define a tied PCFG $\mathcal{H}' = \left(G', (\theta, \lambda), \underline{\underline{G'_{R'}}}\right)$, where λ is the vector of probabilities over interpolation rules (e.g. $A \rightarrow A^i$) and θ is the vector of probabilities for all other rules. Let $\bar{\mathcal{A}}$ be a partition of \mathcal{A} where each partition $\bar{\mathcal{A}}_j$ is composed of nonterminals whose rules we want

to smooth together. This will require that for each partition $\bar{\mathcal{A}}_j \in \bar{\mathcal{A}}, \forall A, A' \in \bar{\mathcal{A}}_j |R_A| = |R_{A'}|$. We will define set $\mathcal{N}^2 = \{\{A^2 \mid A \in \bar{\mathcal{A}}_j\} \mid \bar{\mathcal{A}}_j \in \bar{\mathcal{A}}\}$ of sets of nonterminals to tie. For each set of nonterminals $\mathcal{A}^2 \in \mathcal{N}^2$, we define the tied-PCFG equivalence relation $\stackrel{G'}{\equiv}_{R'}$ over the nonterminals in \mathcal{A}^2 . Rules that are analogous according to the model should be made equivalent. Let \bar{R}' denote the set of rule equivalence classes defined by this relation. We must finally define parameters θ_r for each rule equivalence class $r \in \bar{R}'$. These together define a tied-PCFG \mathcal{H}' .

For example, in EVG to smooth $P(A = DT \mid d = left, H = NN, v = 0)$ with $P_2(A = DT \mid d = left, v = 0)$ we define the backoff set to be $\{L_H^1 \mid H \in V_\tau\}$. In the extended grammar we define the tying relation to form rule equivalence classes by the argument they generate, i.e. for each argument $A \in V_\tau$, we have a rule equivalence class $\{L_H^{1b_2} \rightarrow Y_A H_L \mid H \in V_\tau\}$.

We can see that in grammar G' each $N \in \mathcal{B}$ eventually ends up rewriting to one of N 's expansions β in G . There are two indirect paths, one through N^{b_1} and one through N^{b_2} . Thus if we define a PCFG (G, ϕ) , we would assign the probability of $N \rightarrow \beta$ in G , $\phi_{N \rightarrow \beta}$, as the probability of rewriting N as β in G' via N^{b_1} and N^{b_2} . That is:

$$\phi_{N \rightarrow \beta} = \lambda_{N \rightarrow N^{b_1}} \theta_{N^{b_1} \rightarrow \beta} + \lambda_{N \rightarrow N^{b_2}} \theta_{N^{b_2} \rightarrow \beta}$$

The example in Figure 3.1 shows the probability that L_{dog}^1 rewrites to $Y_{big} dog_L$ in grammar G .

3.4 Estimation from Held-out data

If we use linear interpolation smoothing we need to have some method of estimating the mixing weights. In supervised models, the mixing weights are typically set to maximize

$$P_G \left(\begin{array}{c} L_{dog}^1 \\ \swarrow \quad \searrow \\ Y_{big} \quad dog_L \end{array} \right) = P_{G'} \left(\begin{array}{c} L_{dog}^1 \\ | \\ L_{dog}^{1b1} \\ \swarrow \quad \searrow \\ Y_{big} \quad dog_L \end{array} \right) + P_{G'} \left(\begin{array}{c} L_{dog}^1 \\ | \\ L_{dog}^{1b2} \\ \swarrow \quad \searrow \\ Y_{big} \quad dog_L \end{array} \right)$$

Figure 3.1: Using linear interpolation to smooth $L_{dog}^1 \rightarrow Y_{big} dog_L$: The first component represents the distribution fully conditioned on head dog , while the second component represents the distribution ignoring the head conditioning event. This later is accomplished by tying the rule $L_{dog}^{1b2} \rightarrow Y_{big} dog_L$ to, for instance, $L_{cat}^{1b2} \rightarrow Y_{big} cat_L$, $L_{fish}^{1b2} \rightarrow Y_{big} fish_L$ etc.

the likelihood of a held-out data set. The held-out set gives an idea of how often in general events will occur that were unseen or rare in the training data. Furthermore, the weights are typically bucketed into c equivalence classes by some measure of the frequency of the conditioning information. In general we would expect more infrequent conditioning events to require more smoothing. For instance applying the method described by

Chen (1996) we would group conditioning information \bar{A} by the per-nonzero-event frequency of rules for that conditioning information, i.e. by:

$$\sum_{r \in \bar{R}_{\bar{A}}} \frac{f(\mathbf{t}, r)}{|\{r \in \bar{R}_{\bar{A}} : f(\mathbf{t}, r) > 0\}|}$$

This is done by dividing the range of such values into partitions, and assigning each \bar{A} to it's corresponding partition. One challenge of using a similar approach in the case where \mathbf{t} is a hidden variable is that $f(\mathbf{t}, r)$ will not be observed, and hence the bucketing of \bar{A} will not be observed.

One solution is to only use whatever observed data is available for a given distribution. For instance, if we are conditioning on parts-of-speech, then those are fully observed, even if the valence is not. Thus we bucket One extreme variant of this is to simply have a single bucket. Another option is simply not use bucketing, and have a parameter for each conditioning event. The mixing parameters can then be set to maximize the likelihood of

Bucketing	Directed Acc		Undirected Acc.	
	Train	Dev	Train	Dev
No interpolation	0.546 $\begin{pmatrix} +0.037 \\ -0.064 \end{pmatrix}$	0.512 $\begin{pmatrix} +0.038 \\ -0.062 \end{pmatrix}$	0.675 $\begin{pmatrix} +0.015 \\ -0.025 \end{pmatrix}$	0.652 $\begin{pmatrix} +0.016 \\ -0.027 \end{pmatrix}$
1 Bucket	0.562 $\begin{pmatrix} +0.039 \\ -0.068 \end{pmatrix}$	0.527 $\begin{pmatrix} +0.038 \\ -0.058 \end{pmatrix}$	0.679 $\begin{pmatrix} +0.013 \\ -0.029 \end{pmatrix}$	0.656 $\begin{pmatrix} +0.014 \\ -0.026 \end{pmatrix}$
8 Buckets	0.543 $\begin{pmatrix} +0.032 \\ -0.064 \end{pmatrix}$	0.506 $\begin{pmatrix} +0.025 \\ -0.062 \end{pmatrix}$	0.673 $\begin{pmatrix} +0.013 \\ -0.026 \end{pmatrix}$	0.649 $\begin{pmatrix} +0.010 \\ -0.027 \end{pmatrix}$
Individual	0.584 $\begin{pmatrix} +0.059 \\ -0.038 \end{pmatrix}$	0.549 $\begin{pmatrix} +0.057 \\ -0.038 \end{pmatrix}$	0.692 $\begin{pmatrix} +0.025 \\ -0.014 \end{pmatrix}$	0.671 $\begin{pmatrix} +0.026 \\ -0.015 \end{pmatrix}$

Table 3.1: Results smoothed DMV using held-out estimation for Linear Interpolation using EM. All results use randomized initialization

the held-out set using EM, as in the supervised case.

We estimate by alternating an iteration of EM to estimate θ on the training set leaving λ fixed, and an iteration of EM to estimate λ on the held-out set, leaving θ fixed.

$$\theta^* = \operatorname{argmax}_{\theta} \log p(\mathbf{s}|\theta, \lambda^*)$$

$$\lambda^* = \operatorname{argmax}_{\lambda} \log p(\mathbf{s}'|\theta^*, \lambda)$$

While this clearly does not globally maximize the likelihoods of either the training or development sets, if we commit to finishing on either side, we are guaranteed a local maximum in that likelihood.

The results are shown in table 3.1. We can see that having a bucket for each POS outperforms having 1 bucket for all parts of speech or dividing the set of POS into 8 buckets, based on frequency.

3.5 Bayesian Estimation with Linear Interpolation Smoothing

An alternative to the heldout estimation is to use Bayesian priors which incorporate our prior beliefs about what the mixture parameters are doing. When smoothing, we should expect first of all that there should be an a prior bias towards the backoff distribution, since its estimate, requiring fewer examples, will presumably be better. We can then perform estimation using the Variational Bayes technique (Kurihara and Sato, 2004).

We place Dirichlet priors on both the interpolation mixing parameters λ and component distribution parameters θ . In this approach we will use Beta priors on the λ s and uninformative symmetric Dirichlet priors on θ . This gives an overall model which is:

$$\begin{aligned} \forall N \in \mathcal{N}_\lambda \quad \lambda_N | \beta_1, \beta_2 &\sim \text{Beta}(\beta_1, \beta_2) \\ \forall N \in \mathcal{N}_\theta \quad \theta_N | \alpha_N &\sim \text{Dir}(\alpha_N) \\ \mathbf{t} &\sim \text{PCFG}(G', \theta, \lambda) \end{aligned}$$

where \mathcal{N}_λ is the subset of \mathcal{N}' whose rules correspond to interpolation parameters, and $\mathcal{N}_\theta = \mathcal{N}' \setminus \mathcal{N}_\lambda$. Since the priors on θ are uninformative $\alpha_N = 1 \forall N \in \mathcal{N}_\theta$.

Since the Beta is the two-dimensional analogue to the Dirichlet, we can estimate this using the Variational Bayes technique for tied-PCFGs with Dirichlet priors which we discussed above.

One thing to notice is we can think of linear interpolation in the Bayesian framework as placing a mixture of Dirichlets prior on the rule probabilities of G .

As we will see the choice of hyperparameters β_1, β_2 is important, as their setting reflects the prior information we wish to describe. Our initial strategy was inspired by an

early technique we tried called collapsed interpolation, which we discuss in Appendix D. This it turned out performed similarly to linear interpolation. Collapsed interpolation incorporates the prior knowledge that conditioning events that have been seen fewer times should be more strongly smoothed, and that the model will only start to ignore the backoff distribution after having seen a sufficiently large number of training examples. In collapsed interpolation, for each nonterminal N we wish to smooth, we add a new rule $N \rightarrow N^b$ which represents choosing to back off. Thus the smoothed probability of a rule $N \rightarrow \beta$ becomes: $P(N \rightarrow \beta) = P_1(N \rightarrow \beta) + P_1(N \rightarrow N^b)P_2(N^b \rightarrow \beta)$.

In those experiments we set the hyperparameter corresponding $N \rightarrow N^b$ to 2 times the number of other rules in N . We can interpret the linear interpolation setting as factoring $P_1(N \rightarrow \beta)$ into two decisions: one of which decides not to use the backoff distribution, and the other of which decides which rule to use, given the decision not to use the backoff distribution.

Suppose we take the Dirichlet distribution over θ_r associated with N , and examine the marginal probability of $N \rightarrow N^b$. We would have that $\theta_{N \rightarrow N^b} \sim \text{Beta}(2K, K)$.

We can accomplish a similar hyperparameter setting in the linear interpolation case by setting the Dirichlet hyperparameters for each $N \rightarrow N^{b_1}, N \rightarrow N^{b_2}$ decision to $(K, 2K)$ respectively, where $K = |\mathcal{R}_{N^{b_1}}|$ is the number of rewrite rules for N in G .¹

Results using this technique for DMV and EVG are given in Table 3.2.

One advantage to the Linear Interpolation approach, of course, is we have a bit more flexibility about setting the hyperparameters. Table 3.3 shows the results of leaving the 2:1 ratio constant between backing off and not, while varying its strength. We can see for both DMV and EVG, having reasonably strong preference is important.

One concern is whether for a strong prior, the model can really be said to be learning

¹We set the other Dirichlet hyperparameters to 1.

Model	Smoothing	Directed Acc		Undirected Acc.	
		Train	Dev	Train	Dev
DMV	None	0.583 $\begin{pmatrix} +0.064 \\ -0.034 \end{pmatrix}$	0.549 $\begin{pmatrix} +0.065 \\ -0.031 \end{pmatrix}$	0.689 $\begin{pmatrix} +0.025 \\ -0.014 \end{pmatrix}$	0.668 $\begin{pmatrix} +0.027 \\ -0.013 \end{pmatrix}$
	Lin. Interp.	0.625 $\begin{pmatrix} +0.018 \\ -0.015 \end{pmatrix}$	0.593 $\begin{pmatrix} +0.018 \\ -0.014 \end{pmatrix}$	0.706 $\begin{pmatrix} +0.009 \\ -0.006 \end{pmatrix}$	0.683 $\begin{pmatrix} +0.009 \\ -0.007 \end{pmatrix}$
EVG	None	0.526 $\begin{pmatrix} +0.017 \\ -0.080 \end{pmatrix}$	0.500 $\begin{pmatrix} +0.013 \\ -0.080 \end{pmatrix}$	0.679 $\begin{pmatrix} +0.008 \\ -0.028 \end{pmatrix}$	0.657 $\begin{pmatrix} +0.006 \\ -0.032 \end{pmatrix}$
	Lin. Interp.-Fixed $q(\lambda)$	0.617 $\begin{pmatrix} +0.137 \\ -0.061 \end{pmatrix}$	0.590 $\begin{pmatrix} +0.124 \\ -0.057 \end{pmatrix}$	0.716 $\begin{pmatrix} +0.064 \\ -0.028 \end{pmatrix}$	0.696 $\begin{pmatrix} +0.055 \\ -0.026 \end{pmatrix}$
	Lin. Interp.	0.658 $\begin{pmatrix} +0.038 \\ -0.021 \end{pmatrix}$	0.629 $\begin{pmatrix} +0.036 \\ -0.018 \end{pmatrix}$	0.734 $\begin{pmatrix} +0.018 \\ -0.009 \end{pmatrix}$	0.712 $\begin{pmatrix} +0.018 \\ -0.007 \end{pmatrix}$

Table 3.2: Results for DMV, EVG smoothed with Linear Interpolation trained using Variational Bayes.

the estimates. To explore this, we ran an experiment where we do not learn the mixing parameters. Each $q(\lambda_N)$ is fixed to its prior, and we reestimate $q(\mathbf{t})$ and $q(\theta)$ using Variational Bayes. The result is shown in Table 3.3. As can be seen, there is a clear benefit to learning the $q(\lambda)$ s, improving the development set directed accuracy from 0.590 to 0.629.

3.5.1 Priors Preferring Backoff distribution

Another option is to use a beta prior on the mixing parameters which encodes the intuition that we should prefer the backoff distribution to the more fully conditioned distribution.

