Smoothed marginal distribution constraints for language modeling

Brian Roark[†°], **Cyril Allauzen**[°] **and Michael Riley**[°]
[†]Oregon Health & Science University, Portland, Oregon [°]Google, Inc., New York
roarkbr@gmail.com, {allauzen,riley}@google.com

Abstract

We present an algorithm for re-estimating parameters of backoff n-gram language models so as to preserve given marginal distributions, along the lines of well-known Kneser-Ney (1995) smoothing. Unlike Kneser-Ney, our approach is designed to be applied to any given smoothed backoff model, including models that have already been heavily pruned. As a result, the algorithm avoids issues observed when pruning Kneser-Ney models (Siivola et al., 2007; Chelba et al., 2010), while retaining the benefits of such marginal distribution constraints. We present experimental results for heavily pruned backoff n-gram models, and demonstrate perplexity and word error rate reductions when used with various baseline smoothing methods. An open-source version of the algorithm has been released as part of the OpenGrm ngram library.[1]

1 Introduction

Smoothed n-gram language models are the defacto standard statistical models of language for a wide range of natural language applications, including speech recognition and machine translation. Such models are trained on large text corpora, by counting the frequency of n-gram collocations, then normalizing and smoothing (regularizing) the resulting multinomial distributions. Standard techniques store the observed n-grams and derive probabilities of unobserved n-grams via their longest observed suffix and "backoff" costs associated with the prefix histories of the unobserved suffixes. Hence the size of the model grows with the number of observed n-grams, which is very large for typical training corpora.

[1] www.opengrm.org

Natural language applications, however, are commonly used in scenarios requiring relatively small footprint models. For example, applications running on mobile devices or in low latency streaming scenarios may be required to limit the complexity of models and algorithms to achieve the desired operating profile. As a result, statistical language models – an important component of many such applications – are often trained on very large corpora, then modified to fit within some pre-specified size bound. One method to achieve significant space reduction is through randomized data structures, such as Bloom (Talbot and Osborne, 2007) or Bloomier (Talbot and Brants, 2008) filters. These data structures permit efficient querying for specific n-grams in a model that has been stored in a fraction of the space required to store the full, exact model, though with some probability of false positives. Another common approach – which we pursue in this paper – is model pruning, whereby some number of the n-grams are removed from explicit storage in the model, so that their probability must be assigned via backoff smoothing. One simple pruning method is count thresholding, i.e., discarding n-grams that occur less than k times in the corpus. Beyond count thresholding, the most widely used pruning methods (Seymore and Rosenfeld, 1996; Stolcke, 1998) employ greedy algorithms to reduce the number of stored n-grams by comparing the stored probabilities to those that would be assigned via the backoff smoothing mechanism, and removing those with the least impact according to some criterion.

While these greedy pruning methods are highly effective for models estimated with most common smoothing approaches, they have been shown to be far less effective with Kneser-Ney trained language models (Siivola et al., 2007; Chelba et al., 2010), leading to severe degradation in model quality relative to other standard smoothing meth-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 43–52,
Sofia, Bulgaria, August 4-9 2013.

4-gram models	Backoff			Interpolated		
	Perplexity		n-grams	Perplexity		n-grams
Smoothing method	full	pruned	(×1000)	full	pruned	(×1000)
Absolute Discounting (Ney et al., 1994)	120.5	197.3	383.4	119.8	198.1	386.2
Witten-Bell (Witten and Bell, 1991)	118.8	196.3	380.4	121.6	202.3	396.4
Ristad (1995)	126.4	203.6	395.6	——- N/A ——-		
Katz (1987)	119.8	198.1	386.2	——- N/A ——-		
Kneser-Ney (Kneser and Ney, 1995)	114.5	285.1	388.2	115.8	274.3	398.7
Mod. Kneser-Ney (Chen and Goodman, 1998)	116.3	280.6	396.2	112.8	270.7	399.1

Table 1: Reformatted version of Table 3 in Chelba et al. (2010), demonstrating perplexity degradation of Kneser-Ney smoothed models in contrast to other common smoothing methods. Data: English Broadcast News, 128M words training; 692K words test; 143K word vocabulary. 4-gram language models, pruned using Stolcke (1998) relative entropy pruning to approximately 1.3% of the original size of 31,095,260 n-grams.

ods. Thus, while Kneser-Ney may be the preferred smoothing method for large, unpruned models – where it can achieve real improvements over other smoothing methods – when relatively sparse, pruned models are required, it has severely diminished utility.

Table 1 presents a slightly reformatted version of Table 3 from Chelba et al. (2010). In their experiments (see Table 1 caption for specifics on training/test setup), they trained 4-gram Broadcast News language models using a variety of both backoff and interpolated smoothing methods and measured perplexity before and after Stolcke (1998) relative entropy based pruning. With this size training data, the perplexity of all of the smoothing methods other than Kneser-Ney degrades from around 120 with the full model to around 200 with the heavily pruned model. Kneser-Ney smoothed models have lower perplexity with the full model than the other methods by about 5 points, but degrade with pruning to far higher perplexity, between 270-285.

