Syntactic Decision Tree LMs: Random Selection or Intelligent Design?

Denis Filimonov†‡
‡Human Language Technology
Center of Excellence
Johns Hopkins University
den@cs.umd.edu

Mary Harper†
†Department of Computer Science
University of Maryland, College Park
mharper@umd.edu

Abstract

Decision trees have been applied to a variety of NLP tasks, including language modeling, for their ability to handle a variety of attributes and sparse context space. Moreover, forests (collections of decision trees) have been shown to substantially outperform individual decision trees. In this work, we investigate methods for combining trees in a forest, as well as methods for diversifying trees for the task of syntactic language modeling. We show that our tree interpolation technique outperforms the standard method used in the literature, and that, on this particular task, restricting tree contexts in a principled way produces smaller and better forests, with the best achieving an 8% relative reduction in Word Error Rate over an n-gram baseline.

1 Introduction

Language Models (LMs) are an essential part of NLP applications that require selection of the most fluent word sequence among multiple hypotheses. The most prominent applications include Automatic Speech Recognition (ASR) and Machine Translation (MT).

Statistical LMs formulate the problem as the computation of the model's probability to generate the word sequence $w_1, w_2, \ldots, w_m$ (denoted as w_1^m), assuming that higher probability corresponds to more fluent hypotheses. LMs are often represented in the following generative form:

$$p(w_1^m) = \prod_{i=1}^{m} p(w_i|w_1^{i-1})$$

Note the context space for this function, w_1^{i-1} is arbitrarily long, necessitating some independence assumption, which usually consists of reducing the relevant context to $n-1$ immediately preceding tokens:

$$p(w_i|w_1^{i-1}) \approx p(w_i|w_{i-n+1}^{i-1}) \quad (1)$$

These distributions are typically estimated from observed counts of n-grams w_{i-n+1}^{i} in the training data. The context space is still far too large[1]; therefore, the models are recursively smoothed using lower order distributions. For instance, in a widely used n-gram LM, the probabilities are estimated as follows:

$$\tilde{p}(w_i|w_{i-n+1}^{i-1}) = \rho(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1}) \cdot \tilde{p}(w_i|w_{i-n+2}^{i-1}) \quad (2)$$

where ρ is a *discounted* probability[2].

Note that this type of model is a simple Markov chain lacking any notion of syntax. It is widely accepted that languages do have some structure. Moreover, it has been shown that incorporating syntax into a language model can improve its performance (Bangalore, 1996; Heeman, 1998; Chelba and Jelinek, 2000; Filimonov and Harper, 2009). A straightforward way of incorporating syntax into a language model is by assigning a *tag* to each word and modeling them *jointly*; then to obtain the proba-

[1] $O(|V|^{n-1})$ in n-gram model with typical order $n = 3 \ldots 5$, and a vocabulary size of $|V| = 10^4 \ldots 10^6$.

[2] We refer the reader to (Chen and Goodman, 1996) for a survey of the discounting methods for n-gram models.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 691–699,
Edinburgh, Scotland, UK, July 27–31, 2011. ©2011 Association for Computational Linguistics

bility of a word sequence, the tags must be marginalized out:

$$p(w_1^m) = \sum_{t_1 \ldots t_m} p(w_1^m t_1^m) = \sum_{t_1 \ldots t_m} \prod_{i=1}^{m} p(w_i t_i | w_1^{i-1} t_1^{i-1})$$

An independence assumption similar to Eq. 1 can be made:

$$p(w_i t_i | w_1^{i-1} t_1^{i-1}) \approx p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \quad (3)$$

A primary goal of our research is to build strong syntactic language models and provide effective methods for constructing them to the research community. Note that the tags in the context of the joint model in Eq. 3 exacerbate the already sparse problem in Eq. 1, which makes the probability estimation particularly challenging. We utilize decision trees for joint syntactic language models to cluster context because of their strengths (reliance on information theoretic metrics to cluster context in the face of extreme sparsity and the ability to incorporate attributes of different types[3]), and at the same time, unlike log-linear models (Rosenfeld et al., 1994), computationally expensive probability normalization does not have to be postponed until runtime.