The way we do this is to use a beta prior on λ_1, λ_2 with hyperparameters $\beta, 1$, for $0 < \beta < 1$. A visualization of this prior is given in Figure 3.2. Note that a smaller β results in a Beta more concentrated around the distribution that puts all its mass on selecting to backoff. By setting the second hyperparameter to 1, and noting that $\Gamma(\beta + 1) = \beta\Gamma(\beta)$ and $\Gamma(1) = 1$, the density function reduces to:

Model	β_1	β_2	Directed Acc		Undirected Acc.	
			Train	Dev	Train	Dev
DMV	5	10	0.585 (+0.052 -0.036)	0.551 (+0.051 -0.032)	0.690 (+0.021 -0.014)	0.668 (+0.023 -0.012)
	10	20	0.574 (+0.015 -0.057)	0.540 (+0.013 -0.057)	0.684 (+0.006 -0.023)	0.661 (+0.006 -0.023)
	20	40	0.634 (+0.045 -0.006)	0.603 (+0.045 -0.003)	0.709 (+0.019 -0.004)	0.687 (+0.020 -0.003)
	35	70	0.625 (+0.018 -0.015)	0.593 (+0.018 -0.014)	0.706 (+0.009 -0.006)	0.683 (+0.009 -0.007)
	60	120	0.622 (+0.026 -0.008)	0.592 (+0.025 -0.007)	0.703 (+0.012 -0.004)	0.683 (+0.012 -0.003)
EVG	5	10	0.555 (+0.053 -0.053)	0.530 (+0.048 -0.052)	0.693 (+0.021 -0.020)	0.671 (+0.020 -0.021)
	10	20	0.657 (+0.072 -0.023)	0.624 (+0.066 -0.022)	0.731 (+0.028 -0.011)	0.708 (+0.028 -0.010)
	20	40	0.632 (+0.031 -0.051)	0.608 (+0.033 -0.042)	0.723 (+0.013 -0.021)	0.703 (+0.015 -0.017)
	35	70	0.658 (+0.038 -0.021)	0.629 (+0.036 -0.018)	0.734 (+0.018 -0.009)	0.712 (+0.018 -0.007)
	60	120	0.657 (+0.023 -0.024)	0.623 (+0.018 -0.020)	0.732 (+0.010 -0.012)	0.708 (+0.008 -0.009)

Table 3.3: DMV and EVG smoothed using Linear Interpolation, varying strength of interpolation hyperparameters β_1, β_2

Model	β_1	β_2	Directed Acc		Undirected Acc.	
			Train	Dev	Train	Dev
DMV	35	100	0.605 (+0.006 -0.021)	0.576 (+0.006 -0.020)	0.693 (+0.002 -0.012)	0.671 (+0.002 -0.013)
	35	70	0.625 (+0.018 -0.015)	0.593 (+0.018 -0.014)	0.706 (+0.009 -0.006)	0.683 (+0.009 -0.007)
	35	35	0.633 (+0.049 -0.007)	0.595 (+0.046 -0.008)	0.708 (+0.020 -0.004)	0.684 (+0.020 -0.004)
	70	35	0.642 (+0.048 -0.003)	0.599 (+0.045 -0.005)	0.712 (+0.019 -0.002)	0.687 (+0.019 -0.004)
	100	35	0.623 (+0.062 -0.011)	0.581 (+0.056 -0.011)	0.703 (+0.024 -0.006)	0.678 (+0.022 -0.006)
EVG	35	100	0.686 (+0.030 -0.014)	0.652 (+0.024 -0.013)	0.748 (+0.015 -0.006)	0.722 (+0.011 -0.005)
	35	70	0.658 (+0.038 -0.021)	0.629 (+0.036 -0.018)	0.734 (+0.018 -0.009)	0.712 (+0.018 -0.007)
	35	35	0.658 (+0.059 -0.014)	0.623 (+0.053 -0.014)	0.728 (+0.022 -0.009)	0.705 (+0.022 -0.008)
	70	35	0.611 (+0.057 -0.053)	0.572 (+0.050 -0.054)	0.705 (+0.024 -0.026)	0.679 (+0.022 -0.027)
	100	35	0.607 (+0.057 -0.044)	0.568 (+0.053 -0.043)	0.703 (+0.024 -0.019)	0.677 (+0.024 -0.021)

Table 3.4: DMV and EVG smoothed using Linear Interpolation, varying relative strengths of β_1, β_2

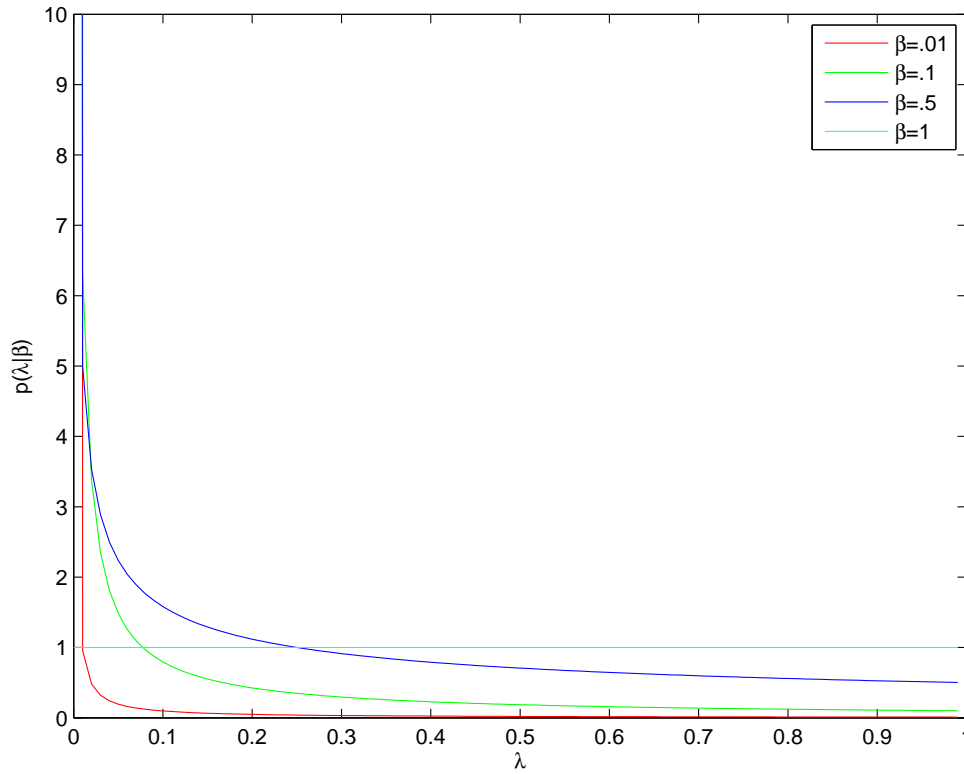


Figure 3.2: Plot of $p(\lambda_1|\beta)$ vs. λ_1 for various β hyperparameters.

$$\begin{aligned}
 P(\lambda_1|\beta, 1) &= \frac{\Gamma(\beta + 1)}{\Gamma(\beta)\Gamma(1)} \lambda_1^{\beta-1} (1 - \lambda_1)^0 \\
 &= \beta \lambda_1^{\beta-1}
 \end{aligned}$$

One important question is how to set β . A smaller β implies a stronger bias towards the backoff distribution.

Table 3.5 shows the result of varying β with a prefer-backoff prior for the smoothed DMV model. As we can see the resulting model varies widely with the choice of β . For $\beta = .1$ the result is encouraging, with directed accuracy almost as high as with the previous

DMV-Smoothed	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
β				
.1	0.612 (+0.061 -0.022)	0.582 (+0.060 -0.019)	0.701 (+0.024 -0.008)	0.679 (+0.025 -0.008)
.2	0.592 (+0.083 -0.014)	0.565 (+0.081 -0.016)	0.694 (+0.046 -0.002)	0.674 (+0.046 -0.004)
.3	0.582 (+0.091 -0.018)	0.553 (+0.085 -0.019)	0.687 (+0.044 -0.006)	0.669 (+0.044 -0.008)
.4	0.573 (+0.076 -0.023)	0.544 (+0.073 -0.021)	0.686 (+0.031 -0.008)	0.667 (+0.031 -0.007)
.5	0.530 (+0.033 -0.061)	0.498 (+0.031 -0.059)	0.660 (+0.010 -0.031)	0.636 (+0.009 -0.032)
.6	0.522 (+0.006 -0.093)	0.493 (+0.007 -0.088)	0.667 (+0.002 -0.037)	0.645 (+0.003 -0.037)
.7	0.572 (+0.049 -0.040)	0.541 (+0.048 -0.040)	0.687 (+0.020 -0.016)	0.664 (+0.020 -0.017)
.8	0.565 (+0.065 -0.059)	0.538 (+0.060 -0.058)	0.684 (+0.028 -0.022)	0.665 (+0.027 -0.024)
.9	0.526 (+0.031 -0.065)	0.494 (+0.028 -0.065)	0.669 (+0.013 -0.024)	0.647 (+0.013 -0.025)

Table 3.5: Smoothed DMV with prefer-backoff prior 1 on λ s

prior.

Another Prior preferring the Backoff Distribution:

We can make a similarly shaped prior favoring the backoff distribution by replacing the $\text{Beta}(\beta, 1)$ of the model in the previous section with a $\text{Beta}(1, \beta)$ distribution, for $\beta > 1$ (shown in Figure 3.3). That is, the hyperparameter associated with the backoff distribution here is β , and the hyperparameter associated with the fully conditioned distribution is 1, the reverse of the case in the previous model. However, since $\beta > 1$ it has a similar effect on shape of the prior. Note that while both versions of the prior place most of the mass on smaller λ 's, prior 2 places more on small λ s such as $\lambda < 0.2$; in contrast prior 1 places most of the mass right near $\lambda = 0$, but does not strongly prefer $\lambda = 0.1$ to $\lambda = 0.2$. The results of estimating this model with Variational Bayes are shown in Table 3.6, for various β .

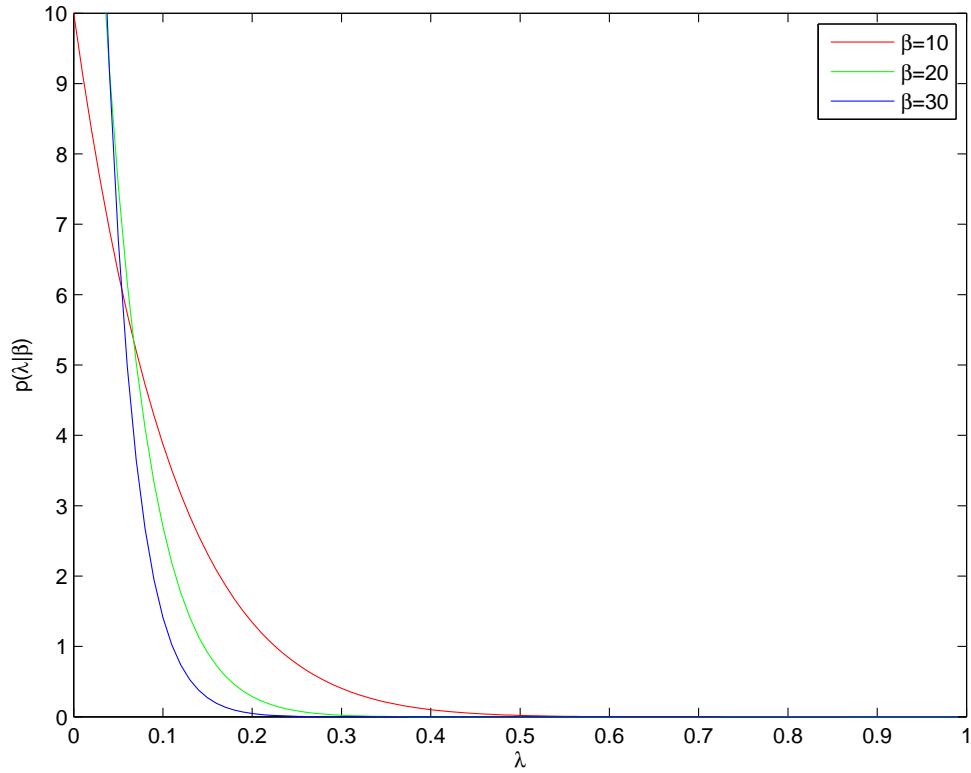


Figure 3.3: Beta Prior favoring Backoff distribution, version 2

	β	Directed Accuracy		Undirected Accuracy	
		Train	Dev.	Train	Dev.
DMV-Smoothed	30	0.629 (+0.046 -0.006)	0.594 (+0.044 -0.004)	0.706 (+0.022 -0.003)	0.682 (+0.022 -0.001)
DMV-Smoothed	20	0.608 (+0.059 -0.010)	0.573 (+0.054 -0.016)	0.698 (+0.023 -0.004)	0.674 (+0.021 -0.007)
DMV-Smoothed	10	0.558 (+0.071 -0.032)	0.524 (+0.065 -0.034)	0.678 (+0.028 -0.012)	0.654 (+0.025 -0.015)
EVG-Smoothed	30	0.623 (+0.058 -0.041)	0.595 (+0.052 -0.040)	0.720 (+0.025 -0.017)	0.699 (+0.023 -0.019)
EVG-Smoothed	20	0.630 (+0.049 -0.036)	0.597 (+0.044 -0.032)	0.720 (+0.023 -0.017)	0.696 (+0.021 -0.013)
EVG-Smoothed	10	0.621 (+0.056 -0.052)	0.593 (+0.054 -0.048)	0.718 (+0.024 -0.022)	0.697 (+0.024 -0.020)

Table 3.6: Smoothed DMV and EVG with prefer-backoff prior 2 on λ s

3.6 Analysis

In order to see where the DMV improves with linear interpolation, we examine the difference in correct dependencies over 10 runs for DMV with linear interpolation less DMV without. These are presented for Sections 2-21 broken down in various ways in Table 3.7 (by child part-of-speech and direction), Table 3.8 (by head part-of-speech and direction), and Tables 3.9 and 3.10 (by head/child part-of-speech pair).

Looking at Table 3.7 we can see that the major improvements come from DTs that are left dependents, RBs that are right dependents, and noun dependents in both directions. This comes at the expense of many fewer correct VB and VBN right dependents and JJ left dependents. From Table 3.8 we can see improvements in the left dependents of NN, NNP, VBZ and VBD, as well as right dependents of IN; left dependents of JJs fair much worse.

Breaking this down further, from Table 3.9 of the 5301 net new correct dependencies with a DT as the child, these almost exclusively come from NN, and NNS, with a handful of NNP. Of net new correct dependencies with RB as the child, the most prominent are VBZ, VBD, VBP, and MD; these are offset by a large number of net incorrect RB dependencies whose head is JJ. Also to be noted are 2274 net correct NNP→NNP dependencies. A large net decrease in correct dependencies are MD→VB.

DMV vs. EVG

We analyze the difference between DMV with linear interpolation and EVG with linear interpolation by again looking at the difference in correct edges in the training set. These are presented for Sections 2-21 in Table 3.11 (by child part-of-speech and direction), Table 3.12 (by head part-of-speech and direction), and Tables 3.13 and 3.14 (by head/child part-of-speech pair).

Most of the major improvements to adding valence in the argument distribution accrue

Child POS	Correct DMV	Correct DMV_LI	Difference (DMV_LI-DMV)	Difference Left	Difference Right
DT	24405	29706	5301	5388	-87
RB	6667	11472	4805	-1541	6346
NNP	24856	28693	3837	3077	760
VB	7104	4881	-2223	-16	-2207
JJ	15385	13205	-2180	-1231	-949
VBN	5960	4283	-1677	-64	-1613
NN	39688	41326	1638	867	771
NNS	21306	22409	1103	713	390
CD	9712	8984	-728	-619	-109
IN	10821	10094	-727	-141	-586
VBG	4207	3661	-546	-23	-523
RP	630	1011	381	0	381
CC	964	1299	335	318	17
VBZ	14758	15065	307	-51	358
POS	1611	1879	268	261	7
PRP\$	2095	2362	267	267	0
VBP	8051	8250	199	29	170
JJR	908	738	-170	-59	-111
TO	2598	2732	134	-1	135
PRP	14946	14858	-88	-210	122
NNPS	1139	1055	-84	-63	-21
WP	576	493	-83	-68	-15
JJS	265	225	-40	-51	11
\$	635	600	-35	146	-181
PDT	84	51	-33	-28	-5
WRB	465	434	-31	-18	-13
EX	843	821	-22	-24	2
MD	3941	3955	14	3	11
WDT	163	150	-13	10	-23
VBD	15440	15451	11	14	-3
RBS	109	98	-11	-17	6
UH	170	160	-10	-24	14
LS	24	18	-6	-6	0
RBR	519	521	2	18	-16
FW	89	91	2	4	-2
SYM	145	146	1	-9	10

Table 3.7: Change in correct dependencies by child part-of-speech across 10 runs for DMV with and without Linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.

Head POS	Correct DMV	Correct DMV_LI	Difference (DMV_LI-DMV)	Difference Left	Difference Right
NN	33387	36428	3041	3309	-268
NNP	10056	12312	2256	2482	-226
VBZ	26550	27980	1430	1285	145
JJ	2414	986	-1428	-1425	-3
IN	21358	22557	1199	-63	1262
VBD	27195	28353	1158	1139	19
ROOT	45985	46937	952	0	952
VBP	17108	17930	822	656	166
NNS	20158	20962	804	533	271
RB	621	1401	780	-6	786
VBN	4072	3408	-664	-420	-244
MD	9592	10011	419	309	110
TO	2855	2557	-298	5	-303
JJR	375	94	-281	-282	1
POS	2312	2557	245	245	0
CD	2076	1840	-236	-373	137
VBG	3431	3273	-158	-160	2
WP	125	76	-49	-12	-37
NNPS	758	715	-43	-43	0
WRB	305	269	-36	0	-36
\$	990	958	-32	12	-44
VB	9071	9044	-27	-302	275
CC	120	145	25	0	25
SYM	69	88	19	-4	23
JJS	12	0	-12	-12	0
PRP	74	82	8	6	2
RP	3	10	7	0	7
FW	65	58	-7	-6	-1
DT	100	107	7	-10	17
UH	27	22	-5	-9	4
WDT	3	7	4	2	2
RBR	12	10	-2	-5	3
RBS	0	0	0	0	0
PRP\$	0	0	0	0	0
PDT	0	0	0	0	0
LS	0	0	0	0	0
EX	0	0	0	0	0

Table 3.8: Change in correct dependencies by head part-of-speech across 10 runs for DMV with and without Linear Interpolation smoothing. Left indicates child is a left dependent of its head in system output.