The cause of this degradation is Kneser-Ney's unique method for estimating smoothed language models, which will be presented in more detail in Section 3. Briefly, the smoothing method reestimates lower-order n-gram parameters in order to avoid over-estimating the likelihood of n-grams that already have ample probability mass allocated as part of higher-order n-grams. This is done via a marginal distribution constraint which requires the expected frequency of the lower-order n-grams to match their observed frequency in the training data, much as is commonly done for maximum entropy model training. Goodman (2001) proved that, under certain assumptions, such constraints can only improve language models. Lower-order n-gram parameters resulting from Kneser-Ney are not relative frequency estimates, as with other smoothing methods; rather they are parameters estimated specifically for use within the larger smoothed model.

There are (at least) a couple of reasons why such parameters do not play well with model pruning. First, the pruning methods commonly use lower order n-gram probabilities to derive an estimate of state marginals, and, since these parameters are no longer smoothed relative frequency estimates, they do not serve that purpose well. For this reason, the widely-used SRILM toolkit recently provided switches to modify their pruning algorithm to use another model for state marginal estimates (Stolcke et al., 2011). Second, and perhaps more importantly, the marginal constraints that were applied prior to smoothing will not in general be consistent with the much smaller pruned model. For example, if a bigram parameter is modified due to the presence of some set of trigrams, and then some or all of those trigrams are pruned from the model, the bigram associated with the modified parameter will be unlikely to have an overall expected frequency equal to its observed frequency anymore. As a result, the resulting model degrades dramatically with pruning.

In this paper, we present an algorithm that imposes marginal distribution constraints of the sort used in Kneser-Ney modeling on arbitrary smoothed backoff n-gram language models. Our approach makes use of the same sort of derivation as the original Kneser-Ney modeling, but, among other differences, relies on smoothed estimates of the empirical relative frequency rather than the unsmoothed observed frequency. The algorithm can be applied after the smoothed model has been pruned, hence avoiding the pitfalls associated with Kneser-Ney modeling. Furthermore, while Kneser-Ney is conventionally defined as a variant of absolute discounting, our method can be applied to models smoothed with any backoff smoothing, including mixtures of models, widely

used for domain adaptation.

We next establish formal preliminaries and our smoothed marginal distribution constraints method.

2 Preliminaries

N-gram language models are typically presented mathematically in terms of words w, the strings (histories) h that precede them, and the suffixes of the histories (backoffs) h' that are used in the smoothing recursion. Let V be a vocabulary (alphabet), and V^* a string of zero or more symbols drawn from V. Let V^k denote the set of strings $\mathbf{w} \in V^*$ of length k, i.e., $|\mathbf{w}| = k$. We will use variables $u, v, w, x, y, z \in V$ to denote single symbols from the vocabulary; $h, g \in V^*$ to denote history sequences preceding the specific word; and $h', g' \in V^*$ the respective backoff histories of h and g as typically defined (see below). For a string $\mathbf{w} = w_1 \dots w_{|\mathbf{w}|}$ we can calculate the smoothed conditional probability of each word w_i in the sequence given the k words that preceded it, depending on the order of the Markov model. Let $h_i^k = w_{i-k} \dots w_{i-1}$ be the previous k words in the sequence. Then the smoothed model is defined recursively as follows:

$$\mathrm{P}(w_i \mid h_i^k) = \begin{cases} \overline{\mathrm{P}}(w_i \mid h_i^k) & \text{if } c(h_i^k w_i) > 0 \\ \alpha(h_i^k)\, \mathrm{P}(w_i \mid h_i^{k-1}) & \text{otherwise} \end{cases}$$

where $c(h_i^k w_i)$ is the count of the n-gram sequence $w_{i-k} \dots w_i$ in the training corpus; $\overline{\mathrm{P}}$ is a regularized probability estimate that provides some probability mass for unobserved n-grams; and $\alpha(h_i^k)$ is a factor that ensures normalization. Note that for $h = h_i^k$, the typically defined backoff history $h' = h_i^{k-1}$, i.e., the longest suffix of h that is not h itself. When we use h' and g' (for notational convenience) in future equations, it is this definition that we are using.

There are many ways to estimate $\overline{\mathrm{P}}$, including absolute discounting (Ney et al., 1994), Katz (1987) and Witten and Bell (1991). Interpolated models are special cases of this form, where the $\overline{\mathrm{P}}$ is determined using model mixing, and the α parameter is exactly the mixing factor value for the lower order model.

N-gram language models allow for a sparse representation, so that only a subset of the possible n-grams must be explicitly stored. Probabilities for the rest of the n-grams are calculated through the "otherwise" semantics in the equation above. For an n-gram language model G, we will say that an n-gram $hw \in G$ if it is explicitly represented in the model; otherwise $hw \notin G$. In the standard n-gram formulation above, the assumption is that if $c(h_i^k w_i) > 0$ then the n-gram has a parameter; yet with pruning, we remove many observed n-grams from the model, hence this is no longer the appropriate criterion. We reformulate the standard equation as follows:

$$\mathrm{P}(w_i|h_i^k) = \begin{cases} \beta(h_i^k w_i) & \text{if } h_i^k w_i \in G \\ \alpha(h_i^k, h_i^{k-1})\, \mathrm{P}(w_i|h_i^{k-1}) & \text{otherwise} \end{cases} \quad (1)$$

where $\beta(h_i^k w_i)$ is the parameter associated with the n-gram $h_i^k w_i$ and $\alpha(h_i^k, h_i^{k-1})$ is the backoff cost associated with going from state h_i^k to state h_i^{k-1}. We assume that, if $hw \in G$ then all prefixes and suffixes of hw are also in G.