In Section 2, we describe the details of the syntactic decision tree LM. Construction of a single-tree model is difficult due to the inevitable greediness of the tree construction process and its tendency to overfit the data. This problem is often addressed by interpolating with lower order decision trees. In Section 3, we point out the inappropriateness of backoff methods borrowed from n-gram models for decision tree LMs and briefly describe a generalized interpolation for such models. The generalized interpolation method allows the addition of any number of trees to the model, and thus raises the question: what is the best way to create diverse decision trees so that their combination results in a stronger model, while at the same time keeping the total number of trees in the model relatively low for computational practicality. In Section 4, we explore and evaluate a variety of methods for creating different trees. To support our findings, we evaluate several of the models on an ASR rescoring task in Section 5. Finally, we discuss our findings in Section 6.

[3]For example, morphological features can be very helpful for modeling highly inflectional languages (Bilmes and Kirchhoff, 2003).

2 Joint Syntactic Decision Tree LM

A decision tree provides us with a clustering function $\Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \rightarrow \{\Phi^1, \ldots, \Phi^N\}$, where N is the number of clusters, and clusters Φ^k are disjoint subsets of the context space. The probability estimation for a joint decision tree model is approximated as follows:

$$p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \approx p(w_i t_i | \Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1})) \quad (4)$$

In the remainder of this section, we briefly describe the techniques that we use to construct such a decision tree Φ and to estimate the probability distribution for the joint model in Eq. 4.

2.1 Decision Tree Construction

We use recursive partitioning to grow decision trees. In this approach, a number of alternative binary splits of the training data associated with a node are evaluated using some *metric*, the best split is chosen, checked against a *stopping rule* (which aims at preventing overfitting to the training data and usually involves a heldout set), and then the two partitions become the child nodes if the stopping rule does not apply. Then the algorithm proceeds recursively into the newly constructed leaves.

Binary splits are often referred to as *questions* about the context because a binary partition can be represented by a binary function that decides whether an element of context space belongs to one partition or the other. We utilize univariate questions where each question partitions the context on one attribute, e.g., w_{i-2} or t_{i-1}. The questions about words and tags are constructed differently:

- The questions q about the words are in the form $q(x) \equiv w_{i+x} \in S$, where x is an integer between $-n+1$ and -1, and $S \subset V$ is a subset of the word vocabulary V. To construct the set S, we take the set of words S_o *observed* at the offset x in the training data associated with the

current node and split it into two complementary subsets $S \cup \bar{S} = S_o$ using the Exchange algorithm (Martin et al., 1998). Because the algorithm is greedy and depends on the initialization, we construct 4 questions per word position using different random initializations of the Exchange algorithm.

Since we need to account for words that were not observed in the training data, we utilize the structure depicted in Figure 1. To estimate the probability at the backoff node (B in Figure 1), we can either use the probability from its grandparent node A or estimate it using a lower order tree (see Section 3), or combine the two. We have observed no noticeable difference between these methods, which suggests that only a small fraction of probability is estimated from these nodes; therefore, for simplicity, we use the probability estimated at the backoff node's grandparent.

- To create questions about tags we create a hierarchical clustering of all tags in the form of a binary tree. This is done beforehand, using the Minimum Discriminating Information algorithm (Zitouni, 2007) with the entire training data set. In this tree, each leaf is an individual tag and each internal node is associated with the subset of tags that the node dominates. Questions about tags are constructed in the form $q(x, k) \equiv t_{i+x} \in T_k$, where k is a node in the tag tree and T_k is the subset of tags associated with that node. The rationale behind constructing tag questions in this form is that it enables a more efficient decoding algorithm than standard HMM decoding (Filimonov and Harper, 2009).

Questions are evaluated in two steps. First the context attribute x is selected using a metric similar to information gain ratio proposed by (Quinlan, 1986):

$$\mathcal{M} = 1 - \frac{H(w_i) - H(w_i|x)}{H(x)} = 1 - \frac{I(x; w_i)}{H(x)}$$

where x is one of the context attributes, e.g., w_{i-2} or t_{i-1}. Then, among the questions about attribute x, we select the question that maximizes the entropy reduction.