	NN	NNP	DT	NNS	JJ	IN	RB	VBZ	VBD	CD	PRP	VB	VBP	VBN	TO	CC	VBG	MD	PRP\$
NN	-156	-101	4306	-179	-950	0	-173	-1	-19	44	0	5	0	-37	-50	130	-50	0	199
ROOT	-7	363	0	-11	-1	-63	83	312	28	175	0	-88	181	-54	7	-10	-3	22	0
VBD	398	386	3	280	-242	-325	1468	-59	30	8	-13	-308	56	-640	173	8	-85	1	0
VBZ	258	423	-61	79	-497	-140	2161	6	25	28	-5	-244	3	-465	25	49	-149	5	0
NNS	-164	-91	980	-31	-48	241	54	0	0	-210	-3	0	0	-62	3	126	-21	0	56
IN	703	304	-48	299	4	-11	10	32	-1	42	1	-1	2	-5	0	2	-20	-10	0
VBP	35	40	-19	818	-317	-215	1427	7	-11	-4	24	-484	-4	-235	1	-43	-204	8	0
NNP	32	2274	169	0	-2	-57	-3	0	0	-147	0	3	1	4	0	4	0	0	4
VB	145	9	-14	44	-5	-89	-15	19	4	0	-49	-76	-5	-99	-3	2	-4	3	0
MD	102	80	-1	98	0	6	1108	11	0	2	56	-1040	-2	0	0	17	-10	0	0
VBN	-41	-1	-14	-76	-80	-183	-153	0	0	-6	-46	-1	0	-79	-15	1	-1	-1	0
JJ	42	-1	5	-69	-30	0	-1396	0	0	-20	-25	0	0	0	0	43	3	0	15
VBG	3	-8	-1	-36	-24	-21	-62	3	0	-5	5	6	0	-15	-20	2	-11	0	0
POS	83	193	-45	12	0	0	3	0	0	1	0	0	0	0	0	0	0	0	-2
TO	18	17	0	-146	-3	5	4	0	0	-132	-10	0	0	0	0	0	0	0	0
CD	-28	13	-9	26	-10	4	116	0	0	-506	0	0	0	-1	0	3	0	0	0
RB	140	0	48	19	52	88	384	0	0	54	0	7	0	9	12	-2	8	0	0
\$	28	0	0	0	-5	0	-11	0	0	-43	0	0	0	0	0	-1	0	0	0
NNPS	0	-62	25	0	-20	0	4	0	0	0	0	0	0	0	0	10	0	0	0
JJR	5	0	-14	-26	-7	-4	-206	0	0	-12	-7	0	0	0	0	-8	0	0	1
WP	0	0	0	0	0	7	-9	-3	-19	0	0	0	-19	0	0	-3	0	-3	0
DT	0	0	0	0	0	23	-14	-2	-1	3	-16	0	-4	0	0	0	1	0	0
WRB	26	0	0	0	1	0	4	-18	-25	0	0	-4	-10	0	0	0	-10	0	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0
CC	-6	1	0	0	0	0	23	0	0	0	0	2	0	0	0	0	0	-1	0
RBR	7	0	0	-10	1	1	-2	0	0	0	0	0	0	0	1	0	0	0	0
JJS	0	0	-8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-3
PRP	6	-6	3	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0
FW	-1	0	-4	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	-3
SYM	3	0	0	12	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.9: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for DMV with Linear Interpolation less DMV across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

	POS	JJR	\$	NNPS	RP	WP	RBR	EX	JJS	WRB	WDT	SYM	UH	PDT	RBS	FW	LS
NN	179	-39	-16	-4	-3	0	4	0	-25	0	-12	-2	-2	-7	0	0	0
ROOT	0	0	12	-24	0	0	0	0	0	16	0	10	6	0	0	-2	0
VBD	3	-48	-12	-2	72	-2	10	-7	0	-9	2	0	9	0	0	3	0
VBZ	4	-13	-7	-7	43	-75	2	-17	-3	-4	6	-1	0	0	0	-1	2
NNS	66	-29	0	-1	0	0	0	0	-21	1	-6	4	0	-40	0	0	0
IN	2	10	-101	5	0	-15	8	2	16	-26	0	-4	0	-1	0	0	0
VBP	0	-7	-9	14	22	-14	0	5	-2	-11	0	0	0	0	0	0	0
NNP	18	0	0	-54	0	12	0	0	0	0	-2	0	0	0	0	0	0
VB	-4	-42	-4	-4	150	9	0	0	0	7	-1	0	-7	0	6	0	-4
MD	0	0	0	1	0	0	0	-5	0	-6	2	0	0	0	0	0	0
VBN	0	1	-1	-5	53	4	-4	0	-5	0	0	-4	0	0	-3	0	-4
JJ	0	0	-1	-1	0	0	21	0	0	0	0	0	0	0	-14	0	0
VBG	0	1	-12	1	41	-6	-1	0	0	0	0	0	2	0	0	0	0
POS	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	2	0
TO	1	-11	-40	-2	0	3	0	0	0	0	0	0	-2	0	0	0	0
CD	-4	0	156	0	3	1	0	0	0	0	0	0	0	0	0	0	0
RB	4	8	0	0	0	-3	-38	0	0	0	0	0	-6	-4	0	0	0
\$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNPS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJR	0	-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DT	0	0	0	0	0	-1	0	0	0	0	-2	0	0	20	0	0	0
WRB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	2	0	0	0	4	0	0	0	1	0	0	-1	0	0	0	0
RBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJS	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
PRP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	0	0	0	0	0	0	0	0	0	-2	0	0	0	0	0
SYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	0	0	0	0	0	0	0	0	0	0	0	-9	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.10: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for DMV with Linear Interpolation less DMV across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

Child POS	Correct DMV	Correct EVG	Difference (EVG-DMV)	Difference Left	Difference Right
NNP	28693	22584	-6109	-5139	-970
IN	10094	15976	5882	1242	4640
DT	29706	26320	-3386	-3542	156
CC	1299	4200	2901	2756	145
NN	41326	44127	2801	687	2114
VBN	4283	6766	2483	28	2455
JJ	13205	15139	1934	983	951
VBZ	15065	16465	1400	309	1091
VBP	8250	9649	1399	427	972
MD	3955	5017	1062	96	966
NNS	22409	23361	952	13	939
RB	11472	12419	947	1220	-273
VBD	15451	16365	914	48	866
VB	4881	5683	802	76	726
NNPS	1055	534	-521	-326	-195
CD	8984	9435	451	134	317
VBG	3661	4022	361	58	303
POS	1879	1532	-347	-338	-9
PRP	14858	15139	281	-118	399
JJR	738	889	151	-43	194
JJS	225	368	143	91	52
WP	493	620	127	47	80
TO	2732	2824	92	-419	511
RP	1011	1102	91	0	91
WRB	434	509	75	35	40
PRP\$	2362	2290	-72	-72	0
UH	160	225	65	58	7
\$	600	556	-44	56	-100
EX	821	855	34	24	10
LS	18	51	33	33	0
RBR	521	547	26	-146	172
RBS	98	79	-19	-37	18
WDT	150	139	-11	-1	-10
SYM	146	151	5	5	0
FW	91	87	-4	-5	1
PDT	51	54	3	2	1

Table 3.11: Change in correct dependencies by child part-of-speech across 10 runs for EVG and DMV with linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.

Head POS	Correct DMV	Correct EVG	Difference (EVG-DMV)	Difference Left	Difference Right
VBD	28353	33569	5216	1989	3227
VBZ	27980	33131	5151	2147	3004
ROOT	46937	50173	3236	0	3236
NNP	12312	9131	-3181	-3636	455
VBP	17930	20776	2846	1531	1315
VBN	3408	5107	1699	-10	1709
IN	22557	21329	-1228	-325	-903
MD	10011	11075	1064	760	304
\$	958	404	-554	-11	-543
POS	2557	2028	-529	-529	0
NNPS	715	210	-505	-505	0
CD	1840	2321	481	386	95
TO	2557	2993	436	-19	455
NNS	20962	21288	326	-594	920
VB	9044	9336	292	-376	668
VBG	3273	3545	272	14	258
NN	36428	36212	-216	-2142	1926
WP	76	174	98	-5	103
JJR	94	37	-57	-69	12
JJS	0	53	53	10	43
JJ	986	938	-48	-226	178
RBR	10	54	44	-3	47
WRB	269	303	34	0	34
RB	1401	1369	-32	-94	62
PRP	82	62	-20	-21	1
UH	22	38	16	9	7
CC	145	131	-14	0	-14
RBS	0	11	11	2	9
SYM	88	92	4	0	4
DT	107	111	4	-35	39
FW	58	60	2	-1	3
WDT	7	8	1	-5	6
RP	10	10	0	0	0
PRP\$	0	0	0	0	0
PDT	0	0	0	0	0
LS	0	0	0	0	0
EX	0	0	0	0	0

Table 3.12: Change in correct dependencies by head part-of-speech across 10 runs for EVG and DMV with linear interpolation smoothing. Left indicates child is a left dependent of its head in system output.

	NN	NNP	DT	NNS	JJ	IN	RB	VBZ	VBD	CD	PRP	VB	VBP	VBN	TO	CC	VBG	MD	PRP\$
NN	139	-7	-2555	-31	915	1609	47	0	9	-125	-1	-4	0	85	64	-104	8	0	-43
ROOT	65	-554	0	100	9	3	29	1046	740	1	0	73	842	114	-3	3	17	853	0
VBD	871	-483	33	457	311	953	475	163	104	287	51	-49	147	582	196	775	107	74	0
VBZ	1538	-675	39	262	346	524	373	173	6	36	39	-6	259	804	33	993	136	48	0
NNS	64	-18	-577	74	94	727	-33	0	0	-4	2	0	1	97	43	-113	-27	0	-6
IN	-545	-408	0	-273	43	28	-47	1	17	74	-10	0	29	8	1	-2	-16	34	0
VBP	243	-36	71	-30	142	517	356	24	9	12	62	-4	58	345	5	906	87	9	0
NNP	-19	-3144	-310	2	-39	271	-11	0	0	97	-2	-1	-1	14	0	51	0	0	-9
VB	331	22	45	142	78	-507	8	8	6	0	153	-32	12	333	-427	45	17	-1	0
MD	105	-106	2	53	2	152	-332	-9	0	1	-82	832	2	0	0	403	12	2	0
VBN	6	-7	8	24	10	1384	-7	0	0	5	47	-1	0	53	130	35	6	11	0
JJ	-45	1	36	1	28	108	75	0	0	-4	-1	0	0	0	3	-88	8	0	-10
VBG	103	2	2	68	8	-46	8	0	0	8	17	-6	1	35	27	26	7	0	0
POS	-106	-365	4	-34	0	0	-3	0	0	-7	0	0	0	0	0	0	0	0	0
TO	125	-12	0	149	12	1	-4	0	0	206	18	4	0	0	0	0	1	0	0
CD	7	-7	12	-16	-2	38	48	0	0	356	0	0	0	1	-3	-10	0	0	0
RB	-34	0	-35	-1	-2	-38	-16	0	0	69	0	-2	0	-1	13	-4	-1	0	0
\$	0	0	0	0	0	3	-11	0	0	-546	0	0	0	0	0	0	0	0	0
NNPS	0	-306	-168	0	-29	0	-4	0	0	0	0	0	0	0	0	4	0	0	0
JJR	-42	0	0	0	0	8	4	0	0	-12	-1	0	0	0	0	-3	0	0	-8
WP	0	0	0	0	0	34	0	-11	10	0	0	0	44	0	0	-1	0	22	0
DT	0	0	0	0	0	42	-6	0	0	-3	-11	0	1	0	0	0	-1	0	0
WRB	-7	0	0	0	0	0	-4	5	13	0	0	1	4	7	0	0	0	15	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	-5	0	0	0
CC	0	-2	0	0	0	-13	7	0	0	0	0	-3	0	0	0	0	0	-5	0
RBR	-6	0	2	1	8	29	0	0	0	0	0	0	0	0	10	0	0	0	0
JJS	0	0	9	0	0	43	0	0	0	0	0	0	0	0	0	0	0	0	1
PRP	2	-4	-9	0	0	6	-5	0	0	0	0	0	0	0	0	-10	0	0	0
FW	0	0	3	0	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	3
SYM	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RBS	0	0	2	0	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.13: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 1) for EVG less DMV (with linear interpolation) across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

	POS	JJR	\$	NNPS	RP	WP	RBR	EX	JJS	WRB	WDT	SYM	UH	PDT	RBS	FW	LS
NN	-234	11	1	-9	-6	0	-3	0	12	0	1	0	0	4	0	1	0
ROOT	0	0	-1	<i>-129</i>	0	19	0	0	0	0	0	0	9	0	0	0	0
VBD	-1	95	-4	<i>-134</i>	55	5	<i>106</i>	1	12	17	0	0	10	0	3	-3	0
VBZ	-8	57	-1	-32	34	59	32	25	11	36	-1	1	0	0	9	0	1
NNS	-76	4	1	-5	0	0	0	0	64	-9	0	4	0	16	3	0	0
IN	-4	<i>-56</i>	<i>-34</i>	<i>-55</i>	0	-5	<i>-41</i>	5	23	14	<i>-10</i>	0	0	1	0	0	0
VBP	0	10	-3	-78	16	21	27	0	18	8	0	0	20	0	1	0	30
NNP	-18	0	0	-62	0	0	0	0	0	0	0	0	0	0	0	0	0
VB	1	25	1	1	-8	21	8	0	0	-1	0	0	6	0	5	0	0
MD	0	0	0	-3	0	0	0	3	0	8	-1	0	20	0	0	0	0
VBN	0	7	-1	0	0	-2	-7	0	1	0	0	0	0	0	-5	0	2
JJ	0	0	0	-3	0	0	<i>-125</i>	0	3	0	0	0	0	0	<i>-35</i>	0	0
VBG	0	0	-4	3	-1	9	1	0	6	0	0	0	-2	0	0	0	0
POS	0	0	0	-16	0	0	0	0	0	0	0	0	0	0	0	-2	0
TO	-1	-3	<i>-53</i>	-4	0	1	0	0	0	0	0	0	-4	0	0	0	0
CD	2	0	54	0	1	0	0	0	0	0	0	0	0	0	0	0	0
RB	-4	0	0	0	0	-4	28	0	0	0	0	0	0	0	0	0	0
\$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNPS	0	0	0	5	0	0	0	0	-7	0	0	0	0	0	0	0	0
JJR	-4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DT	0	0	0	0	0	0	0	0	0	0	0	0	0	<i>-18</i>	0	0	0
WRB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	0	3	0	0	0	2	0	0	-3	0	0	0	0
RBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.14: Difference in correct dependencies by child POS(columns) vs head POS(rows)(Part 2) for EVG less DMV (with linear interpolation) across 10 runs. Bold: abs(difference)>1000. Italics: abs(difference)>100.

to dependencies headed by verbs, as well as ROOT (Table 3.12). Breaking the results down by child part-of-speech, major improvements accrue to IN, CC, NN, JJ, VBN, VBZ, and VBP dependents (Table 3.11). In Table 3.13 we see that ROOT improvements mostly come from identifying verbs as the root. Net newly correct dependencies headed by verbs come from across all child categories other than NNP (see below), notably NN, IN, and CC.

The major thing that EVG does worse than DMV are for NNPs as the child, as well as NN→DT dependencies, of which EVG with linear interpolation nets 2555 fewer than DMV with linear interpolation. (Table 3.13). Of the 6109 net fewer net correct dependencies with NNP as the child, 3144 are NNP headed, with most of the remainder distributed among VBZ, ROOT, VBD, IN, POS, and NNPS heads. It is notable that NNP dependents and NN→DT dependencies where a point of major improvement in going from DMV to EVG with linear interpolation, so these results can be thought of as partially offsetting each other when looking at the whole change from DMV to EVG.

We also plotted development set directed and undirected accuracy vs. variational lower bound for DMV and EVG each with linear interpolation. These are given in Figures 3.4a, 3.4b, 3.5a, and 3.5b, respectively. Each point represents 20 random restarts of its respective model, of which the one with the highest lower bound value is trained until convergence on WSJ10. Compared to Figure 2.7 for each of these models there is a clearer group of points in the upper right of the graphs, which corresponds to high accuracy and a relatively high value for the variational lower bound. One can also see that the upper-right group of points for EVG with linear interpolation has higher accuracy than those for DMV with linear interpolation.

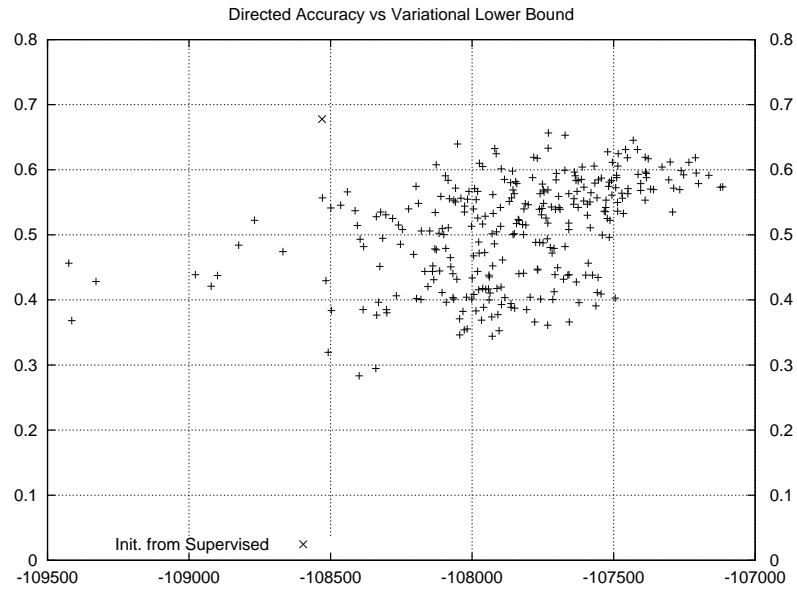
Additionally, we tried initializing each model from the gold standard dependency trees, and training the models until convergence. Mixing parameter variational posteriors are initialized to their priors. The results are marked with an X in Figures 3.4a, 3.4b, 3.5a, and

3.5b In both cases the run initialized from the gold standard results in a high accuracy by both measures, and a low value of the variational upper bound.