Figure 1 presents a schema of an automaton representation of an n-gram model, of the sort used in the OpenGrm library (Roark et al., 2012). States represent histories h, and the words w, whose probabilities are conditioned on h, label the arcs, leading to the history state for the subsequent word. State labels are provided in Figure 1 as a convenience, to show the (implicit) history encoded by the state, e.g., 'xyz' indicates that the state represents a history with the previous three symbols being x, y and z. Failure arcs, labeled with a ϕ in Figure 1, encode an "otherwise" semantics and have as destination the origin state's backoff history. Many higher order states will back off to the same lower order state, specifically those that share the same suffix.

Note that, in general, the recursive definition of backoff may require the traversal of several back-

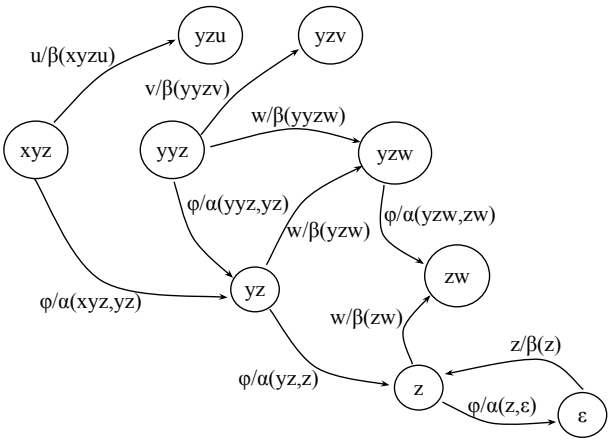

Figure 1: N-gram weighted automaton schema. State labels are presented for convenience, to specify the history implicitly encoded by the state.

off arcs before emitting a word, e.g., the highest order states in Figure 1 needing to traverse a couple of ϕ arcs to reach state 'z'. We can define the backoff cost between a state h_i^k and any of its suffix states as follows. Let $\alpha(h, h) = 1$ and for $m > 1$,

$$\alpha(h_i^k, h_i^{k-m}) = \prod_{j=1}^{m} \alpha(h_i^{k-j+1}, h_i^{k-j}).$$

If $h_i^k w \notin G$ then the probability of that n-gram will be defined in terms of backoff to its longest suffix $h_i^{k-m} w \in G$. Let h^{wG} denote the longest suffix of h such that $h^{wG} w \in G$. Note that this is not necessarily a proper suffix, since h^{wG} could be h itself or it could be ϵ. Then

$$\mathrm{P}(w \mid h) = \alpha(h, h^{wG})\, \beta(h^{wG} w) \quad (2)$$

which is equivalent to equation 1.

3 Marginal distribution constraints

Marginal distribution constraints attempt to match the expected frequency of an n-gram with its observed frequency. In other words, if we use the model to randomly generate a very large corpus, the n-grams should occur with the same relative frequency in both the generated and original (training) corpus. Standard smoothing methods overgenerate lower-order n-grams. Using standard n-gram notation (where g' is the backoff history for g), this constraint is stated in Kneser and Ney (1995) as

$$\widehat{\mathrm{P}}(w \mid h') = \sum_{g : g' = h'} \mathrm{P}(g, w \mid h') \quad (3)$$

where $\widehat{\mathrm{P}}$ is the empirical relative frequency estimate. Taking this approach, certain base smoothing methods end up with very nice, easy to calculate solutions based on counts. Absolute discounting (Ney et al., 1994) in particular, using the above approach, leads to the well-known Kneser-Ney smoothing approach (Kneser and Ney, 1995; Chen and Goodman, 1998). We will follow this same approach, with a couple of changes. First, we will make use of regularized estimates of relative frequency $\overline{\mathrm{P}}$ rather than raw relative frequency $\widehat{\mathrm{P}}$. Second, rather than just looking at observed histories h that back off to h', we will look at all histories (observed or not) of the length of the longest history in the model. For notational simplicity, suppose we have an $n{+}1$-gram model, hence the longest history in the model is of length n. Assume the length of the particular backoff history $|h'| = k$. Let $V^{n-k}h'$ be the set of strings $h \in V^n$ with h' as a suffix. Then we can restate the marginal distribution constraint in equation 3 as

$$\overline{\mathrm{P}}(w \mid h') = \sum_{h \in V^{n-k} h'} \mathrm{P}(h, w \mid h') \quad (4)$$

Next we solve for $\beta(h'w)$ parameters used in equation 1. Note that h' is a suffix of any $h \in V^{n-k}h'$, so conditioning probabilities on h and h' is the same as conditioning on just h. Each of the following derivation steps simply relies on the chain rule or definition of conditional probability, as well as pulling terms out of the summation.