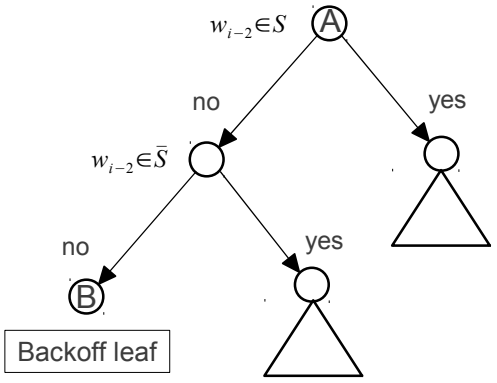

Figure 1: A fragment of the decision tree with a backoff node. $S \cup \bar{S}$ is the set of words observed in the training data at the node A. To account for unseen words, we add the backoff node B.

Instead of dedicating an explicit heldout set for the stopping criterion, we utilize a technique similar to cross validation: the training data set is partitioned into four folds, and the best question is required to reduce entropy on each of the folds.

Note that the tree induction algorithm can also be used to construct trees without tags:

$$p(w_i|w_{i-n+1}^{i-1}) \approx p(w_i|\Phi(w_{i-n+1}^{i-1}))$$

We refer to this model as the *word-tree* model. By comparing syntactic and word-tree models, we are able to separate the effects of decision tree modeling and syntactic information on language modeling by comparing both models to an n-gram baseline.

2.2 In-tree Smoothing

A decision tree offers a hierarchy of clusterings that can be exploited for smoothing. We can interpolate the observed distributions at leaves recursively with their parents, as in (Bahl et al., 1990; Heeman, 1998):

$$\tilde{p}_k(w_i t_i) = \lambda_k p_{ML}(w_i t_i) + (1 - \lambda_k)\tilde{p}_{k'}(w_i t_i) \quad (5)$$

where p_{ML} is the observed distribution at node k and k' is the parent of k. The coefficients λ_k are estimated using an EM algorithm.

We can also combine $p(w_i t_i|\Phi(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}))$ with lower order decision trees, i.e.,

$p(w_i t_i|\Phi(w_{i-n+2}^{i-1} t_{i-n+2}^{i-1}))$, and so on up until $p(w_i t_i)$ which is a one-node tree (essentially a unigram model). Although superficially similar to backoff in n-gram models, lower order decision trees differ substantially from lower order n-gram models and require different interpolation methods. In the next section, we discuss this difference and present a generalized interpolation that is more suitable for combining decision tree models.

3 Interpolation with Backoff Tree Models

In this section, for simplicity of presentation, we focus on the equations for word models, but the same equations apply equally to joint models (Eq. 3) with trivial transformations.

3.1 Backoff Property

Let us rewrite the interpolation Eq. 2 in a more generic way:

$$\tilde{p}(w_i|w_1^{i-1}) = \rho_n(w_i|\Phi_n(w_1^{i-1})) + \gamma(\Phi_n(w_1^{i-1})) \cdot \tilde{p}(w_i|BO_{n-1}(w_1^{i-1})) \quad (6)$$

where, ρ_n is a *discounted* distribution, Φ_n is a clustering function of order n, and $\gamma(\Phi_n(w_1^{i-1}))$ is the backoff weight chosen to normalize the distribution. BO_{n-1} is the *backoff* clustering function of order $n-1$, representing a reduction of context size. In the case of an n-gram model, $\Phi_n(w_1^{i-1})$ is the set of word sequences where the last $n-1$ words are w_{i-n+1}^{i-1}. Similarly, $BO_{n-1}(w_1^{i-1})$ is the set of sequences ending with w_{i-n+2}^{i-1}. In the case of a decision tree model, the same backoff function is typically used, but the clustering function can be arbitrary.