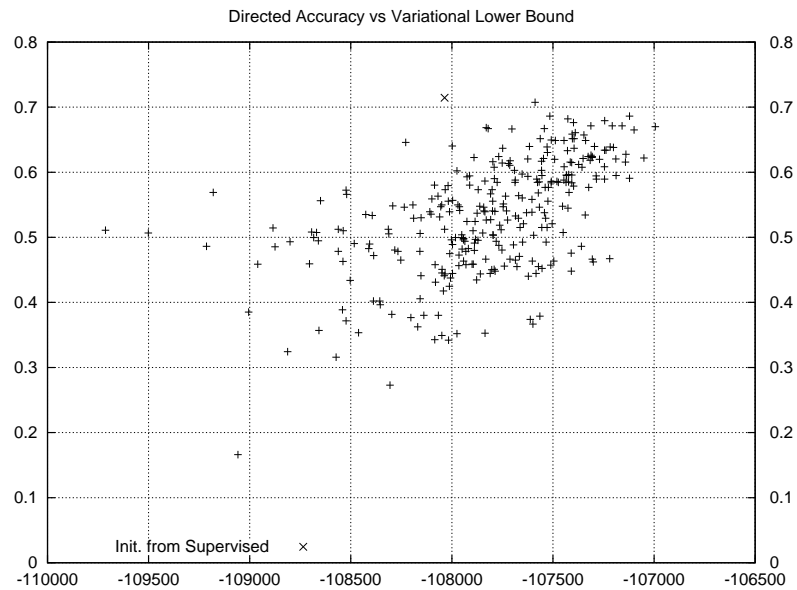
Also in both cases, the accuracy decreased from a high initial value (DMV: 0.778, EVG: 0.781) to a relatively lower converged value (DMV: 0.678, EVG: 0.714), indicating model errors. This is unsurprising, since we know our unsupervised models to be very unsophisticated relative to those used in supervised parsing.

3.7 Conclusions

In this chapter we examined several different smoothing schemes for PCFGs, and evaluated these schemes on the DMV and EVG. Overall we saw that the Variational Bayes approaches tended to perform better than setting the hyperparameters via maximum likelihood on a held-out set. The linear interpolation approach in which prior weight is placed on both parameters outperform the linear interpolation techniques in which the prior simply puts weight on the backoff distribution. We saw that having a reasonably strong prior was useful, but also that estimating the mixture weights contributes a great deal to good performance, so we are not simply setting the mixture weights by having a strong prior.

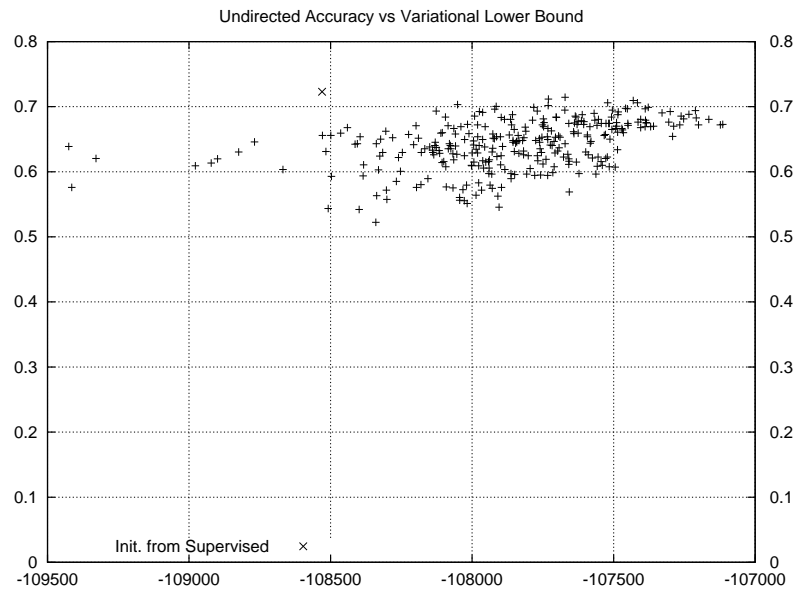


(a) DMV with linear interpolation.

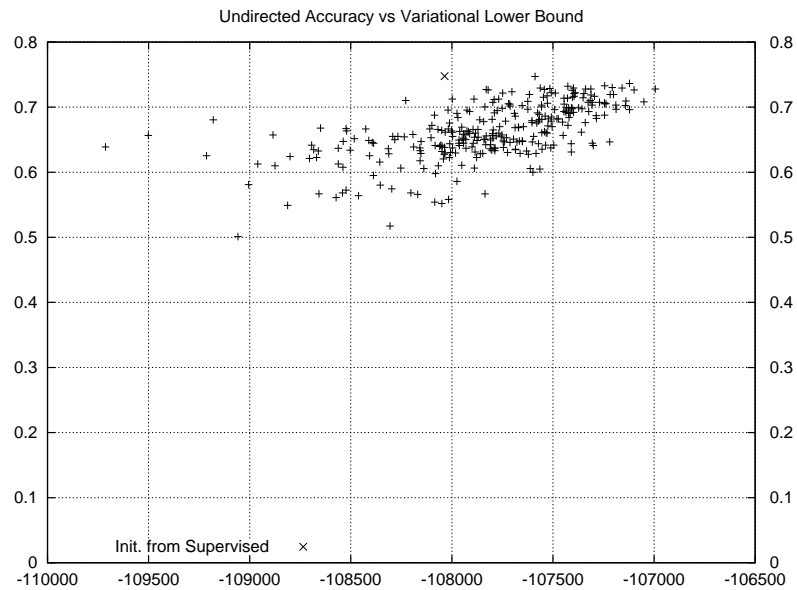


(b) EVG with linear interpolation.

Figure 3.4: Directed Accuracy vs Lower Bound, 300 runs of DMV/EVG with linear interpolation. Run initialized from gold standard dependencies marked with x.



(a) DMV with linear interpolation.



(b) EVG with linear interpolation.

Figure 3.5: Undirected Accuracy vs Lower Bound, 300 runs of DMV/EVG with linear interpolation. Run initialized from gold standard dependencies marked with x.

Chapter 4

Lexicalization

4.1 Introduction

In the previous chapter we saw that smoothing can improve the learning performance of DMV and EVG. These models, as we have seen, treat the parts-of-speech as observed when inducing the dependency structure and model parameters. We have, however, an additional source of observed information that these models are not leveraging: the words themselves. Lexical features are a key element of state-of-the-art supervised dependency and constituency parsers (e.g. (Charniak, 2000; Collins, 1999; Eisner, 1996; McDonald et al., 2005)...), and since that lexical information is available it seems reasonable to try to make use of it.

The reason lexical features are potentially useful is because the current units of generalization DMV and EVG use, the parts-of-speech, are too coarse to capture many of the phenomena we see in syntax. For instance, we might be interested in capturing the fact that transitive, intransitive and ditransitive verbs should have different numbers of arguments. Similarly, we might presume different nouns or verbs might be more or less likely to be modified, or have particular modifiers or arguments.

In this chapter we will explore a series of different ways of incorporating lexical features into the basic EVG model. These will examine both the extant distribution classes in EVG, namely the stopping and argument part-of-speech distributions, as well as the new distribution over argument word.

4.2 Lexicalized Grammar Framework

The basic approach in this chapter will be to extend the EVG split-head bilexical PCFG described in Chapter 2 to incorporate any of the lexical features we will make use of here. We will then be able to make independence assumptions, using the tied-PCFG framework, to investigate the effects of different model structures.

We can first replace every instance of a part-of-speech A as an event with the part-of-speech/word pair Aa . We will then want to factor the distribution that predicts the part-of-speech/word pair Aa given some conditioning context information C as:

$$P(Aa|C) = P(A|C)P(a|C)$$

In order to incorporate this, we have to extend EVG to allow the nonterminals to be annotated with both the words and parts-of-speech of the head. This is done as follows. First we remove the old rules $Y_A \rightarrow L_A R_A$ for each part of speech $A \in V_\tau$. Then we mark each nonterminal annotated with a part-of-speech as also annotated with its head, with a single exception: Y_A . We will replace Y_A with a nonterminal Y_{AHh} , one for each (argument part-of-speech, head part-of-speech, head word) triplet. Argument part-of-speech generating rules such as $L_{Hh}^1 \rightarrow Y_A L_{Hh}^0$ will be replaced with $L_{Hh}^1 \rightarrow Y_{AHh} L_{Hh}^0$. We will add a new nonterminal Y_{Aa} for each $A \in V_\tau, a \in V_w$, and the rules $Y_{AHh} \rightarrow Y_{Aa}$ and $Y_{Aa} \rightarrow L_{Aa} R_{Aa}$. The rule $Y_{AHh} \rightarrow Y_{Aa}$ corresponds to selecting the word, given its

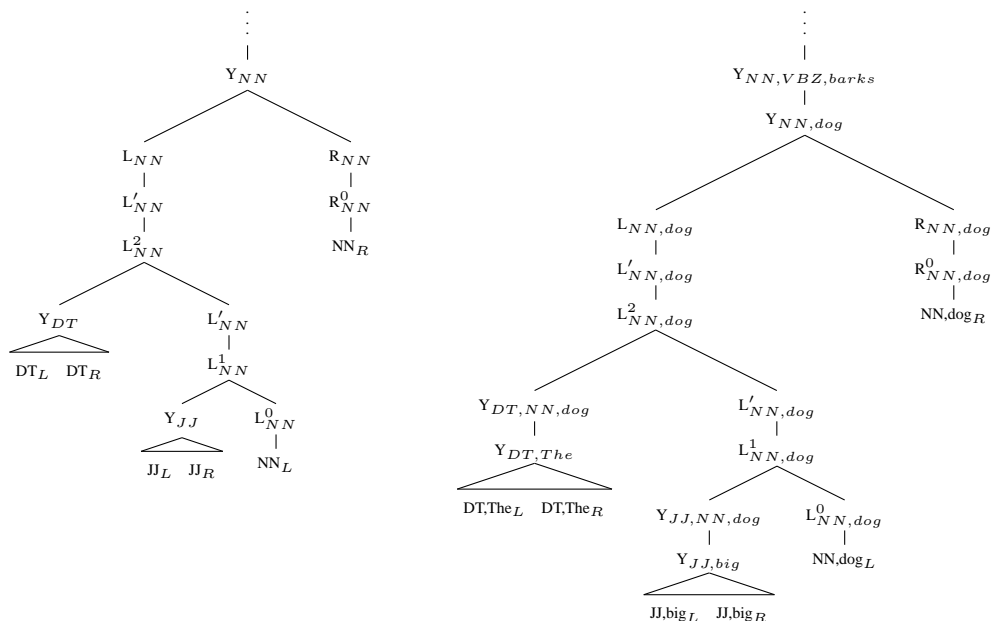


Figure 4.1: A subtree from an unlexicalized EVG parse, and the corresponding subtree in lexicalized EVG, which adds word annotations.

part-of-speech, the part-of-speech of its head, and its head.

4.3 Experimental Setup

Since lexical features are much sparser than unlexical features, we will train our lexical models on larger datasets than those in Chapter 3.

In addition to WSJ10, we include 1,327,754 words of New York Times from Gigaword (Graff, 2003), prepared as follows. We used sections 199407 to 199410. We included only documents marked as article type “story”, and included only unique sentences of greater than length 1, which excludes many clear examples of non-text in the input.

We trained Ratnaparkhi’s part-of-speech tagger on the full Penn Treebank Wall Street Journal, sections 2-21. We then tag the New York Times data, removed punctuation, and kept only sentences of length no more than 10. This results in an additional 1,327,754

Lexicalized Extended Valence Grammar (LEVG)	
Rule	Description
$S \rightarrow Y_{ARr}$	select A as root part-of-speech, (Rr indicates root)
$Y_{AHh} \rightarrow Y_{Aa}$	generate a as word of part-of-speech A
$Y_{Hh} \rightarrow L_{Hh} \quad R_{Hh}$	Move to split-head representation
$L_{Hh} \rightarrow L_{0Hh}$	stop generating arguments left,head= H , no arguments
$L_{Hh} \rightarrow L'_{Hh}$	continue generating arguments left,head= H no arguments
$L'_{Hh} \rightarrow L_{1Hh}$	stop generating arguments left,head= H , one or more arguments
$L'_{Hh} \rightarrow L_{2Hh}$	continue generating arguments left,head= H , one or more arguments
$L_{2Hh} \rightarrow Y_{AHh} \quad L'_{Hh}$	argument= A left,head= H , argument is not nearest to head
$L_{1Hh} \rightarrow Y_{AHh} \quad L_{0Hh}$	argument= A left,head= H , argument is nearest to head
$L_{0Hh} \rightarrow Hh_L$	
$R_{Hh} \rightarrow R_{0Hh}$	stop generating arguments right,head= H , no arguments
$R_{Hh} \rightarrow R'_{Hh}$	continue generating arguments right,head= H no arguments
$R'_{Hh} \rightarrow R_{1Hh}$	stop generating arguments right,head= H , one or more arguments
$R'_{Hh} \rightarrow R_{2Hh}$	continue generating arguments right,head= H , one or more arguments
$R_{2Hh} \rightarrow R'_{Hh} \quad Y_{AHh}$	argument= A right,head= H , argument is not nearest to head
$R_{1Hh} \rightarrow R_{0Hh} \quad Y_{AHh}$	argument= A right,head= H , argument is not nearest to head
$R_{0Hh} \rightarrow Hh_R$	

Table 4.1: Schema for Lexicalized EVG

words of data.

Each lexicalized model is initialized from the best result of running 600 smoothed EVG models, trained on WSJ10. All results in this section will report the average over 10 runs, meaning in sum we run 6000 EVG models, select the top model of each cohort of 600, and initialize the lexicalized model from that. We use the variant of EVG smoothed using linear interpolation with $\text{Beta}(2K, K)$ priors (K = number of parts-of-speech) on the mixture parameters, which bias the learners to put weight on both terms.

To separate out effects of the random initialization of EVG across each of our lexicalized models, as well as make experiments on larger amounts of data practical, we use the same initializing trained EVG parameters for each of the models below.

To prepare for unknown words, we replaced any word seen C or fewer times in the training sets with “unk”, where unless otherwise specified $C = 5$.

4.4 Models

4.4.1 Argument Distribution: Model 1

One possibility is to consider how the distribution over argument parts-of-speech might depend on the head word. We might imagine the head part-of-speech to be too coarse a substitute for the head word, and that a more precise class might be helpful. For instance, “says” is usually tagged as a VBZ. However, “says” often occurs in constructions such as “John says the big dog barks.” so it is much more likely to have a verb as its argument than e.g. “gives” does. We will model this phenomenon by conditioning the probability of an argument part-of-speech on its head word.

Our first model explores extending EVG by conditioning the probability of the dependent part-of-speech A on the head word h in addition to the head POS H , valence position

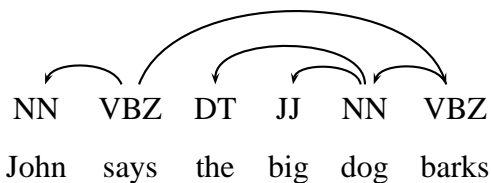


Figure 4.2: Dependency tree for sentence with head “says.” Notice that the right argument of “says” is a VBZ, which would be unlikely as the right argument of the VBZ “gives”.

v and direction d . We smooth this distribution with the smoothed EVG model described in Chapter 3. The backoff structure is outlined in Tables 4.2 and 4.3. We explored two conditioning structures for the λ s associated with selecting between our new component, and the old smoothed EVG dependent part-of-speech distribution. In one, Lexicalized Model 1a, λ s are conditioned on the head word, head part-of-speech, valence and direction; that is, the entire conditioning information available. In the second, Lexicalized Model 1b, λ s are conditioned on the head part-of-speech, valence and direction, and ignoring the head word in determining whether to smooth. The reason this might be advantageous is that in cases where the head word has been relatively rarely seen, we would have less information about whether to smooth where we really want that information.

In this experiment the “Stop” distribution, which determines whether to generate an additional dependent of head h in direction d with valence bit v is taken from the earlier described smoothed EVG model. The distribution over the dependent word will be conditioned only upon its part-of-speech, both of which are observed in this case.