$$\begin{aligned}
\overline{\mathrm{P}}(w \mid h') &= \sum_{h \in V^{n-k} h'} \mathrm{P}(h, w \mid h') \\
&= \sum_{h \in V^{n-k} h'} \mathrm{P}(w \mid h, h')\, \mathrm{P}(h \mid h') \\
&= \sum_{h \in V^{n-k} h'} \mathrm{P}(w \mid h)\, \frac{\mathrm{P}(h)}{\sum_{g \in V^{n-k} h'} \mathrm{P}(g)} \\
&= \frac{1}{\sum_{g \in V^{n-k} h'} \mathrm{P}(g)} \sum_{h \in V^{n-k} h'} \mathrm{P}(w \mid h)\, \mathrm{P}(h) \quad (5)
\end{aligned}$$

Then, multiplying both sides by the normalizing denominator on the right-hand side and using equation 2 to substitute $\alpha(h, h^{wG})\ \beta(h^{wG} w)$ for $\mathrm{P}(w \mid h)$:

$$\begin{aligned}
\overline{\mathrm{P}}(w \mid h') \sum_{g \in V^{n-k} h'} \mathrm{P}(g) &= \sum_{h \in V^{n-k} h'} \mathrm{P}(w \mid h)\, \mathrm{P}(h) \\
&= \sum_{h \in V^{n-k} h'} \alpha(h, h^{wG})\, \beta(h^{wG} w)\, \mathrm{P}(h) \quad (6)
\end{aligned}$$

Note that we are only interested in $h'w \in G$, hence there are two disjoint subsets of histories $h \in V^{n-k}h'$ that are being summed over: those such that $h^{wG} = h'$ and those such that $|h^{wG}| > |h'|$. We next separate these sums in the next step of the derivation:

$$\begin{aligned}
\overline{\mathrm{P}}(w \mid h') \sum_{g \in V^{n-k} h'} \mathrm{P}(g) = \\
\sum_{h \in V^{n-k} h' : |h^{wG}| > |h'|} \alpha(h, h^{wG})\, \beta(h^{wG} w)\, \mathrm{P}(h) \ + \\
\sum_{h \in V^{n-k} h' : h^{wG} = h'} \alpha(h, h')\, \beta(h' w)\, \mathrm{P}(h) \quad (7)
\end{aligned}$$

Finally, we solve for $\beta(h'w)$ in the second sum on the right-hand side of equation 7, yielding the formula in equation 8. Note that this equation is the correlate of equation (6) in Kneser and Ney

$$\beta(h'w) = \frac{\overline{\mathrm{P}}(w \mid h') \sum_{g \in V^{n-k}h'} \mathrm{P}(g) - \sum_{h \in V^{n-k}h' : |h^{wG}| > |h'|} \alpha(h, h^{wG})\, \beta(h^{wG}w)\, \mathrm{P}(h)}{\sum_{h \in V^{n-k}h' : h^{wG} = h'} \alpha(h, h')\, \mathrm{P}(h)} \quad (8)$$

(1995), modulo the two differences noted earlier: use of smoothed probability $\overline{\mathrm{P}}$ rather than raw relative frequency; and summing over all history substrings in $V^{n-k}h'$ rather than just those with count greater than zero, which is also a change due to smoothing. Keep in mind, $\overline{\mathrm{P}}$ is the target expected frequency from a given smoothed model. Kneser-Ney models are not useful input models, since their $\overline{\mathrm{P}}$ n-gram parameters are not relative frequency estimates. This means that we cannot simply 'repair' pruned Kneser-Ney models, but must use other smoothing methods where the smoothed values are based on relative frequency estimation.

There are, in addition, two other important differences in our approach from that in Kneser and Ney (1995), which would remain as differences even if our target expected frequency were the unsmoothed relative frequency $\widehat{\mathrm{P}}$ instead of the smoothed estimate $\overline{\mathrm{P}}$. First, the sum in the numerator is over histories of length n, the highest order in the n-gram model, whereas in the Kneser-Ney approach the sum is over histories that immediately back off to h', i.e., from the next highest order in the n-gram model. Thus the unigram distribution is with respect to the bigram model, the bigram model is with respect to the trigram model, and so forth. In our optimization, we sum instead over all possible history sequences of length n. Second, an early assumption made in Kneser and Ney (1995) is that the denominator term in their equation (6) (our Eq. 8) is constant across all words for a given history, which is clearly false. We do not make this assumption. Of course, the probabilities must be normalized, hence the final values of $\beta(h'w)$ will be proportional to the values in Eq. 8.

We briefly note that, like Kneser-Ney, if the baseline smoothing method is consistent, then the amount of smoothing in the limit will go to zero and our resulting model will also be consistent.

The smoothed relative frequency estimate $\overline{\mathrm{P}}$ and higher order β values on the right-hand side of Eq. 8 are given values (from the input smoothed model and previous stages in the algorithm, respectively), implying an algorithm that estimates highest orders of the model first. In addition, steady state history probabilities $\mathrm{P}(h)$ must be calculated. We turn to the estimation algorithm next.