The intuition behind Eq. 6 is that the backoff context $BO_{n-1}(w_1^{i-1})$ allows for a more robust (but less informed) probability estimation than the context cluster $\Phi_n(w_1^{i-1})$. More precisely:

$$\forall_{w_1^{i-1},W} : W \in \Phi_n(w_1^{i-1}) \Rightarrow W \in BO_{n-1}(w_1^{i-1}) \quad (7)$$

that is, every word sequence W that belongs to a context cluster $\Phi_n(w_1^{i-1})$, belongs to the same backoff cluster $BO_{n-1}(w_1^{i-1})$ (hence has the same backoff distribution). For n-gram models, Property 7 trivially holds since $BO_{n-1}(w_1^{i-1})$ and $\Phi_n(w_1^{i-1})$ are defined as sets of sequences ending with w_{i-n+2}^{i-1} and w_{i-n+1}^{i-1}, with the former clearly being a superset of the latter. However, when Φ can be arbitrary, e.g., a decision tree, the property is not necessarily satisfied. Figure 2 illustrates cases when the Property 7 is satisfied (a) and violated (b).

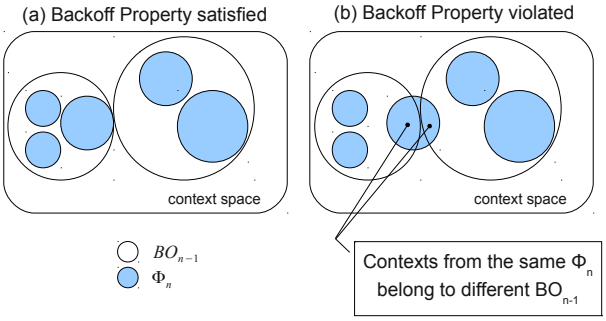

Figure 2: Backoff Property

Let us consider what happens when we have two context sequences W and W' that belong to the same cluster $\Phi_n(W) = \Phi_n(W')$ but different backoff clusters $BO_{n-1}(W) \neq BO_{n-1}(W')$. For example: suppose we have $\Phi(w_{i-2}w_{i-1}) = (\{\text{on}\}, \{\text{may,june}\})$ and two corresponding backoff clusters: $BO' = (\{\text{may}\})$ and $BO'' = (\{\text{june}\})$. Following *on*, the word *may* is likely to be a month rather than a modal verb, although the latter is more frequent and will dominate in BO'. Therefore we have much less faith in $\tilde{p}(w_i|BO')$ than in $\tilde{p}(w_i|BO'')$ and would like a much smaller weight γ assigned to BO'. However this would not be possible in the backoff scheme in Eq. 6, thus we will have to settle on a compromise value of γ, resulting in suboptimal performance.

Hence arbitrary clustering (an advantage of decision trees) leads to a violation of Property 7, which is likely to produce a degradation in performance *if* backoff interpolation Eq. 6 is used.

3.2 Generalized Interpolation

Recursive linear interpolation similar to Jelinek-Mercer smoothing for n-gram models (Jelinek and Mercer, 1980) has been applied to decision tree models:

$$\tilde{p}_n(w_i|w_{i-n+1}^{i-1}) = \lambda_n(\phi_n) \cdot p_n(w_i|\phi_n) + (1 - \lambda_n(\phi_n)) \cdot \tilde{p}_{n-1}(w_i|w_{i-n+2}^{i-1}) \quad (8)$$

where $\phi_n \equiv \Phi_n(w_{i-n+1}^{i-1})$, and $\lambda_n(\phi_n) \in [0, 1]$ are assigned to each cluster and are optimized on a held-out set using EM. $p_n(w_i|\phi_n)$ is the probability distribution at the cluster ϕ_n in the tree of order n. This interpolation method is particularly useful as, unlike count-based discounting methods (e.g., Kneser-Ney), it can be applied to already smoothed distributions p_n.