To initialize, we leave the stop, dependent word, and unlexicalized dependent POS components set to their smoothed EVG settings. $P(A|Hhvd)$ is initialized to $P(A|Hvd)$, and the new lambdas are initialized to their prior settings.

Results

We trained using Variational Bayes, with nonsparse priors, which seemed to be the most consistently effective technique from Chapter 3. The basic results are given in Table 4.4.

Component		λ conditioning
Stop	$P(s Hvd)$	—
Dependent POS	$P(A Hhvd)$	$hHvd$
	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a A)$	—

Table 4.2: Lexicalized Model 1a backoff chains. The dependent part-of-speech distribution is lexicalized. The λ s associated with interpolating between $P(A|Hhvd)$ and $P(A|Hvd)$ are conditioned on the events $hHvd$.

Component		λ conditioning
Stop	$P(s Hvd)$	—
Dependent POS	$P(A Hhvd)$	Hvd
	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a A)$	—

Table 4.3: Lexicalized Model 1b backoff chains. The dependent part-of-speech distribution is lexicalized. It differs from Lexicalized model 1a in that the λ s associated with interpolating between $P(A|Hhvd)$ and $P(A|Hvd)$ are conditioned on the events Hvd rather than $hHvd$.

Model	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
Smoothed EVG	0.658	0.629	0.734	0.712
Lexicalized 1a	0.684	0.650	0.751	0.722
Lexicalized 1b	0.694	0.666	0.757	0.737

Table 4.4: Results of Lexicalized models 1 and 1b

As we can see, conditioning on the head word give a major improvement over the basic smoothed EVG model, using either variant. Variant 1b seems to perform as well or better than 1a.

4.4.2 Stop Distribution

One important area where we expect to see a benefit to lexicalization is with with regards to the transitivity of verbs. For instance, the verb “give” often has both a direct object and an indirect object, while “sleep” is very unlikely to have an indirect object.

Our second model explores extending EVG by conditioning the probability of generating another dependent on the head word in addition to the head POS, valence position and direction. The main hope is that we should be able to learn more specific valence information, particularly with regards to the transitivity of verbs. The backoff chain for this model is given in Table 4.5. We also examined adding the lexicalized stop distribution to Lexicalized Model 1b, which we denote as Model 3 (see Table 4.6).

The results are shown in Table 4.7. We can see that adding lexical conditioning to the stop distributions does not help performance either when starting from Smoothed EVG (Lexicalized Model 2), or when starting from Lexicalized Model 1b (Lexicalized Model 3).

Component		λ conditioning
Stop	$P(s hHvd)$	Hvd
	$P(s Hvd)$	—
Dependent POS	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a A)$	—

Table 4.5: Lexicalized model 2 backoff chains. The stop distribution is lexicalized, while the dependent part-of-speech distribution is not. Equivalent to smoothed EVG, with a lexicalized stop distribution.

Component		λ conditioning
Stop	$P(s hHvd)$	Hvd
	$P(s Hvd)$	—
Dependent POS	$P(A hHvd)$	Hvd
	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a A)$	—

Table 4.6: Lexicalized model 3 backoff chains. Both the stop and dependent part-of-speech distributions are lexicalized. Equivalent to Lexicalized Model 1b with the addition of a lexicalized stop distribution.

Model	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
Smoothed EVG	0.658	0.629	0.737	0.713
Lexicalized 1b	0.694	0.666	0.757	0.737
Lexicalized 2	0.663	0.626	0.735	0.707
Lexicalized 3	0.687	0.656	0.753	0.729

Table 4.7: Results of Lexicalized models 2 and 3.

4.4.3 Generating the Dependent word.

In the models above we have taken the probability of a word to depend only upon its part-of-speech. This makes the event of predicting the word given its part-of-speech be observed, and so does not rely on any unobserved structure such the word's head. In this section we explore using additional conditioning information for this distribution.

Our fourth model explores conditioning the dependent word on not only its part-of-speech, but the head part-of-speech as well. We extended model 1b by adding this conditioning event, smoothing it using the probability of the word given its part-of-speech. The backoff chains are summarized in Table 4.8.

Our fifth model modifies model 4 by conditioning the dependent word on the head word rather than head part-of-speech. This can be thought of as making use of the intuition of Paskin's Lexical Attraction model (Paskin, 2001), where the argument word is generated conditioned on the head word. The idea is that pairs of words have a semantic affinity in addition to their syntactic affinity, and this can be used

Where as Paskin's model takes dependency trees to be equally likely, we have both the valence modeling inherited from DMV and EVG to guide us, as well as part-of-speech information. The backoff chains are given in Table 4.9.

In our experiments with Model 5, we raise the unknown word cutoff to 100, to make the model estimation feasible within reasonable memory requirements. This results in a vocabulary size of 755 words, with the remainder replaced with the word "unk".

The results are given in Table 4.10. Unfortunately neither model improves upon the basic Lexicalized model 1b.

Component		λ conditioning
Stop	$P(s hHvd)$	Hvd
Dependent POS	$P(A hHvd)$	Hvd
	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a AHd)$	Hd
	$P(a A)$	—

Table 4.8: Lexicalized model 4 backoff chains

Component		λ conditioning
Stop	$P(s hHvd)$	Hvd
Dependent POS	$P(A hHvd)$	Hvd
	$P(A Hvd)$	Hvd
	$P(A vd)$	—
Dependent Word	$P(a Ahd)$	AHD
	$P(a A)$	—

Table 4.9: Lexicalized model 5 backoff chains

Model	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
Smoothed EVG	0.658	0.629	0.737	0.713
Lexicalized 1b	0.694	0.666	0.757	0.737
Lexicalized 4	0.696	0.660	0.762	0.734
Lexicalized 5	0.693	0.659	0.755	0.728

Table 4.10: Results of Lexicalized models 4 and 5

Training set	Number of Words	Directed Accuracy		Undirected Accuracy	
		Train	Dev.	Train	Dev.
WSJ10	5777	0.658	0.629	0.737	0.713
WSJ+NYT199407	50771	0.639	0.600	0.719	0.694
WSJ+NYT199407-10	195805	0.616	0.581	0.703	0.679

Table 4.11: Effect of training linear interpolated EVG with different amounts of training data. We can see that adding increase amounts of NYT degrades accuracy on WSJ.

4.5 Discussion

One difference between the experiments in this chapter in contrast with previous sections is the addition of the additional NYT199407-10 data. We examined the effect of using additional data with EVG. We ran the system with both the WSJ+NYT199407 and WSJ+NYT199407-10 datasets, in addition to the regular WSJ10 set. Recall that our search scheme involves running 600 random models in the first stage, and selecting the highest objective function model from each cohort of 20, and running each of those 30 models to convergence. As in the lexicalized experiments the first search stage was run on WSJ10, and the selected models were then to convergence on each of the training sets in question. As usual we report the average over 10 rounds. The results are presented in Table 4.11. As we can see, adding the additional NYT data to training seems to degrade performance by quite a bit.

Since what we expected was that more data should help, or at least not hinder performance, we were interested in why this was the case. We examined two hypotheses: that using part-of-speech tags from the Ratnaparkhi tagger was sufficiently worse than the gold tags to cause the drop in performance, and that the additional data was noisy or diverged in domain from the WSJ development set. To evaluate the first hypothesis, we split the training set into two sections, trained the Ratnaparkhi tagger on one half to tag the other,

Training set	Number of Sentences	Directed Accuracy		Undirected Accuracy	
		Train	Dev.	Train	Dev.
WSJ10-gold tags	5777	0.658	0.629	0.737	0.713
WSJ10-tagged	5777	0.679	0.643	0.746	0.721
NYT10-tagged	5622	–	0.548	–	0.669

Table 4.12: Effect of training linear interpolated EVG when training from automatically tagged WSJ and NYT Gigaword corpora, controlling for corpus size. Note that since we do not have gold standard trees for Gigaword we cannot present parsing results on the training set.

and vice versa. We then train on this tagged WSJ10, and evaluate on the dev. set. To evaluate the second hypothesis, we trained on a small subcorpus of 39278 words of the NYT Gigaword corpus (of comparable size to the WSJ10 corpus). The idea here was to control for corpus size by using a corpus of about the same size, while training on this additional data. The results are presented in Table 4.12. First of all it is clear that using non-gold-standard parts-of-speech does not impede accurate model learning—in fact in this case we get slightly better performance from the automatically tagged corpus. Second of all, we do in fact see a severe loss in performance when we train on the short Gigaword corpus. This could be indicative that the Gigaword corpus despite our efforts to clean it up is still sufficiently noisy to degrade the performance of the system. Or put another way, our system is unfortunately very sensitive to noisy data, which is of course a major weakness in an unsupervised approach.

One question this leads us to whether the lexicalized systems help when trained on only the WSJ10 corpus. Recall that a major difficulty will be whether we can get reasonable statistics for lexical items on such a small corpus. In this case we raised the cutoff for words to be marked “unk” if they have not been seen 100 times. We tried both the Lexicalized Model 1b and the Lexicalized model 3, which if you recall extends Lexicalized model 1b

Model	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
Smoothed EVG	0.658	0.629	0.737	0.713
Lexicalized Model 1b-100	0.686	0.651	0.748	0.722
Lexicalized Model 3-100	0.666	0.635	0.737	0.714

Table 4.13: Effect of training Lexicalized models 1b and 3 on WSJ10, setting the unknown word cutoff to 100 times.

by adding stop probability conditioning information. The results are given in Table 4.13. The results are qualitatively similar to our earlier experiments, although in this situation Model 3 does not perform quite as well. There is a benefit for both Lexicalized models from using more data.

4.5.1 Lexicalizing DMV

In all of the above experiments we considered the effect of adding lexicalization to the smoothed EVG model, which was the most effective model from Chapter 3. However, in order to examine whether adding lexicalization is useful for models other than smoothed EVG, we here look at adding lexicalization to the basic Dependency Model with Valence.

Since conditioning the part-of-speech distribution on the head word was so successful for EVG, we will do the same for DMV. We smooth $P(A|Hhd)$ with $P(A|Hd)$ using linear interpolation, with a $\text{Beta}(|\mathcal{R}_N|, 2|\mathcal{R}_N|)$ prior on the mixture distributions. The stop distribution we leave the same as in DMV. Words are conditioned on their parts-of-speech. The backoff schema is given in Table 4.14.

We trained this model on WSJ10, setting words seen fewer than 100 times to unk. As with the other lexicalized models, we train the first stage of the beam using regular DMV, selecting the restart with the highest lower-bound, and using that to initialize the

Component		λ conditioning
Stop	$P(s Hvd)$	—
Dependent POS	$P(A hHd)$	Hvd
	$P(A Hd)$	—
Dependent Word	$P(a A)$	—

Table 4.14: Lexicalized DMV backoff chains. The dependent part-of-speech distribution is lexicalized, and smoothed with the basic DMV dependent part-of-speech distribution.

Model	Directed Accuracy		Undirected Accuracy	
	Train	Dev.	Train	Dev.
DMV	0.583	0.549	0.650	0.631
Lexicalized DMV	0.611	0.582	0.705	0.687

Table 4.15: Effect of training Lexicalized DMV on WSJ10, setting the unknown word cutoff to 100 times. We compare it to DMV, to show that adding just lexicalization to DMV improves performance even without the gains from Chapter 3.

Lexicalized DMV.

The results are given in Table 4.15. As we can see adding lexicalization to the argument distribution improves development set directed accuracy from 0.549 to 0.582, and undirected accuracy from 0.631 to 0.687. Hence adding lexicalization is useful, even in the absence of additional valence information or other smoothing.

4.6 Analysis: Lexicalized 1b-100 vs EVG

We analyze the difference between Lexicalized Model 1b-100 and EVG on section 2-21, by presenting the net differences in correct edges, shown in Table 4.16 (by child part-of-speech and direction), Table 4.17 (by head part-of-speech and direction), and Tables 4.18 and 4.19 (by head/child part-of-speech pair).

In Table 4.16 we see that the major improvements in the lexicalized model come for dependencies whose children are NNP, DT, NN,NNS, VBN, JJ, VB, and NNPS. The main thing it does worse than unlexicalized EVG is for dependencies whose children are RB. Breaking the difference down by head part-of-speech in Table 4.17, the largest improvements accrue to NNP, NN, IN, VBD, NNS, and JJ, though improvements seem very broadly distributed. Looking at Tables 4.18 and 4.19 (NNP,NNP) and (NN,DT) dependencies are the two largest type of net improvement in moving to lexicalization.

Scatterplots of development set directed accuracy vs. variational lower bound and undirected accuracy vs. lower bound for Lexicalized Model 1b-100 are given in Figures 4.3 and 4.4 respectively. Each point represents 20 random restarts of EVG, of which the one with the highest EVG lower bound value initializes a Lexicalized Model 1b-100 model, which is trained until convergence on WSJ10. We can see very starkly that the right-hand side of the graph corresponds with an area of high accuracy.

Child POS	Correct EVG	Correct Lex1b	Difference (Lex1b-EVG)	Difference Left	Difference Right
NNP	22584	27070	4486	3749	737
DT	26320	28503	2183	2261	-78
NN	44127	45347	1220	672	548
RB	12419	11324	-1095	511	-1606
NNS	23361	24376	1015	863	152
VBN	6766	7750	984	33	951
JJ	15139	15958	819	471	348
VB	5683	6411	728	-20	748
CD	9435	10001	566	709	-143
NNPS	534	1022	488	276	212
VBD	16365	16721	356	122	234
CC	4200	3852	-348	-358	10
POS	1532	1864	332	326	6
IN	15976	15729	-247	-160	-87
VBP	9649	9500	-149	-108	-41
VBZ	16465	16344	-121	-128	7
PRP	15139	15039	-100	91	-191
VBG	4022	4106	84	-23	107
TO	2824	2906	82	3	79
MD	5017	4948	-69	-16	-53
WP	620	673	53	-4	57
JJS	368	316	-52	-44	-8
PRP\$	2290	2331	41	41	0
UH	225	185	-40	-30	-10
RBR	547	583	36	98	-62
PDT	54	86	32	26	6
JJR	889	859	-30	11	-41
\$	556	531	-25	-100	75
RBS	79	95	16	26	-10
WRB	509	495	-14	0	-14
LS	51	64	13	13	0
FW	87	100	13	14	-1
WDT	139	148	9	-1	10
EX	855	863	8	6	2
RP	1102	1099	-3	0	-3
SYM	151	153	2	2	0

Table 4.16: Change in correct dependencies by child part-of-speech across 10 runs. Left indicates child is a left dependent of its head in system output.

Head POS	Correct EVG	Correct Lex1b	Difference (Lex1b-EVG)	Difference Left	Difference Right
NNP	9131	11227	2096	2429	-333
NN	36212	37851	1639	2346	-707
IN	21329	22584	1255	136	1119
VBD	33569	34567	998	769	229
NNS	21288	22068	780	741	39
JJ	938	1649	711	728	-17
VBN	5107	5738	631	162	469
ROOT	50173	50795	622	0	622
POS	2028	2575	547	547	0
VBZ	33131	33640	509	362	147
NNPS	210	690	480	480	0
VB	9336	9762	426	129	297
\$	404	654	250	8	242
RB	1369	1171	-198	37	-235
VBG	3545	3742	197	28	169
CD	2321	2427	106	138	-32
TO	2993	3097	104	11	93
MD	11075	11177	102	114	-12
JJR	37	116	79	83	-4
WP	174	221	47	3	44
DT	111	79	-32	-8	-24
UH	38	13	-25	-19	-6
JJS	53	32	-21	-10	-11
FW	60	70	10	13	-3
RBR	54	45	-9	18	-27
PRP	62	53	-9	-10	1
RBS	11	3	-8	-1	-7
WRB	303	296	-7	0	-7
CC	131	125	-6	0	-6
SYM	92	88	-4	2	-6
RP	10	14	4	0	4
WDT	8	7	-1	0	-1
VBP	20776	20776	0	96	-96
PRP\$	0	0	0	0	0
PDT	0	0	0	0	0
LS	0	0	0	0	0
EX	0	0	0	0	0

Table 4.17: Change in correct dependencies by head part-of-speech across 10 runs. Left indicates child is a left dependent of its head in system output.