4 Model constraint algorithm

Our algorithm takes a smoothed backoff n-gram language model in an automaton format (see Figure 1) and returns a smoothed backoff n-gram language model with the same topology. For all n-grams in the model that are suffixes of other n-grams in the model – i.e., that are backed-off to – we calculate the weight provided by equation 8 and assign it (after normalization) to the appropriate n-gram arc in the automaton. There are several important considerations for this algorithm, which we address in this section. First, we must provide a probability for every state in the model. Second, we must memoize summed values that are used repeatedly. Finally, we must iterate the calculation of certain values that depend on the n-gram weights being re-estimated.

4.1 Steady state probability calculation

The steady state probability $\mathrm{P}(h)$ is taken to be the probability of observing h after a long word sequence, i.e., the state's relative frequency in a long sequence of randomly-generated sentences from the model:

$$\mathrm{P}(h) = \lim_{m \to \infty} \sum_{w_1 \ldots w_m} \hat{\mathrm{P}}(w_1 \ldots w_m h) \quad (9)$$

where $\hat{\mathrm{P}}$ is the *corpus* probability derived as follows: The smoothed n-gram probability model $\mathrm{P}(w \mid h)$ is naturally extended to a sentence $s = w_0 \ldots w_l$, where $w_0 =$ <s> and $w_l =$ </s> are the sentence initial and final words, by $\mathrm{P}(s) = \prod_{i=1}^{l} \mathrm{P}(w_i \mid h_i^n)$. The corpus probability $s_1 \ldots s_r$ is taken as:

$$\hat{\mathrm{P}}(s_1 \ldots s_r) = (1-\lambda)\lambda^{r-1} \prod_{i=1}^{r} \mathrm{P}(s_i) \quad (10)$$

where λ parameterizes the corpus length distribution.[2] Assuming the n-gram language model automaton G has a single final state </s> into

[2] $\hat{\mathrm{P}}$ models words in a corpus rather than a single sentence since Equation 9 tends to zero as $m \to \infty$ otherwise. In Markov chain terms, the corpus distribution is made irreducible to allow a non-trivial stationary distribution.

which all </s> arcs enter, adding a λ weighted ϵ arc from the </s> state to the initial state and having a final weight $1 - \lambda$ in order to leave the automaton at the </s> state will model this corpus distribution. According to Eq. 9, $P(h)$ is then the stationary distribution of the finite irreducible Markov Chain defined by this altered automaton. There are many methods for computing such a stationary distribution; we use the well-known power method (Stewart, 1999).

One difficulty remains to be resolved. The backoff arcs have a special interpretation in the automaton: they are traversed only if a word fails to match at the higher order. These failure arcs must be properly handled before applying standard stationary distribution calculations. A simple approach would be for each word w' and state h such that $hw' \notin G$, but $h'w' \in G$, add a w' arc from state h to $h'w'$ with weight $\alpha(h, h')\beta(h'w')$ and then remove all failure arcs (see Figure 2a). This however results in an automaton with $|V|$ arcs leaving every state, which is unwieldy with larger vocabularies and n-gram orders. Instead, for each word w and state h such that $hw \in G$, add a w arc from state h to $h'w$ with weight $-\alpha(h, h')\beta(h'w)$ and then replace all failure labels with ϵ labels (see Figure 2b). In this case, the added negatively-weighted arcs compensate for the excess probability mass allowed by the epsilon arcs[3]. The number of added arcs is no more than found in the original model.

4.2 Accumulation of higher order values

We are summing over all possible histories of length n in equation 8, and the steady state probability calculation outlined in the previous section includes the probability mass for histories $h \notin G$. The probability mass of states not in G ends up being allocated to the state representing their longest suffix that is explicitly in G. That is the state that would be active when these histories are encountered. Hence, once we have calculated the steady state probabilities for each state in the smoothed model, we only need to sum over states explicitly in the model.

As stated earlier, the use of $\beta(h^{wG}w)$ in the numerator of equation 8 for h^{wG} that are larger than h' implies that the longer n-grams must be re-estimated first. Thus we process each history length in descending order, finishing with the unigram state. Since we assume that, for every n-gram $hw \in G$, every prefix and suffix is also in G, we know that if $h'w \notin G$ then there is no history h such that h' is a suffix of h and $hw \in G$. This allows us to recursively accumulate the $\alpha(h, h')\,\mathrm{P}(h)$ in the denominator of Eq. 8.

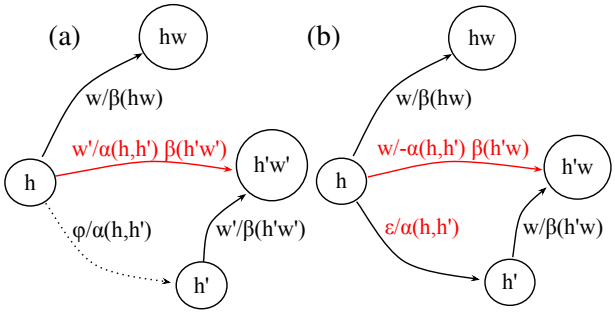

Figure 2: Schemata showing failure arc handling: (a) ϕ removal: add w' arc (red), delete ϕ arc; (b) ϕ replacement: add w arc (red), replace ϕ by ϵ (red)