In (Filimonov and Harper, 2011), we observed that because of the violation of Property 7 in decision tree models, the interpolation method of Eq. 8 is not appropriate for such models. Instead we proposed the following generalized form of linear interpolation:

$$\tilde{p}_n(w_i|w_{i-n+1}^{i-1}) = \frac{\sum_{m=1}^{n} \lambda_m(\phi_m) \cdot p_m(w_i|\phi_m)}{\sum_{m=1}^{n} \lambda_m(\phi_m)} \quad (9)$$

Note that the recursive interpolation of Eq. 8 can be represented in this form with the additional constraint $\sum_{m=1}^{n} \lambda_m(\phi_m) = 1$, which is not required in the generalized interpolation of Eq. 9; thus, the generalized interpolation, albeit having the same number of parameters, has more degrees of freedom. We also showed that the recursive interpolation Eq. 8 is a special case of Eq. 9 that occurs when the Property 7 holds.

4 From Backoff Trees to Forest

Note that, in Eq. 9, individual trees do not have explicit higher-lower order relations, they are treated as a collection of trees, i.e., as a forest. Naturally, to benefit from the forest model, its trees must differ in *some* way. Different trees can be created based on differences in the training data, differences in the tree growing algorithm, or some non-determinism in the way the trees are constructed.

(Xu, 2005) used randomization techniques to produce a large forest of decision trees that were combined as follows:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{1}{M} \sum_{m=1}^{M} p_m(w_i|w_{i-n+1}^{i-1}) \quad (10)$$

where M is the number of decision trees in the forest (he proposed $M = 100$) and p_m is the m-th tree model[4]. Note that this type of interpolation assumes that each tree model is "equal" a priori and therefore is only appropriate when the tree models are grown in the same way (particularly, using the same order of context). Note that Eq. 10 is a special case of Eq. 9 when all parameters λ are equal.

(Xu, 2005) showed that, although each individual tree is a fairly weak model, their combination outperforms the n-gram baseline substantially. However, we find this approach impractical for online application of any sizable model: In our experiments, fourgram trees have approximately 1.8 million leaves and the tree structure itself (without probabilities) occupies nearly 200MB of disk space after compression. It would be infeasible to apply a model consisting of more than a handful of such trees without distributed computing of some sort. Therefore, we pose the following question: If we can afford to have only a handful of trees in the model, what would be best approach to construct those trees?

In the remainder of this section, we will describe the experimental setup, discuss and evaluate different ways of building decision tree forests for language modeling, and compare combination methods based on Eq. 9 and Eq. 10 (when Eq. 10 is applicable).

4.1 Experimental Setup

To train our models we use 35M words of WSJ 94-96 from LDC2008T13. The text was converted into speech-like form, namely numbers and abbreviations were verbalized, text was downcased, punctuation was removed, and contractions and possessives were joined with the previous word (i.e., *they 'll* becomes *they'll*). For the syntactic modeling, we used tags comprised of the POS tags of the word and it's head. Parsing of the text for tag extraction occurred after verbalization of numbers and abbreviations but

[4]Note that (Xu, 2005) used lower order models to estimate p_m.

before any further processing; we used a latent variable PCFG parser as in (Huang and Harper, 2009). For reference, we include an n-gram model with modified interpolated KN discounting. All models use the same vocabulary of approximately 50k words.

Perplexity numbers reported in Tables 1, 2, 3, and 4 are computed on WSJ section 23 (tokenized in the same way)[5].

In Table 1, we show results reported in (Filimonov and Harper, 2011), which we use as the baseline for further experiments. We constructed two sets of decision trees (a joint syntactic model and a word-tree model) as described in Section 2. Each set was comprised of a fourgram tree with backoff trigram, bigram, and unigram trees. We combined these trees using either Eq. 8 or Eq. 9. The λ parameters in Eq. 8 were estimated using EM by maximizing likelihood of a heldout set (we utilized 4-way cross-validation); whereas, the parameters in Eq. 9 were estimated using L-BFGS because the denominator in Eq. 9 makes the maximization step problematic.

4.2 Random Forest

(Xu, 2005) evaluated a variety of randomization techniques that can be used to build trees. He used a word-only model, with questions constructed using the Exchange algorithm, similar to our model. He tried two methods of randomization: selecting the positions in the history for question construction by a Bernoulli trials[6], and random initialization of the Exchange algorithm. He found that when the Exchange algorithm was initialized randomly, the Bernoulli trial parameter did not matter; however, when the Exchange algorithm was initialized deterministically; lower values for the Bernoulli trial parameter r yielded better overall forest performance. We implemented a similar method, namely, initializing the Exchange algorithm randomly and using $r = 0.1$ for Bernoulli trials[7].