	NN	NNP	DT	NNS	JJ	IN	RB	VBZ	VBD	CD	PRP	VB	VBP	VCN	TO	CC	VBG	MD	PRP\$
NN	-98	-59	1612	174	157	-577	11	0	0	244	-4	1	0	-50	-16	7	-21	0	39
ROOT	30	386	0	-20	0	51	-11	-33	112	15	0	33	-65	81	-2	-6	1	-59	0
VBD	286	400	-2	288	9	-172	-375	-6	62	-14	-15	157	-12	285	58	-95	49	-5	0
VBZ	273	489	2	36	166	-112	-479	-131	28	-27	4	97	-92	321	5	-137	17	-17	0
NNS	-44	14	295	98	161	-37	1	0	0	191	5	0	0	44	28	0	-24	0	-14
IN	544	268	-25	73	11	6	25	34	156	-46	-14	7	25	-2	0	1	39	11	0
VBP	-43	4	-30	248	135	-90	-481	-7	-5	-3	-44	203	-9	162	-9	-94	30	-2	0
NNP	4	2192	151	0	23	-113	1	0	0	-181	-2	-1	1	5	0	-29	0	0	7
VB	34	22	-13	15	0	198	29	3	-6	0	-27	80	-6	111	-11	-2	-6	1	0
MD	61	73	2	25	8	-22	-156	-3	0	8	7	148	-2	0	0	-44	-7	-1	0
VCN	-9	8	4	-2	-12	595	3	0	1	-7	-13	4	0	13	25	2	3	0	0
JJ	-10	1	33	15	116	-21	403	0	0	9	4	0	4	0	10	26	1	0	10
VBG	25	3	-1	33	11	89	30	-10	0	2	1	0	5	11	-21	1	2	3	0
POS	109	363	-3	54	0	0	1	0	0	6	0	0	0	0	0	0	0	0	0
TO	40	22	0	-36	2	0	1	0	0	42	-3	0	0	0	0	0	0	0	0
CD	25	13	15	6	10	-13	-17	0	0	161	0	0	0	2	-2	8	0	0	0
RB	-9	0	-3	-1	-11	19	-147	0	0	-63	0	-1	0	-2	21	11	-1	0	0
\$	-8	0	0	0	3	3	13	0	0	239	0	0	0	0	0	0	0	0	0
NNPS	0	296	147	0	32	0	0	0	0	0	0	0	0	0	0	-5	0	0	0
JJR	7	0	3	3	4	-2	50	0	0	3	1	0	0	0	0	7	0	0	3
WP	0	0	0	0	0	-6	1	30	19	0	0	0	1	0	0	1	0	1	0
DT	0	0	0	0	0	-15	2	0	0	-13	0	0	3	0	0	0	1	0	0
WRB	1	0	0	0	0	0	-1	2	-11	0	0	1	-2	4	0	0	0	-1	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
CC	-2	0	0	0	0	2	-3	0	0	0	0	-1	0	0	0	0	0	0	0
RBR	8	0	-2	10	-7	-16	2	0	0	0	0	0	0	0	-4	0	0	0	0
JJS	0	0	-9	0	0	-11	0	0	0	0	0	0	0	0	0	0	0	0	-1
PRP	-3	-5	-1	0	0	-2	2	0	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	9	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	-3
SYM	0	0	0	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	-2	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RBS	0	0	-1	0	-3	-4	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	1	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.18: Correct: Child POS(columns) vs Head POS(rows)(Part 1)

	POS	JJR	\$	NNPS	RP	WP	RBR	EX	JJS	WRB	WDT	SYM	UH	PDT	RBS	FW	LS
NN	234	-6	3	7	1	1	-5	0	-12	0	-1	0	0	-2	0	-1	0
ROOT	0	0	-6	139	0	-4	0	0	0	0	0	0	-20	0	0	0	0
VBD	4	-24	23	122	-8	0	-17	3	-2	-6	0	0	2	0	0	3	0
VBZ	7	-9	5	29	4	22	-6	7	-1	-1	0	5	0	0	-4	9	-1
NNS	67	-3	0	4	0	-8	0	0	-34	4	0	-3	0	38	-3	0	0
IN	4	19	8	52	0	32	15	-1	0	2	10	0	2	-1	0	0	0
VBP	0	-3	1	74	6	7	-11	-2	-9	-10	0	0	-8	0	0	0	-10
NNP	13	0	0	32	0	-7	0	0	0	0	0	0	0	0	0	0	0
VB	1	-1	4	1	-2	0	-3	0	0	-4	0	0	-2	0	-6	0	16
MD	0	0	0	2	0	0	0	1	0	2	0	0	0	0	0	0	0
VBN	0	-2	0	1	-3	2	0	0	2	0	0	0	0	0	8	0	8
JJ	0	0	0	3	0	0	84	0	2	0	0	0	0	0	21	0	0
VBG	0	0	4	-3	0	9	1	0	-3	0	0	0	5	0	0	0	0
POS	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	2	0
TO	2	2	33	1	0	0	0	0	0	0	0	0	-2	0	0	0	0
CD	-1	0	-100	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
RB	0	-2	0	4	0	1	-22	0	0	0	0	0	1	7	0	0	0
\$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NNPS	0	0	0	5	0	0	0	0	5	0	0	0	0	0	0	0	0
JJR	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DT	0	0	0	0	0	0	0	0	0	0	0	0	0	-10	0	0	0
WRB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WDT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	0	-2	0	0	0	-1	0	0	1	0	0	0	0
RBR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SYM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
UH	0	0	0	0	0	0	0	0	0	0	0	0	-19	0	0	0	0
RBS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.19: Correct: Child POS(columns) vs Head POS(rows)(Part 2)

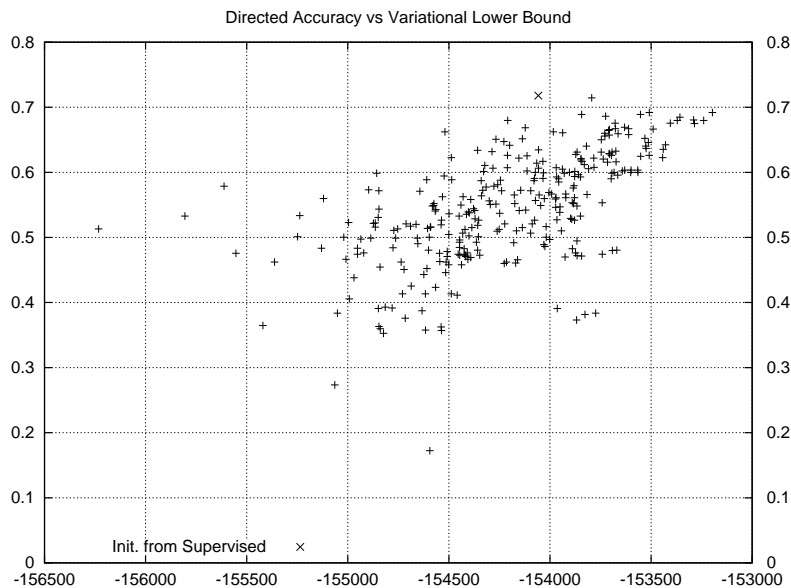


Figure 4.3: Directed Accuracy vs Lower Bound, 300 runs of Lexicalized Model 1b-100.

We also tried initializing Lexicalized Model 1b from the gold standard trees, and running variational Bayes until convergence. As in the previous chapter we initialized mixing parameter variational posteriors to their priors. This is denoted with an X in Figures 4.3 and 4.4. As was the case for the analogous experiments for DMV and EVG in Chapter 3, while the accuracy with supervised initialization is a little bit higher than those found through the usual method, the lower bound associated with that point is relatively low.

Again, the accuracy found by initializing the model from the gold-standard trees is much higher, and decreases as we run. It starts with a dev set directed accuracy of 0.780 and undirected accuracy of 0.812, and ends at 0.710 and 0.749 respectively. This indicates, as one might expect, that our model is biased and that some of our errors are model errors.

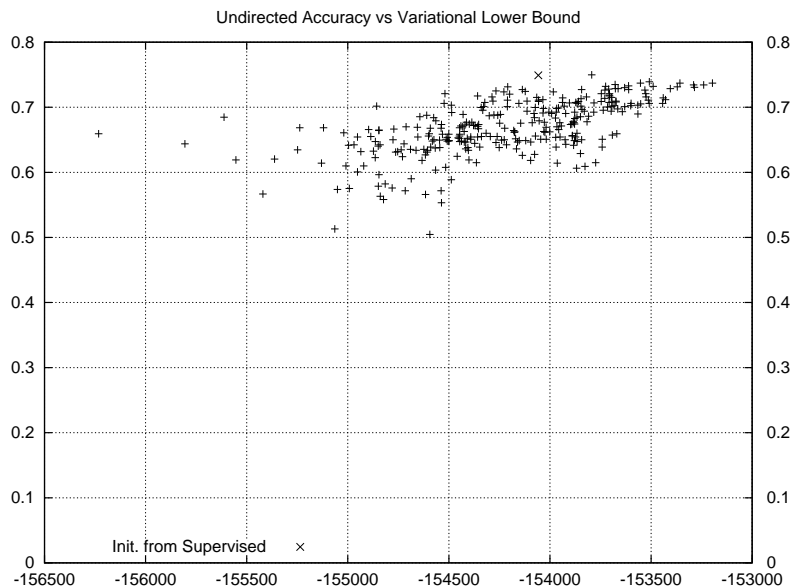


Figure 4.4: Undirected Accuracy vs Lower Bound, 300 runs of Lexicalized Model1b-100.

4.7 Conclusions

In this chapter we examined the utility of lexical features for dependency grammar induction. We found that we were able to make great improvements to the linearly interpolated EVG model by adding lexical conditioning when predicting the argument part-of-speech. Conditioning on the head part-of-speech also improved the basic DMV model, without additional valence information.

Disappointingly however, we were unable to get an improvement by incorporating lexicalization when predicting the valence of a particular word, as one would expect would matter for verb transitivity. We were also unable to find any improvement by conditioning the probability of an argument word on its head word, making use of bilexical affinities.

We also examined the effect of data size on our models. We saw a slight improvement for using additional data for our lexicalized models; however, the additional dataset we chose was far less clean than the Penn Treebank, and we saw a major decrease in performance on smoothed EVG when run on the New York Times data. This unfortunately

indicates that our technique is sensitive to noise, which is undesirable in an unsupervised technique.

Chapter 5

Final Matters

While the experiments presented throughout most of the thesis present results on the training and development sections of the Penn Treebank Wall Street Journal, (2-21 and 24 respectively), to place the work in this thesis in context with the rest of the field we will present results here for several models on the test section, 23.

DMV with a Logistic Normal Prior (Cohen et al., 2008) and with a Shared Logistic Normal Prior (Cohen and Smith, 2009), which we described in Section 1.2.5, as well as DMV with a log-linear prior (Berg-Kirkpatrick et al., 2010) are the best previous work on this task. The linear interpolated models presented are those with the $\text{Beta}(|R_N|, 2|R_N|)$ prior. Smoothed EVG and Lexicalized model 1b outperform both the previous work, and Lexicalized model 1b gives the highest reported accuracy on section 23.

We were also interested in whether the improvements we found for sentences of length no more than 10 remained for longer sentences. As such, we used the models learned to parse sentences with a length cutoff of no longer than 20 words. Table 5.2 shows the results of DMV, smoothed DMV, smoothed EVG, and Lexicalized model 1b on those sentences of Section 24. As we can see the same trends we saw for the shorter set remain for this larger testing set. Lexicalization proves its effectiveness, and we still see a major benefit

Model	Variant	Directed Accuracy
		Test
DMV	harmonic initializer	0.469
DMV	random initializer	0.548 $\left(\begin{smallmatrix} +0.034 \\ -0.067 \end{smallmatrix}\right)$
DMV	Log-Normal prior *	0.594
DMV	Shared Log-Normal prior *	0.624
DMV	Log-linear *	0.630
DMV	Linear Interpolated	0.604 $\left(\begin{smallmatrix} +0.024 \\ -0.019 \end{smallmatrix}\right)$
EVG	Linear Interpolated	0.652 $\left(\begin{smallmatrix} +0.044 \\ -0.015 \end{smallmatrix}\right)$
Lexicalized	Model 1b	0.681 $\left(\begin{smallmatrix} +0.033 \\ -0.018 \end{smallmatrix}\right)$

Table 5.1: Results on WSJ 10 test set (section 23). * Prior work.

from smoothing.

Model	Variant	Directed Accuracy	Undirected Accuracy
		Dev	Dev
DMV	random initializer	0.458 $\left(\begin{smallmatrix} +0.032 \\ -0.049 \end{smallmatrix}\right)$	0.586 $\left(\begin{smallmatrix} +0.010 \\ -0.016 \end{smallmatrix}\right)$
DMV	Linear Interpolated	0.492 $\left(\begin{smallmatrix} +0.008 \\ -0.011 \end{smallmatrix}\right)$	0.593 $\left(\begin{smallmatrix} +0.004 \\ -0.006 \end{smallmatrix}\right)$
EVG	Linear Interpolated	0.551 $\left(\begin{smallmatrix} +0.021 \\ -0.040 \end{smallmatrix}\right)$	0.638 $\left(\begin{smallmatrix} +0.008 \\ -0.020 \end{smallmatrix}\right)$
Lexicalized	Model 1b	0.592 $\left(\begin{smallmatrix} +0.025 \\ -0.022 \end{smallmatrix}\right)$	0.665 $\left(\begin{smallmatrix} +0.018 \\ -0.010 \end{smallmatrix}\right)$

Table 5.2: Results on WSJ 20 development set (section 24), i.e.

5.0.1 Other Languages

We also applied our grammar induction algorithms to several other language corpora from the CoNLL-X shared task (Buchholz and Marsi, 2006). We evaluated on Bulgarian, German, Japanese, Swedish, and Turkish. As with the English experiments, we kept each sentence with fewer than 10 words after punctuation was removed. We trained and evaluate on the training section for each corpus. The results are given for directed accuracy in Table 5.3 and for undirected accuracy in Table 5.4. For all languages other than Japanese, regular DMV with randomized initialization works as well or better than harmonic initialization.

For Bulgarian we see no improvement in directed accuracy going from DMV to DMV-LI, but adding lexicalization to DMV does seem to help. Adding extra valence hurts in both the lexicalized and unlexicalized versions of EVG.

For German and Swedish we see big improvement in moving to randomized initialization, but no major difference between the models otherwise.

For Japanese the harmonic initialization actually works better than randomized. We see an improvement adding the basic linear interpolation to DMV, but no major improvement after that.

For Turkish there is a big improvement in DMV moving to randomized initialization. However, adding linear interpolation to DMV results in dreadful directed accuracy scores. Examining the output, it seems this results in making each token the dependent of the word immediately to its left.

5.1 Another look at Randomized Initialization

It is undoubtedly the case that using a random-restart training procedure was beneficial to our results. One thing to note when comparing models is to what degree the improvement

	Bulgarian	German	Japanese	Swedish	Turkish
Harmonic Initializer					
DMV	0.535	0.372	0.656	0.412	0.443
DMV-LI	0.370	0.304	0.724	0.443	0.382
EVG-LI	0.366	0.362	0.724	0.453	0.382
Lexicalized DMV	0.356	0.305	0.730	0.440	0.381
Lexicalized EVG 1b	0.345	0.361	0.729	0.445	0.381
Randomized Initializer					
DMV	0.530	0.409	0.291	0.561	0.672
DMV-LI	0.525	0.411	0.523	0.564	0.098
EVG-LI	0.513	0.410	0.571	0.562	0.068
Lex. DMV	0.559	0.409	0.690	0.565	0.597
Lex. EVG	0.542	0.426	0.715	0.543	0.606

Table 5.3: Training set directed accuracy across several languages.

	Bulgarian	German	Japanese	Swedish	Turkish
Harmonic Initializer					
DMV-Dirichlet	0.661	0.536	0.777	0.582	0.609
DMV-Linear Interpolation	0.621	0.520	0.809	0.604	0.634
EVG-Linear-Interpolation	0.620	0.528	0.807	0.611	0.634
Lexicalized DMV	0.609	0.520	0.812	0.603	0.633
Lexicalized EVG 1b	0.604	0.528	0.811	0.607	0.633
Randomized Initializer					
DMV	0.648	0.594	0.636	0.660	0.713
DMV-Linear Interpolation	0.655	0.593	0.725	0.661	0.568
EVG-Linear Interpolation	0.650	0.594	0.746	0.663	0.565
Lex. DMV	0.674	0.596	0.792	0.661	0.690
Lex. EVG	0.666	0.607	0.804	0.653	0.685

Table 5.4: Training set Undirected Accuracy across languages.