For every n-gram, we can accumulate values required to calculate the three terms in equation 8, and pass them along to calculate lower order n-gram values. Note, however, that a naive implementation of an algorithm to assign these values is $O(|V|^n)$. This is due to the fact that the denominator factor must be accumulated for all higher order states that do *not* have the given n-gram. Hence, for every state h directly backing off to h' (order $|V|$), and for every n-gram arc leaving state h' (order $|V|$), some value must be accumulated. This can be particularly clearly seen at the unigram state, which has an arc for every unigram (the size of the vocabulary): for every bigram state (also order of the vocabulary), in the naive algorithm we must look for every possible arc. Since there are $O(|V|^{n-2})$ lower order histories in the model in the worst case, we have overall complexity $O(|V|^n)$. However, we know that the number of stored n-grams is very sparse relative to the possible number of n-grams, so the typical case complexity is far lower. Importantly, the denominator is calculated by first assuming that *all* higher order states back off to the current n-gram, then subtract out the mass associated with those that are already observed at the higher order. In such a way, we need only perform work for higher order n-grams hw that are explicitly in the model. This optimization achieves orders-of-magnitude speedups, so that models take seconds to process.

Because smoothing is not necessarily con-

[3]Since each negatively-weighted arc leaving a state exactly cancels an epsilon arc followed by a matching positively-weighted arc in each iteration of the power method, convergence is assured.

strained across n-gram orders, it is possible that higher-order n-grams could be smoothed less than lower order n-grams, so that the numerator of equation 8 can be less than zero, which is not valid. A value less than zero means that the higher order n-grams will already produce the n-gram more frequently than its smoothed expected frequency. We set a minimum value ϵ for the numerator, and any n-gram numerator value less than ϵ is replaced with ϵ (for the current study, $\epsilon = 0.001$). We find this to be relatively infrequent, about 1% of n-grams for most models.

4.3 Iteration

Recall that $\overline{\mathrm{P}}$ and β terms on the right-hand side of equation 8 are given and do not change. But there are two other terms in the equation that change as we update the n-gram parameters. The $\alpha(h, h')$ backoff weights in the denominator ensure normalization at the higher order states, and change as the n-gram parameters at the current state are modified. Further, the steady state probabilities will change as the model changes. Hence, at each state, we must iterate the calculation of the denominator term: first adjust n-gram weights and normalize; then recalculate backoff weights at higher order states and iterate. Since this only involves the denominator term, each n-gram weight can be updated by multiplying by the ratio of the old term and the new term.

After the entire model has been re-estimated, the steady state probability calculation presented in Section 4.1 is run again and model estimation happens again. As we shall see in the experimental results, this typically converges after just a few iterations. At this time, we have no convergence proofs for either of these iterative components to the algorithm, but expect that something can be said about this, which will be a priority in future work.

5 Experimental results

All results presented here are for English Broadcast News. We received scripts for replicating the Chelba et al. (2010) results from the authors, and we report statistics on our replication of their paper's results in Table 2. The scripts are distributed in such a way that the user supplies the data from LDC98T31 (1996 CSR HUB4 Language Model corpus) and the script breaks the collection into training and testing sets, normalizes the text, and trains and prunes the language models using the SRILM toolkit (Stolcke et al., 2011). Presumably due to minor differences in text normalization, resulting in very slightly fewer n-grams in all conditions, we achieve negligibly lower perplexities (one or two tenths of a point) in all conditions, as can be seen when comparing with Table 1. All of the same trends result, thus that paper's result is successfully replicated here. Note that we ran our Kneser-Ney pruning (noted with a † in the table), using the new `-prune-history-lm` switch in SRILM – created in response to the Chelba et al. (2010) paper – which allows the use of another model to calculate the state marginals for pruning. This fixes part of the problem – perplexity does not degrade as much as the Kneser-Ney pruned model in Table 1 – but, as argued earlier in this paper, this is not the sole reason for the degradation and the perplexity remains extremely inflated.

Smoothing method	Perplexity full	Perplexity pruned	n-grams (×1000) model	n-grams (×1000) diff
Abs.Disc.	120.4	197.1	382.3	-1.1
Witten-Bell	118.7	196.1	379.3	-1.1
Ristad	126.2	203.4	394.6	-1.1
Katz	119.7	197.9	385.1	-1.1
Kneser-Ney†	114.4	234.1	375.4	-12.7

Table 2: Replication of Chelba et al. (2010) using provided script. Using the script, the size of the unpruned model is 31,091,219 ngrams, 4,041 fewer than Chelba et al. (2010). † Kneser-Ney model pruned using `-prune-history-lm` switch in SRILM.

We follow Chelba et al. (2010) in training and test set definition, vocabulary size, and parameters for reporting perplexity. Note that unigrams in the models are never pruned, hence all models assign probabilities over an identical vocabulary and perplexity is comparable across models. For all results reported here, we use the SRILM toolkit for baseline model training and pruning, then convert from the resulting ARPA format model to an OpenFst format (Allauzen et al., 2007), as used in the OpenGrm n-gram library (Roark et al., 2012). We then apply the marginal distribution constraints, and convert the result back to ARPA format for perplexity evaluation with the SRILM toolkit. All models are subjected to full normalization sanity checks, as with typical model functions in the OpenGrm library.