There is a key difference between the two randomization methods. Since we do not have an a priori preference for choosing initializations for the Exchange algorithm, by using random initializations it is *hoped* that due to the greedy nature of the algorithm, the constructed trees, while being "undegraded,"[8] will be sufficiently different so that their combination improves over an individual tree. By introducing Bernoulli trials, on the other hand, there is a choice to purposely *degrade* the quality of individual trees in the hope that additional diversity would enable their combination to compensate for the loss of quality in individual trees.

Another way of introducing randomness to the tree construction without apparent degradation of individual tree quality is through varying the data, e.g., using different folds of the training data (see Section 2.1).

Let us take a closer look at the effect of different types of randomization on individual trees and their combinations. In the first set of experiments, we compare the performance of a single undegraded fourgram tree[9] with forests of fourgram trees grown randomly with Bernoulli trials. Having only same-order trees in a forest allows us to apply interpolation of Eq. 10 (used in (Xu, 2005)) and compare with the interpolation method presented in Eq. 9. By comparing forests of different sizes with the baseline from Table 1, we are able to evaluate the effect of randomization in decision tree growing and assess the importance of the lower order trees.

The results are shown in Table 2. Note that, while an undegraded syntactic tree is better than the word tree, the situation is reversed when the trees are grown randomly. This can be explained by the fact that the joint model has a much higher dimensionality of the context space, and therefore is much more sensitive to the clustering method.

As we increase the number of random trees in the forest, the perplexity decreases as expected, with the interpolation method of Eq. 9 showing improvement of a few percentile points over Eq. 10. Note that in the case of the word-tree model, it takes 4 random decision trees to reach the performance of a single undegraded tree, while in the joint model, even

[5]This section was not used for training the parser or for the LM training.

[6]In this method, positions in the history are ignored with probability $1 - r$, where r is the Bernoulli trials parameter.

[7]Note that because in the joint model, the question about tags are deterministic, we use a lower value of r than (Xu, 2005) to increase randomness.

[8]Here and henceforth, by "undegraded" we mean "according to the algorithm described in Section 2."

[9]Since each tree has a smooth distribution based on Eq. 5, lower order trees are not strictly required.

		Eq. 8		Eq. 9 (generalized)	
order	n-gram	word-tree	syntactic	word-tree	syntactic
2-gram	261.0	257.8	214.3	258.1	214.6
3-gram	174.3 (33.2%)	168.7 (34.6%)	156.8 (26.8%)	168.4 (34.8%)	155.3 (27.6%)
4-gram	161.7 (7.2%)	164.0 (2.8%)	156.5 (0.2%)	155.7 (7.5%)	147.1 (5.3%)

Table 1: Perplexity results on PTB WSJ section 23. Percentage numbers in parentheses denote the reduction of perplexity relative to the lower order model of the same type.

	word-tree		syntactic	
	Eq. 10	Eq. 9	Eq. 10	Eq. 9
1 × undgr	204.9		189.1	
1 × rnd	250.2		289.9	
2 × rnd	229.5	221.5	244.6	240.9
3 × rnd	227.5	214.5	226.2	220.0
4 × rnd	219.5	205.0	219.5	212.2
5 × rnd	200.9	184.1	216.5	209.0
baseline	N/A	155.7	N/A	147.1

Table 2: Perplexity numbers obtained using fourgram trees only. Note that "undgr" and "rnd" denote undegraded and randomly grown trees with Bernoulli trials, respectively, and the number indicates the number of trees in the forest. Also "baseline" refers to the fourgram models with lower order trees (from Table 1, Eq. 9).

	word-tree		syntactic	
# trees	Exchng.	+data	Exchng.	+data
1	204.9		189.1	
2	185.9	186.5	174.5	173.7
3	179.5	179.9	168.8	167.2
4	176.2	176.4	165.1	164.0
5	173.7	172.0	163.0	162.0
baseline	155.7		147.1	

Table 3: Perplexity numbers obtained using fourgram trees produced using random initialization of the Exchange algorithm (Exchng. columns) and, additionally, variations in training data folds (+data columns). Note that "baseline" refers to the fourgram models with lower order trees (from Table 1). All models use the interpolation method of Eq. 9.