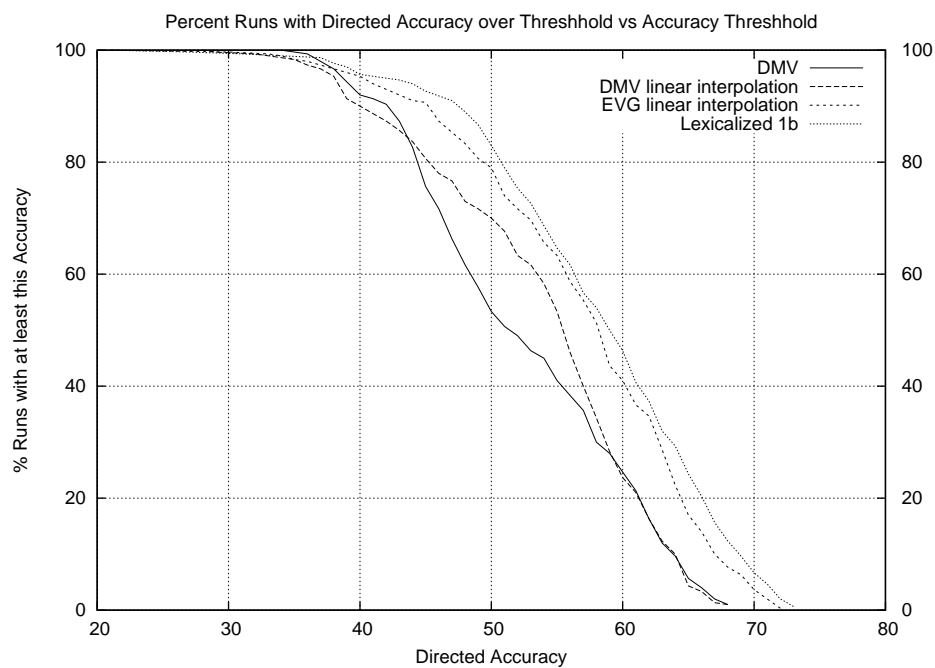


Figure 5.1: % of runs with Directed Accuracy exceeding the threshold given on the x -axis. Each run is the result of has a beam of 20 random-restarts of Variational Bayes, and running the best of these to convergence. Each lexicalized model run however has a beam of 20 smoothed EVG models, from which one is initialized and run to convergence.

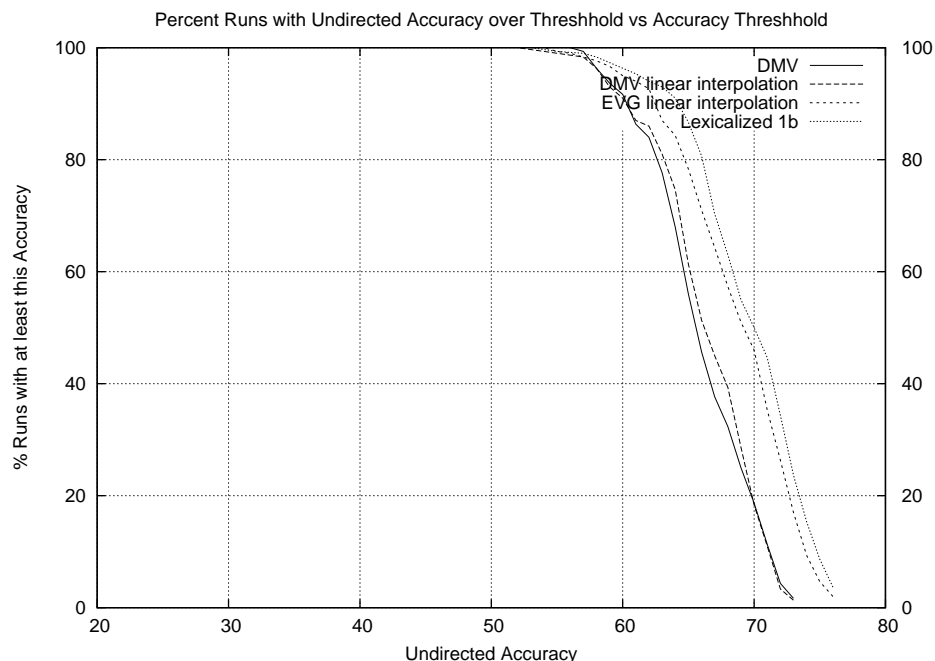


Figure 5.2: % of runs with Undirected Accuracy exceeding the threshold given on the x -axis, using the same setup as Figure 5.1.

comes from selecting the restart with the highest lower bound on the log likelihood.

Figures 5.1 and 5.2 show the results for 4 models in a different way: DMV, DMV smoothed using linear interpolation, EVG smoothed using linear interpolation, and Lexicalized model 1b. For these graphs, we took each “run” consisting of selecting the of 20 random-restarts and running it to convergence. For 300 runs and different accuracy thresholds, we then look at what percentage of the 300 runs has an accuracy at least that high. Thus if one selected a run at random, one could ask what the probability is that its accuracy is above some threshold. Hence better models could be considered those that have a larger percent of high accuracy runs. In Figures 5.1 and 5.2 we show this for Directed and Undirected accuracy on the WSJ10 training set. We can see that the Lexicalized model 1b and smoothed EVG are well to the right of the DMV curves, except at the very top, indicating

an improvement in accuracy across a large number of runs. Smoothed DMV seems to improve over DMV primarily in the percentage of runs with at least 44-59 directed accuracy, while DMV has more runs with lower accuracy.

5.2 Learning from Induced tags

One of the initial goals of this thesis was to learn dependency grammars from words, without the benefit of annotated part-of-speech tags. The approach we proposed was to start from the output of a Hidden Markov Model unsupervised part-of-speech induction system, and to split the given tags, learning them in conjunction with the dependencies. While this approach did not prove fruitful, we would like to describe the experiments conducted here.

A Hidden Markov Model (HMM) describes a generative process for an observed sequence. The sequence of observations $s_i = s_{i1} \dots s_{im_i}$, in our case words, is generated through the following process. First a sequence of hidden states $\tau_{i1} \dots \tau_{im_i}$ is generated, and then for each τ_{ij} a word is emitted, generated conditioned on the state τ_{ij} . In our case the hidden state τ_{ij} will correspond to a part-of-speech for the word s_{ij} .

In a hidden Markov Model each hidden state τ_{ij} is generated conditioned on the previous z states $\tau_{i(j-1)} \dots \tau_{i(j-z)}$. In our experiments z will be 1, corresponding to the bitag model.

$$\begin{aligned} \tau_{ij} | \tau_{i(j-1)} = \sigma_1, \dots, \tau_{i(j-z)} = \sigma_z &\sim \text{Categorical}(\phi_{\sigma_1 \dots \sigma_z}) \\ s_{ij} | \tau_{ij} = \sigma_1 &\sim \text{Categorical}(\chi_{\sigma_1}) \end{aligned}$$

In (III et al., 2008) we explored using several different unsupervised part-of-speech taggers as a first stage for grammar induction. We found that inducing parts-of-speech using an HMM trained using Expectation Maximization was nearly as effective as gold standard

	20 tags		40 tags	
DMV Initialization	Directed Acc.	Undirected Acc.	Directed Acc.	Undirected Acc.
Harmonic	0.412	0.600	0.441	0.621
Randomized	0.288	0.565	0.278	0.554

Table 5.5: DMV on tags induced using bitag HMM, accuracy on the training set.

tags for learning DMV using the Klein and Manning (2004) harmonic initializer.

Unfortunately, when one uses randomized initialization for DMV when run from HMM induced parts-of-speech, this nice result breaks down. For these experiments we trained a bitag HMM¹ using Expectation Maximization on section 2-22, 24 of the Penn Treebank Wall Street Journal, using 20 and 40 tags. We used maximum marginal decoding to extract the most likely tag for each word, as we did in (III et al., 2008). We ran 10 runs of each. The tag output was used as input to DMV. For the harmonic initializer we ran it for each run, and averaged. For the randomized initializer we ran a beam of 20 random restarts for each tagging, selected the one with the highest lower-bound, and then that to convergence. These were also averaged over the 10 taggings.

Table 5.5 shows the results of running DMV on the output of these tags. using the harmonic initializer and using the randomized initialization we presented in Chapter 2. As we can see, the harmonic initializer is nearly as effective as with gold tags, while the randomized initializer does very poorly. Clearly the bias integral to the harmonic initializer is effective for placing the induced tag-DMV into a similar part of the space as it does for gold tags.

We next investigated whether we could possibly combine randomization with the harmonic initializer in a way that could be effective for induced tags. Our approach was to

¹We used an implementation by Mark Johnson, available on his website <http://cog.brown.edu/mj/>.

initialize as in the Harmonic initializer, and then to use a sample from the resulting variational posterior as our initializer.

In particular, we initialize the variational posterior Dirichlets to

$$\hat{\alpha}_r = \alpha_r + E_{q_{\text{har}}(\mathbf{t})}f(\mathbf{t}, r)/D$$

which for $D = 1$ is the usual the harmonic initializer. Then we can then take a sample θ' from this posterior and calculate $E_{p(\mathbf{t}|\theta')}f(\mathbf{t}, r)$ for each rule r in the grammar via the E-step of the Inside-Outside algorithm. We then set $\hat{\alpha}_r = \alpha_r + E_{p(\mathbf{t}|\theta')}f(\mathbf{t}, r)$. This is exactly what we do for our randomized initialization normally, except θ' is in that case sampled from the prior $p(\theta'|\alpha)$.

This process results in an initial $q(\theta)$ which is similar to the one centered on the Harmonic initializer, but with some amount of noise. We can increase the noise by making this initial Dirichlet less peaked by increasing the factor D . We ran DMV using this initialization scheme with $D = 1, 4, 8, 16, 32$ on induced tag input with 20 tags. We use a beam of 20 random restarts for each tagging. The results are shown in Table 5.6. We can see while the randomized initializer with $D = 1$ gives a small improvement over the deterministic harmonic initializer, the result is still not nearly on the order of improvements we saw with randomized initialization for gold standard tags. Furthermore, as we increase D as expected the result gets farther and farther from the harmonic solution; however we do not see an improvement in performance from doing so.

5.2.1 Learning Hidden State Models

In an attempt to learn word classes together with dependencies, we looked at a dependency model in which we learn subclasses of the parts-of-speech. We examined learning subclasses of the given part-of-speech tag input. For induced tags this involves starting with

		20 tags	
DMV Initialization	D	Directed Acc.	Undirected Acc.
Harmonic		0.412	0.600
Rand-Harmonic	1	0.460	0.619
Rand-Harmonic	4	0.421	0.607
Rand-Harmonic	8	0.395	0.601
Rand-Harmonic	16	0.365	0.587
Rand-Harmonic	32	0.314	0.584
Randomized		0.278	0.554

Table 5.6: DMV with randomized Harmonic Initializer

some small number of tags derived from the HMM, and then splitting them into multiple subtags.

This idea is inspired by work in learning to automatically refine PCFGs, by learning grammars whose nonterminals are annotated versions of some initial grammar (see for instance (Liang et al., 2007; Matsuzaki et al., 2005; Dreyer and Eisner, 2006; Petrov et al., 2006; Headden III et al., 2006)). Grammar refinement is a problem that considers the parse-trees to be visible, and the given nonterminals to be sparse versions of the “true” nonterminals. For instance, if A were a nonterminal in one of the parse-trees, we would have a series of latent nonterminals A_1, A_2, A_3, \dots . If the production $A \rightarrow B C$ existed in the original grammar, we would try to learn probabilities for $A_1 \rightarrow B_2 C_1, A_2 \rightarrow B_2 C_2$, etc.

In our case, we have a grammar which is annotated by parts-of-speech. We will try splitting the parts-of-speech in the same way, which will in turn split the CFG rules in our grammar. Another difference is of course that for us the surface trees are not visible.

One important ramification of this is our parsing time will increase a great deal. In the

situation where the part-of-speech tag and word are both observed, we parse using the very small rule schemas presented in Chapter 2, filling in the word and tags as needed to access the relevant statistics. Here, there may be several parts-of-speech for a given word, and so for a given span we need to consider several parts of speech for its left-most and right-most children. If there are T possible parts of speech for each word, this adds a factor of $O(T^2)$ to our parsing time.

We investigated two models. The first is simply DMV, where for each part-of-speech tag $\tau \in V_\tau$ there is a set of D possible annotated part-of-speech tags, which we will denote with a superscript $\{\tau^1, \tau^2, \dots, \tau^D\}$. If one lets $V_{\bar{\tau}}$ denote the set of annotated part-of-speech tags, the model is simply DMV with a part-of-speech set $V_{\bar{\tau}}$.

In the second model we use the annotations to refine the argument distributions, and the distribution over words. The argument distribution predicts the probability of selecting a particular annotated part-of-speech $\bar{A} = \tau^a$ conditioned on the direction and the head annotated part-of-speech $\bar{H} = \tau^b$. We smooth this using linear interpolation with a distribution over annotated part-of-speech arguments \bar{A} conditioned on the direction and the head part-of-speech H , without the annotation. The probability of a word a is conditioned on its annotated part-of-speech \bar{A} . The stop distribution is conditioned on the valence bit v , the direction d and the unannotated part-of-speech A . The backoff scheme is outlined in Table 5.7.

In our experiments we start with DMV run with the Harmonic Initializer for 40 iterations of Variational Bayes. We then initialize the annotated model with the result of that as follows. For the plain DMV with simply more parts-of-speech, we initialize each of the stop and argument word distributions for a given annotated head \bar{H} as the corresponding distribution for unannotated head H . For the argument distributions, we set the variational Dirichlet hyperparameter for a given annotated part-of-speech head, argument, direction to be equal to the variational Dirichlet hyperparameter for the corresponding unannotated

Component		λ conditioning
Stop	$P(s Hvd)$	—
Dependent POS	$P(\bar{A} \bar{H}d)$	$\bar{H}d$
	$P(\bar{A} Hd)$	—
Dependent Word	$P(a \bar{A})$	—

Table 5.7: DMV with annotated parts-of-speech and smoothing backoff chains. The annotated dependent part-of-speech distribution is conditioned on the annotated head part-of-speech, and is smoothed with a distribution conditioned on the unannotated head. The dependent word is conditioned on the annotated part-of-speech.

parts-of-speech in the unannotated DMV, plus a small amount of random noise.

For the second annotated DMV we initialize in the same way, except that we initialize both argument annotated part-of-speech components using the same approach.

We ran each model on the output of 10 runs of the HMM part-of-speech tagger, for each of 20 and 40 tags. The results are given in Table 5.8. As we can see, adding the hidden state information does not help performance, though it does not change very much. The resulting models are still very much constrained by the harmonic initializer. We performed a similar experiment on gold tags, with the harmonic initializer, also shown in Table 5.8. We see a similar dynamic, with adding hidden states to DMV not helping performance, but not by a great degree.

In summary our experiments in learning parts-of-speech in conjunction with dependencies were not very successful. We found that randomized initialization was not effective when the tags are given by the output of a bitag Hidden Markov Model. Reverting to using the harmonic initializer, we proposed two versions of DMV to learn subclasses of the HMM output tags. These approaches did not improve upon using simply the output of the Hidden Markov Model; in contrast the results are very similar.

	States	Gold		20 tags		40 tags	
Model	per tag	D. Acc.	U. Acc.	D. Acc.	U. Acc.	D. Acc.	U. Acc.
DMV	1	0.483	0.651	0.412	0.600	0.441	0.621
DMV HS 1	4	0.468	0.644	0.403	0.592	0.429	0.615
DMV HS 1	8	0.470	0.645	0.403	0.593	0.428	0.612
DMV HS 2	4	0.481	0.652	0.399	0.588	0.428	0.613
DMV HS 2	8	0.479	0.654	0.398	0.587	0.428	0.613

Table 5.8: DMV with hidden states on HMM-induced part-of-speech tags, WSJ10 sections 2-21.

5.3 Conclusions

In this dissertation we have examined the problem of unsupervised learning of syntactic dependency structure. We discussed the ways in which dependency structure has been learned in the past, and how PCFGs may be employed to model dependency grammars. We outlined a particular form of PCFG with tied parameters, and noted how standard PCFG estimation procedures could be employed with them. We used this variety of tied PCFG to introduce smoothing into the PCFG, and explored a variety of smoothing schemes and estimation procedures. We finally explored introducing lexical features into the dependency grammars in various ways, and found that we could get better dependency induction performance through these techniques.

In our exploration of smoothing the schemes we examined linear interpolation. We explored a variety of different priors for the mixture distributions within linear interpolation, as well as exploring both maximum held-out likelihood and Variational Bayes approaches to estimation.

For linear interpolation, while estimating the mixture parameters using EM on a held-out data set was more effective than estimating them together on the training set. However,

estimation using Variational Bayes, placing priors on the mixture parameters was far more effective than either of these approaches. We examined two broad classes of priors, one which biases the mixture distribution to prefer backing off, and one which biases the mixture distribution towards values that place mass on both components. We found that the latter class was generally more effective for dependency learning, although both gave improvement over unsmoothed versions of DMV.

Linear interpolation with a prior that prefers mixtures placing weight on both parameters were the most effective smoothing schemes of those tried.

In our investigation of lexicalization we were able to find improvements from adding lexical conditioning information when predicting the argument part-of-speech. This held true when adding lexicalization to both the plain Dependency Model with Valence and to the smoothed Extended Valence Grammar we developed in Chapter 3. This corresponds to a situation in which different head words with the same part-of-speech have different distributions over argument types. We also examined using head word lexicalization to predict the valence of a head as well as the argument word. These did not turn out to be beneficial under our framework.