Recall that our algorithm assumes that, for every n-gram in the model, all prefix and suffix n-grams are also in the model. For pruned models, the SRILM toolkit does not impose such a requirement, hence explicit arcs are added to the

Smoothing Method	Perplexity Pruned Model	Perplexity Pruned +MDC	Perplexity Δ	n-grams (×1000) in WFST
Abs.Disc.	197.1	187.4	9.7	389.2
Witten-Bell	196.1	185.7	10.4	385.0
Ristad	203.4	190.3	13.1	395.9
Katz	197.9	187.5	10.4	390.8
AD,WB,Katz Mixture	196.6	186.3	10.3	388.7

Table 3: Perplexity reductions achieved with marginal distribution constraints (MDC) on the heavily pruned models from Chelba et al. (2010), and a mixture model. WFST n-gram counts are slightly higher than ARPA format in Table 2 due to adding prefix and suffix n-grams.

model during conversion, with probability equal to what they would receive in the the original model. The resulting model is equivalent, but with a small number of additional arcs in the explicit representation (around 1% for the most heavily pruned models).

Table 3 presents perplexity results for models that result from applying our marginal distribution constraints to the four heavily pruned models from Table 2. In all four cases, we get perplexity reductions of around 10 points. We present the number of n-grams represented explicitly in the WFST, which is a slight increase from those presented in Table 2 due to the reintroduction of prefix and suffix n-grams.

In addition to the four models reported in Chelba et al. (2010), we produced a mixture model by interpolating (with equal weight) smoothed n-gram probabilities from the full (unpruned) absolute discounting, Witten-Bell and Katz models, which share the same set of n-grams. After renormalizing and pruning to approximately the same size as the other models, we get commensurate gains using this model as with the other models.

Figure 3 demonstrates the importance of iterating the steady state history calculation. All of the methods achieve perplexity reductions with subsequent iterations. Katz and absolute discounting achieve very little reduction in the first iteration, but catch back up in the second iteration.

The other iterative part of the algorithm, discussed in Section 4.3, is the denominator of equation 8, which changes due to adjustments in the backoff weights required by the revised n-gram probabilities. If we do not iteratively update the backoff weights when reestimating the weights, the 'Pruned+MDC' perplexities in Table 3 increase by between 0.2–0.4 points. Hence, iterating the steady state probability calculation is quite important, as illustrated by Figure 3; iterating the

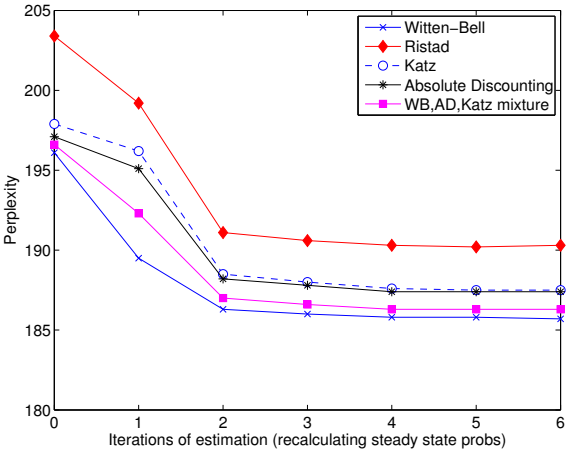

Figure 3: Models resulting from different numbers of parameter re-estimation iterations. Iteration 0 is the baseline pruned model.

denominator calculation much less so, at least for these models. We noted in Section 3 that a key difference between our approach and Kneser and Ney (1995) is that their approach treated the denominator as a constant. If we do this, the 'Pruned+MDC' perplexities increase by between 4.5–5.6 points, i.e., about half of the perplexity reduction is lost for each method. Thus, while iteration of denominator calculation may not be critical, it should not be treated as a constant.

We now look at the impacts on system performance we can achieve with these new models[4], and whether the perplexity differences that we observe translate to real error rate reductions.

For automatic speech recognition experiments, we used as test set the 1997 Hub4 evaluation set consisting of 32,689 words. The acoustic model is a tied-state triphone GMM-based HMM whose input features are 9-frame stacked 13-dimensional PLP-cepstral coefficients projected down to 39 dimensions using LDA. The model was trained on the 1996 and 1997 Hub4 acoustic model training sets (about 150 hours of data) using semi-tied covariance modeling and CMLLR-based speaker adaptive training and 4 iterations of boosted MMI.

We used a multi-pass decoding strategy: two quick passes for adaptation supervision, CMLLR and MLLR estimation; then a slower full decoding pass running about 3 times slower than real time.

Table 4 presents recognition results for the heavily pruned models that we have been considering, both for first pass decoding and rescoring of the resulting lattices using failure transitions rather than epsilon backoff approximations.

[4]For space purposes, we exclude the Ristad method from this point forward since it is not competitive with the others.

Smoothing Method	Word error rate (WER) First pass Pruned Model	First pass Pruned +MDC	Rescoring Pruned Model	Rescoring Pruned +MDC
Abs.Disc.	20.5	19.7	20.2	19.6
Witten-Bell	20.5	19.9	20.1	19.6
Katz	20.5	19.7	20.2	19.7
Mixture	20.5	19.6	20.2	19.6
Kneser-Ney[a]	22.1		22.2	
Kneser-Ney[b]	20.5		20.6	

Table 4: WER reductions achieved with marginal distribution constraints (MDC) on the heavily pruned models from Chelba et al. (2010), and a mixture model. Kneser-Ney results are shown for: a) original pruning; and b) with `-prune-history-lm` switch.