5 trees are much worse than a single decision tree constructed without randomization. Finally, compare the performance of single undegraded fourgram trees in Table 2 with fourgram models in Table 1, which are constructed with lower order trees: both word-tree and joint models in Table 1 have over 20% lower perplexity compared to the corresponding models consisting of a single fourgram tree.

In Table 3, we evaluate forests of fourgram trees produced using randomizations without degrading the tree construction algorithm. That is, we use random initializations of the Exchange algorithm and, additionally, variations in the training data fold. All forests in this table use the interpolation method of Eq. 9. Note that, while these perplexity numbers are substantially better than trees produced with Bernoulli trials in Table 2, they are still significantly worse than the baseline model from Table 1.

These results suggest that, while it is beneficial to combine *different* decision trees, we should introduce differences to the tree construction process without degrading the trees when introducing randomness, especially for joint models. In addition, lower order trees seem to play an important role for high quality model combination.

4.3 Context-Restricted Forest

As we have mentioned above, combining higher and lower order decision trees produces much better results. A lower order decision tree is grown from a lower order context space, i.e., the context space where we purposely ignore some attributes. Note that in this case, rather than *randomly* ignoring contexts via Bernoulli trials at every node in the decision tree, we discard some context attributes upfront in a principled manner (i.e., most distant context) and then grow the decision tree without degradation.

Since the joint model, having more context attributes, affords a larger variety of different contexts, we use this model in the remaining experiments.

In Table 4, we present the perplexity numbers for our standard model with additional trees. We denote context-restricted trees by their Markovian or-

Model	size	PPL
1w1t + 2w2t + 3w3t + 4w4t (*)	294MB	147.1
(*) + 4w3t + 3w2t	579MB	143.5
(*) + 4w3t + 3w4t	587MB	144.9
(*) + 4w3t + 3w4t + 3w2t + 2w3t	699MB	**140.7**
(*) + 1 × bernoulli-rnd	464MB	149.7
(*) + 2 × bernoulli-rnd	632MB	150.4
(*) + 3 × bernoulli-rnd	804MB	151.1
(*) + 1 × data-rnd	484MB	147.0
(*) + 2 × data-rnd	673MB	145.0
(*) + 3 × data-rnd	864MB	145.2

Table 4: Perplexity results using the standard syntactic model with additional trees. "bernoulli-rnd" and "data-rnd" indicate fourgram trees randomized using Bernoulli trials and varying training data, respectively. The second column shows the combined size of decision trees in the forest.

ders (words w and tags t independently), so 3w2t indicates a decision tree implementing the probability function: $p(w_i t_i | w_{i-1} w_{i-2} t_{i-1})$. The fourgram joint model presented in Table 1 has four trees and is labeled with the formula "1w1t + 2w2t + 3w3t + 4w4t" in Table 4. The randomly grown trees (denoted "bernoulli-rnd") are grown utilizing the full context 4w4t using the methods described in Section 4.2. All models utilize the generalized interpolation method described in Section 3.2.

As can be seen in Table 4, adding undegraded trees consistently improves the performance of an already strong baseline, while adding random trees only increases the perplexity because their quality is worse than undegraded trees'. Trees produced by data randomization (denoted "data-rnd") also improve the performance of the model; however, the improvement is not greater than that of additional lower order trees, which are considerably smaller in size.

5 ASR Rescoring Results

In order to verify that the improvements in perplexity that we observe in Tables 1 and 4 are sufficient for an impact on a task, we measure Word Error Rate (WER) of our models on an Automatic Speech Recognition (ASR) rescoring task using the Wall Street Journal corpus (WSJ) for evaluation. The test set consists of 4,088 utterances of WSJ0. We optimized the weights for the combination of acoustic and language model scores on a separate development set comprised of 1,243 utterances from Hub2 5k closed vocabulary and the WSJ1 5k open vocabulary sets.