Appendix A

Summary of Key Notation

s_i	Words of i th sentence
s_{ij}	j th word of i th sentence
τ_{ij}	Part-of-speech tag associated with s_{ij}
t_i	Parse tree associated with i th sentence
V_w	Set of possible words
V_τ	Set of possible parts-of-speech

Conditioning context variables:

a	Argument word
A	Argument part-of-speech
h	Head word
H	Head part-of-speech
d	Direction
v	Valence position

Appendix B

Split Bilexical PCFGs of Dependency

Models

As mentioned in Chapter 2, one can use the fold and unfold transforms to convert the grammar shown in Table B.1 to the grammar in Table B.2. The aim is to construct a grammar where the nonterminal annotations (for instance H in L_H) refer to the word/pos on the end of that nonterminal’s yield. When this is the case for all nonterminals the annotation becomes redundant when the span is known, which means we can parse with the grammar schema, and fill in the probabilities based on the annotations, thereby giving us an $O(|s_i|^3)$ parsing algorithm for a sentence of length $|s_i|$ (Eisner and Blatz, 2007; Johnson, 2007). For instance, L_H must have an H_L as the rightmost terminal in its yield, so when parsing the H is redundant. Whenever we see an L_H we would look to the rightmost position in its span to determine the probabilities.

In order to do so we must eliminate the Y_H nonterminal, which can contain the terminals H_L, H_R interior to its yield. We do so first by unfolding Y_H into $L_H R_H$ everywhere it appears. This gives trinary rules such as $L_H^2 \rightarrow L_A R_A L'_H$ and $L_H^1 \rightarrow L_A R_A L_H^0$. We then fold $R_A L'_H$ into a new nonterminal ML_{AH}^2 , (likewise $R_A L_H^0$ into ML_{AH}^1) and add

Extended Valence Grammar (EVG)	
Rule	Description
$S \rightarrow Y_H$	select H as root
$Y_H \rightarrow L_H R_H$	Move to split-head representation
$L_H \rightarrow L_H^0$	stop generating arguments left,head = H , no arguments
$L_H \rightarrow L'_H$	continue generating arguments left,head = H no arguments
$L'_H \rightarrow L_H^1$	stop generating arguments left,head = H , one or more arguments
$L'_H \rightarrow L_H^2$	continue generating arguments left,head = H , one or more arguments
$L_H^2 \rightarrow Y_A L'_H$	argument = A left,head = H , argument is not nearest to head
$L_H^1 \rightarrow Y_A L_H^0$	argument = A left,head = H , argument is nearest to head
$L_H^0 \rightarrow H_L$	
$R_H \rightarrow R_H^0$	stop generating arguments right,head = H , no arguments
$R_H \rightarrow R'_H$	continue generating arguments right,head = H no arguments
$R'_H \rightarrow R_H^1$	stop generating arguments right,head = H , one or more arguments
$R'_H \rightarrow R_H^2$	continue generating arguments right,head = H , one or more arguments
$R_H^2 \rightarrow R'_H Y_A$	argument = A right,head = H , argument is not nearest to head
$R_H^1 \rightarrow R_H^0 Y_A$	argument = A right,head = H , argument is not nearest to head
$R_H^0 \rightarrow H_R$	

Table B.1: Basic Schema for EVG

Extended Valence Grammar (EVG)-2	
Rule	Description
$S \rightarrow L_H R_H$	select H as root
$L_H \rightarrow L_H^0$	stop generating arguments left,head = H , no arguments
$L_H \rightarrow L'_H$	continue generating arguments left,head = H no arguments
$L'_H \rightarrow L_H^1$	stop generating arguments left,head = H , one or more arguments
$L'_H \rightarrow L_H^2$	continue generating arguments left,head = H , one or more arguments
$L_H^2 \rightarrow L_A M L_{AH}^2$	argument = A left,head = H , argument is not nearest to head
$L_H^1 \rightarrow L_A M L_{AH}^1$	argument = A left,head = H , argument is nearest to head
$L_H^0 \rightarrow H_L$	
$M L_{AH}^2 \rightarrow R_A L'_H$	
$M L_{AH}^1 \rightarrow R_A L_H^0$	
$R_H \rightarrow R_H^0$	stop generating arguments right,head = H , no arguments
$R_H \rightarrow R'_H$	continue generating arguments right,head = H no arguments
$R'_H \rightarrow R_H^1$	stop generating arguments right,head = H , one or more arguments
$R'_H \rightarrow R_H^2$	continue generating arguments right,head = H , one or more arguments
$R_H^2 \rightarrow M R_{HA}^2 R_A$	argument = A right,head = H , argument is not nearest to head
$R_H^1 \rightarrow M R_{HA}^1 R_A$	argument = A right,head = H , argument is not nearest to head
$M R_{HA}^2 \rightarrow R'_H L_A$	
$M R_{HA}^1 \rightarrow R_H^0 L_A$	
$R_H^0 \rightarrow H_R$	

Table B.2: Schema for EVG after fold-unfold transform

the rules $ML_{AH}^2 \rightarrow R_A L'_H$ and $ML_{AH}^1 \rightarrow R_A L_H^0$. We perform analogous operations for the right argument distributions. The resulting schema is given in Table B.2.

Appendix C

Calculating Confidence Bounds using Bootstrap Sampling

The initialization scheme we adopt in Section 2.4.1 uses random restarting to search different parts of the parameter space. This starts with a total of 6000 restarts grouped into 300 jobs, 20 restarts per job. In each job each restart is run for 40 iterations, and the one with the lowest free energy is run until convergence. Thus each job is associated with one set of parameters, and a free energy.

Each of those 300 jobs is broken in to 10 groups of 30; in each group, we select the job whose free energy is the lowest. This gives us 10 sets of parameters; we parse with these and score the resulting dependency trees in terms of Directed Accuracy and Undirected Accuracy. The results presented are an average of these scores.

In order to calculate confidence intervals for this statistic we use bootstrap sampling (Hastie et al., 2009, pp. 261-264). We make 10,000 samples with replacement of size 300 from the 300 jobs. For each sample, we split the 300 jobs in the sample into 10 groups of 30; select the job in each group whose free energy is the lowest, giving us 10 sets of parameters, for which we calculate Directed and Undirected accuracy and average over the

10. This gives us 10,000 values for directed and undirected accuracy. Sorting the directed accuracy, the 250th and 9750th values will give us the lower and upper bounds.

Appendix D

Smoothing with Collapsed Interpolation

One simple smoothing approach to use in a tied PCFG framework which we explored in our initial models is what we call *collapsed interpolation*. The basic idea is, if we are interesting in smoothing $P(E|C)$, to augment the space of possible events with an event b that indicates that we should backoff. In this case an event is selected from a distribution $P_2(E|\bar{C})$ that makes an independence assumption about the conditioning information $C' \subset C$.

In our example of selecting a left argument given a head NN we would smooth the probability that the argument is a DT as:

$$P(A = \text{DT} \mid d = \text{left}, H = \text{NN}) = P_1(A = \text{DT} \mid d = \text{left}, H = \text{NN}) + P_1(A = b \mid d = \text{left}, H = \text{NN})P_2(A = \text{DT} \mid d = \text{left})$$

Suppose we are smoothing the distribution over rules $r \in R_A$ for left-hand-side non-terminal A , and $B \subseteq N$ is the relevant backoff set. Again let $\mathcal{R}(R_A)$ indicate the set of right-hand sides of rules in R_A . For each $\hat{A} \in B$ we add a new nonterminal \hat{A}^b to N and add an additional set of rules $\{\hat{A}^b \rightarrow \beta \mid \beta \in \mathcal{R}(R_{\hat{A}})\}$ to R . $R_{\hat{A}}$ is then augmented with a rule $\hat{A} \rightarrow \hat{A}^b$. The newly added nonterminals \hat{A}^b need their associated rules tied together

by defining their tying equivalence relations. As with linear interpolation, tying these rules gives them the same distribution, effectively forgetting some conditioning information.

A key thing to note is that \hat{A}^b expands to the same set of right-hand-sides as \hat{A} , save for the addition of $\hat{A} \rightarrow \hat{A}^b$. This means that \hat{A} will eventually end up expanding to one of those right-hand sides, even if the indirect route through \hat{A}^b is chosen. For example, consider Figure 8. This figure presents two ways of rewriting $L_{dog}^1 \rightarrow Y_{big}L_{dog}^0$. The left-hand path simply rewrites it directly, while the right-hand path first rewrites to L_{dog}^{1b} , which then rewrites to $Y_{big}L_{dog}^0$. Observe that we can model the probability $p(L_{dog}^1 \implies Y_{big}L_{dog}^0 | dog, v, left)$, where $A \implies \beta$ means A eventually rewrites to β , as:

$$p(L_{dog}^1 \implies Y_{big}L_{dog}^0 | dog, v, left) = \theta_{L_{dog}^1 \rightarrow Y_{big}L_{dog}^0} + \theta_{L_{dog}^1 \rightarrow L_{dog}^{1b}} \theta_{L_{dog}^{1b} \rightarrow Y_{big}L_{dog}^0}$$

As an example of this applied to a particular case, consider the case where we want to smooth the distribution over the right-hand side of L_H^1 in EVG. The backoff set in this example is $B = \{L_H^1 \mid H \in V_\tau\}$. That is, we would smooth with a distribution that ignores the head when generating the first left argument. We add L_H^{1b} for each $H \in V_\tau$, and $L_H^{1b} \rightarrow Y_A L_H^0$ for each $H, A \in V_\tau$. Finally, we need to specify that for each $H, H', A \in V_\tau$ that $L_H^{1b} \rightarrow Y_A L_H^0 \stackrel{GR}{=} L_{H'}^{1b} \rightarrow Y_A L_{H'}^0$, meaning that we tie together rules based on whether they generate the same argument.

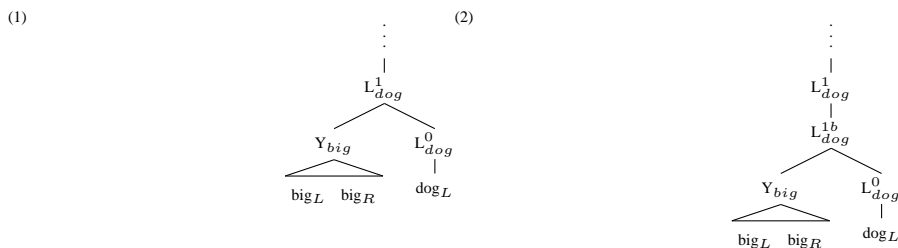


Figure D.1: Using collapsed interpolation to smooth $L_{dog}^1 \rightarrow Y_{big}L_{dog}^0$: note that from L_{dog}^1 there are two paths to the children $Y_{big} L_{dog}^0$: (1) directly and (2) indirectly through L_{dog}^{1b} . The distribution over different arguments given L_{dog}^{1b} is tied

D.0.1 DMV and EVG with Collapsed Interpolation

Our first experiments examine smoothing the distribution over an argument A given head part-of-speech H , direction d , in DMV. We do this by letting the backoff distribution ignore the head-part-of-speech H . Our experiments use Variational Bayes for estimation. We use Dirichlet priors for rule probabilities. We set the Dirichlet hyperparameter $\alpha = 1$ for all rules in the grammar, except for those rules corresponding to deciding to use the backoff distribution. For these rules, $\alpha = 2|V_\tau|$ (i.e. twice the number of parts-of-speech). This places a strong bias towards the backoff distribution, effectively giving each argument in the backoff distribution a pseudocount of 2, while each fully conditioned argument gets a pseudocount of 1.

The analogue to this smoothing scheme EVG again has the backoff distribution ignore the head part-of-speech H and use backoff conditioning event v, d . This would include a notion of how common the arguments are across heads. For our running example smoothing the distribution over rules for L_{NN}^1 , this would give a backoff set $\{L_H^1 \mid H \in N\}$.

Results for both these models are given in table D.1. As we can see adding smoothing even in the unlexicalized case gives a major improvement for both DMV and EVG.

		Directed Acc		Undirected Acc.	
Model	Smoothing	Train	Dev	Train	Dev
DMV	None	0.583 $\left(\begin{smallmatrix} +0.064 \\ -0.034 \end{smallmatrix}\right)$	0.549 $\left(\begin{smallmatrix} +0.065 \\ -0.031 \end{smallmatrix}\right)$	0.689 $\left(\begin{smallmatrix} +0.025 \\ -0.014 \end{smallmatrix}\right)$	0.668 $\left(\begin{smallmatrix} +0.027 \\ -0.013 \end{smallmatrix}\right)$
	Col. Interp.	0.623	0.581	0.703	0.676
EVG	None	0.526 $\left(\begin{smallmatrix} +0.017 \\ -0.080 \end{smallmatrix}\right)$	0.500 $\left(\begin{smallmatrix} +0.013 \\ -0.080 \end{smallmatrix}\right)$	0.679 $\left(\begin{smallmatrix} +0.008 \\ -0.028 \end{smallmatrix}\right)$	0.657 $\left(\begin{smallmatrix} +0.006 \\ -0.032 \end{smallmatrix}\right)$
	Col. Interp	0.659	0.632	0.734	0.713

Table D.1: Results from Smoothing DMV, EVG using Collapsed Interpolation

Bibliography

J.K. Baker. 1979. Trainable grammars for speech recognition. In *Speech Communication Papers presented at the 97th Meeting of the Acoustical Society of America*, pages 547–550, MIT, Cambridge, Massachusetts.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless Unsupervised Learning with Features. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL10)*.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10*, pages 1204–1213. Association for Computational Linguistics.

Samuel Brody. 2010. It depends on the translation: Unsupervised dependency parsing via word alignment. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1222, Cambridge, MA, October. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X '06*, pages 149–164. Association for Computational

Linguistics.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–13. AAAI.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the ACL 2000*, pages 132–139.

Stanley F. Chen. 1996. *Building Probabilistic Models for Natural Language*. Ph.D. thesis, Harvard University.

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL-HLT 2009*.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems 21*.

S. B. Cohen, D. M. Blei, and N. A. Smith. 2010. Variational inference for adaptor grammars. In *Proceedings of NAACL*.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, ACL '98*, pages 16–23. Association for Computational Linguistics.

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

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *International Conference on Machine Learning*, Montreal, Canada.

Markus Dreyer and Jason Eisner. 2006. Better informed training of latent syntactic features. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 317–326, Sydney, Australia. Association for Computational Linguistics.

Jason Eisner and John Blatz. 2007. Program transformations for optimization of parsing algorithms and other weighted logic programs. In Shuly Wintner, editor, *Proceedings of FG 2006: The 11th Conference on Formal Grammar*, pages 45–85. CSLI Publications.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, University of Maryland, June.

Jason Eisner. 1996. An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, Univ. of Pennsylvania.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2011. Posterior sparsity in unsupervised dependency parsing. *Journal of Machine Learning Research*, 12:455–490.

João Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA.

David Graff. 2003. *English Gigaword*. Linguistic Data Consortium.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer, 2nd edition.

William P. Headden III, Eugene Charniak, and Mark Johnson. 2006. Learning phrasal categories. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 301–307, Sydney, Australia. Association for Computational Linguistics.

William P. Headden III, David McClosky, and Eugene Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, Manchester, UK.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, Boulder, Colorado.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *NIPS*, pages 641–648.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of the North American Conference on Computational Linguistics (NAACL'07)*.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of the Association for Computational Linguistics 2007*.

Dan Klein and Christopher Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL 2002*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 2004*, pages 478–485, Barcelona, Spain.

Kenichi Kurihara and Taisuke Sato. 2004. An application of the variational bayesian approach to probabilistic context-free grammars. In *IJCNLP 2004 Workshop Beyond Shallow Analyses*.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697.

M. Marcus et al. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comp. Linguistics*, 19(2):313–330.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 2005 Meeting of the Association for Computational Linguistics*.

David McAllester. 1999. A reformulation of Eisner and Sata's cubic time parser for split head automata grammars. Available from <http://ttic.uchicago.edu/~dmcallester/>.

David McClosky. 2008. Modeling valence effects in unsupervised grammar induction. Technical Report CS-09-01, Brown University, Providence, RI, USA.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10*, pages 1234–1244. Association for Computational Linguistics.

Mark A. Paskin. 2001. Grammatical bigrams. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, Cambridge, MA. MIT Press.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Grammatical Inference Applications*, pages 73–82, Edinburgh.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL)*, pages 569–576, Sydney.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, October.

Valentin I. Spitzkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2009. Baby Steps: How “Less is More” in unsupervised dependency parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*.

Valentin I. Spitzkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From Baby Steps to Leapfrog: How “Less is More” in unsupervised dependency parsing. In *Proc. of NAACL-HLT*.

Valentin I. Spitzkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *In Proceedings of the International Workshop on Parsing Technologies (IWPT)*.

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