The perplexity reductions that were achieved for these models do translate to real word error rate reductions at both stages of between 0.5 and 0.9 percent absolute. All of these gains are statistically significant at $p < 0.0001$ using the stratified shuffling test (Yeh, 2000). For pruned Kneser-Ney models, fixing the state marginals with the `-prune-history-lm` switch reduces the WER versus the original pruned model, but no reductions were achieved vs. baseline methods.

Table 5 presents perplexity and WER results for less heavily pruned models, where the pruning thresholds were set to yield approximately 1.5 million n-grams (4 times more than the previous models); and another set at around 5 million n-grams, as well as the full, unpruned models. While the robust gains we've observed up to now persist with the 1.5M n-gram models (WER reductions significant, Witten-Bell at $p < 0.02$, others at $p < 0.0001$), the larger models yield diminishing gains, with no real WER improvements. Performance of Witten-Bell models with the marginal distribution constraints degrade badly for the larger models, indicating that this method of regularization, unmodified by aggressive pruning, does not provide a well suited distribution for this sort of optimization. We speculate that this is due to underregularization, having noted some floating point precision issues when allowing the backoff recalculation to run indefinitely.

6 Summary and Future Directions

The presented method reestimates lower order n-gram model parameters for a given smoothed backoff model, achieving perplexity and WER reductions for many smoothed models. There remain a number of open questions to investigate in the future. Recall that the numerator in Eq. 8 can be less than zero, meaning that no parameterization would lead to a model with the target frequency of the lower order n-gram, presumably due to over- or under-regularization. We anticipate a pre-constraint on the baseline smoothing method, that would recognize this problem and adjust the smoothing to ensure that a solution does exist. Additionally, it is clear that different regularization methods yield different behaviors, notably that large, relatively lightly pruned Witten-Bell models yield poor results. We will look to identify the issues with such models and provide general guidelines for prepping models prior to processing. Finally, we would like to perform extensive controlled experimentation to examine the relative contribution of the various aspects of our approach.

Acknowledgments

Thanks to Ciprian Chelba and colleagues for the scripts to replicate their results. This work was supported in part by a Google Faculty Research Award and NSF grant #IIS-0964102. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

Smoothing Method	MDC	Less heavily pruned model ngrams ($\times 10^6$)	PPL	WER FP	WER RS	Moderately pruned model ngrams ($\times 10^6$)	PPL	WER FP	WER RS	Full model ngrams ($\times 10^6$)	PPL	WER FP	WER RS
Abs. Disc.	N	1.53	146.6	18.1	17.9	5.19	129.1	17.0	16.6	31.1	120.4	16.2	16.1
	Y		141.2	17.2	17.2		126.3	16.6	16.6	31.1	117.0	16.0	16.0
Witten-Bell	N	1.54	145.8	18.1	17.6	5.08	129.4	17.3	16.8	31.1	118.7	16.3	16.1
	Y		139.7	17.9	17.4		126.4	18.4	17.3	31.1	118.4	18.1	17.6
Katz	N	1.57	146.6	17.8	17.7	5.10	128.9	16.8	16.6	31.1	119.7	16.2	16.1
	Y		141.1	17.3	17.3		125.7	16.6	16.6	31.1	114.7	16.2	16.1
Mixture	N	1.55	145.5	18.1	17.7	5.11	128.2	16.9	16.6	31.1	118.5	16.3	16.1
	Y		139.2	17.3	17.2		123.6	16.6	16.4	31.1	114.6	17.3	16.4
Kneser-Ney backoff model, unpruned:										31.1	114.4	15.8	15.9

Table 5: Perplexity (PPL) and both first pass (FP) and rescoring (RS) WER reductions for less heavily pruned models using marginal distribution constraints (MDC).

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Twelfth International Conference on Implementation and Application of Automata (CIAA 2007), Lecture Notes in Computer Science*, volume 4793, pages 11–23.

Ciprian Chelba, Thorsten Brants, Will Neveitt, and Peng Xu. 2010. Study on interaction between entropy pruning and Kneser-Ney smoothing. In *Proceedings of Interspeech*, page 24222425.

Stanley Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report, TR-10-98, Harvard University.

Joshua Goodman. 2001. A bit of progress in language modeling. *Computer Speech and Language*, 15(4):403–434.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 35(3):400–401.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 181–184.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38.

Eric S. Ristad. 1995. A natural law of succession. Technical Report, CS-TR-495-95, Princeton University.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66.

Kristie Seymore and Ronald Rosenfeld. 1996. Scalable backoff language models. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Vesa Siivola, Teemu Hirsimaki, and Sami Virpioja. 2007. On growing and pruning kneserney smoothed n-gram models. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1617–1624.

William J Stewart. 1999. Numerical methods for computing stationary distributions of finite irreducible markov chains. *Computational Probability*, pages 81–111.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of ACL-08: HLT*, pages 505–513.

David Talbot and Miles Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International COLING*, pages 947–953.