Model	PPL	WER
n-gram	161.7	7.81%
1w1t + 2w2t + 3w3t + 4w4t (Eq.8)	156.5	7.57%
1w1t + 2w2t + 3w3t + 4w4t (*)	147.1	7.32%
(*) + 4w3t + 3w4t + 3w2t + 2w3t	140.7	**7.20%**

Table 5: Perplexity and WER results. Note that the last two rows are syntactic models using the interpolation method of Eq. 9.

The ASR system used to produce lattices is based on the 2007 IBM Speech transcription system for the GALE Distillation Go/No-go Evaluation (Chen et al., 2006). The acoustic models are state-of-the-art discriminatively trained models which are trained on Broadcast News (BN) Hub4 acoustic training data. Lattices were produced using a trigram LM trained on the same data as the models we evaluate, then 1,000 best unique hypotheses were extracted from the lattices. WER of the 1-best hypothesis on the test set is 8.07% and the oracle WER is 3.54%.

In Table 5, we present WER results along with the corresponding perplexity numbers from Tables 1 and 4 for our lowest perplexity syntactic model, as well as the baselines (modified KN n-gram model and standard decision tree models using interpolation methods of Eq. 8 and Eq. 9). The interpolation method of Eq. 9 substantially improves performance over the interpolation method of Eq. 8, reducing WER by 0.25% absolute ($p < 10^{-5}$). Adding four trees utilizing context restricted in different ways further reduces WER by 0.12%, which is also a statistically significant ($p < 0.025$) improvement over the baseline models labeled (*). Altogether, the improvements over the n-gram baseline add up to 0.61% absolute (8% relative) WER reduction.

6 Conclusion

In this paper, we investigate various aspects of combining multiple decision trees in a single language model. We observe that the generalized interpola-

tion (Eq. 9) for decision tree models proposed in (Filimonov and Harper, 2011) is in fact a forest interpolation method rather than a backoff interpolation because, in Eq. 9, models do not have explicit higher-lower order relation as they do in backoff interpolation (Eq. 6). Thus, in this paper we investigate the question of how to construct decision trees so that their combination results in improved performance (under the assumption that computational tractability allows only a handful of decision trees in a forest). We compare various techniques for producing forests of trees and observe that methods that diversify trees by introducing random degradation of the tree construction algorithm perform more poorly (especially with joint models) than methods in which the trees are constructed without degradation and with variability being introduced via parameters that are inherently arbitrary (e.g., training data fold differences or initializations of greedy search algorithms). Additionally, we observe that simply restricting the context used to construct trees in different ways, not only produces smaller trees (because of the context reduction), but the resulting variations in trees also produce forests that are *at least* as good as forests of larger trees.

7 Acknowledgments

We would like to thank Ariya Rastrow for providing word lattices for the ASR rescoring experiments.

References

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. 1990. A tree-based statistical language model for natural language speech recognition. *Readings in speech recognition*, pages 507–514.

Srinivas Bangalore. 1996. 'Almost parsing' technique for language modeling. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1173–1176.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NAACL*, pages 4–6.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling for speech recognition. *CoRR*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318.

S. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig. 2006. Advances in speech transcription at IBM under the DARPA EARS program. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1596–1608.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *Proceedings of the EMNLP 2009*.

Denis Filimonov and Mary Harper. 2011. Generalized interpolation in decision tree LM. In *Proceedings of the 49st Annual Meeting of the Association for Computational Linguistics*.

Peter Heeman. 1998. POS tagging versus classes in language modeling. In *Sixth Workshop on Very Large Corpora*.

Zhongqiang Huang and Mary Harper. 2009. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the EMNLP 2009*.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397.

Sven Martin, Jorg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256.

J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ronald Rosenfeld, Jaime Carbonell, and Alexander Rudnicky. 1994. Adaptive statistical language modeling: A maximum entropy approach. Technical report.

Peng Xu. 2005. *Random Forests and Data Sparseness Problem in Language Modeling*. Ph.D. thesis, Baltimore, Maryland, April.

Imed Zitouni. 2007. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104.