Aggregate and mixed-order Markov models for statistical language processing

Lawrence Saul and Fernando Pereira
{lsaul,pereira}@research.att.com
AT&T Labs – Research
180 Park Ave, D–130
Florham Park, NJ 07932

Abstract

We consider the use of language models whose size and accuracy are intermediate between different order n-gram models. Two types of models are studied in particular. Aggregate Markov models are class-based bigram models in which the mapping from words to classes is probabilistic. Mixed-order Markov models combine bigram models whose predictions are conditioned on different words. Both types of models are trained by Expectation-Maximization (EM) algorithms for maximum likelihood estimation. We examine smoothing procedures in which these models are interposed between different order n-grams. This is found to significantly reduce the perplexity of unseen word combinations.

1 Introduction

The purpose of a statistical language model is to assign high probabilities to likely word sequences and low probabilities to unlikely ones. The challenge here arises from the combinatorially large number of possibilities, only a fraction of which can ever be observed. In general, language models must learn to recognize word sequences that are functionally similar but lexically distinct. The learning problem, one of generalizing from sparse data, is particularly acute for large-sized vocabularies (Jelinek, Mercer, and Roukos, 1992).

The simplest models of natural language are n-gram Markov models. In these models, the probability of each word depends on the $n - 1$ words that precede it. The problems in estimating robust models of this form are well-documented. The number of parameters—or transition probabilities—scales as V^n, where V is the vocabulary size. For typical models (e.g., $n = 3$, $V = 10^4$), this number exceeds by many orders of magnitude the total number of words in any feasible training corpus.

The transition probabilities in n-gram models are estimated from the counts of word combinations in the training corpus. Maximum likelihood (ML) estimation leads to zero-valued probabilities for unseen n-grams. In practice, one adjusts or *smoothes* (Chen and Goodman, 1996) the ML estimates so that the language model can generalize to new phrases. Smoothing can be done in many ways—for example, by introducing artificial counts, backing off to lower-order models (Katz, 1987), or combining models by interpolation (Jelinek and Mercer, 1980).

Often a great deal of information is lost in the smoothing procedure. This is due to the great discrepancy between n-gram models of different order. The goal of this paper is to investigate models that are intermediate, in both size and accuracy, between different order n-gram models. We show that such models can "intervene" between different order n-grams in the smoothing procedure. Experimentally, we find that this significantly reduces the perplexity of unseen word combinations.

The language models in this paper were evaluated on the ARPA North American Business News (NAB) corpus. All our experiments used a vocabulary of sixty-thousand words, including tokens for punctuation, sentence boundaries, and an unknown word token standing for all out-of-vocabulary words. The training data consisted of approximately 78 million words (three million sentences); the test data, 13 million words (one-half million sentences). All sentences were drawn randomly without replacement from the NAB corpus. All perplexity figures given in the paper are computed by combining *sentence* probabilities; the probability of sentence $w_0 w_1 \cdots w_n w_{n+1}$ is given by $\prod_{i=1}^{n+1} P(w_i|w_0 \cdots w_{i-1})$, where w_0 and w_{n+1} are the start- and end-of-sentence markers, respectively. Though not reported below, we also confirmed that the results did not vary significantly for different randomly drawn test sets of the same size.

The organization of this paper is as follows. In Section 2, we examine *aggregate* Markov models, or class-based bigram models (Brown *et al.*, 1992) in which the mapping from words to classes

is probabilistic. We describe an iterative algorithm for discovering "soft" word classes, based on the Expectation-Maximization (EM) procedure for maximum likelihood estimation (Dempster, Laird, and Rubin, 1977). Several features make this algorithm attractive for large-vocabulary language modeling: it has no tuning parameters, converges monotonically in the log-likelihood, and handles probabilistic constraints in a natural way. The number of classes, C, can be small or large depending on the constraints of the modeler. Varying the number of classes leads to models that are intermediate between unigram ($C = 1$) and bigram ($C = V$) models.

In Section 3, we examine another sort of "intermediate" model, one that arises from combinations of non-adjacent words. Language models using such combinations have been proposed by Huang *et al.* (1993), Ney, Essen, and Kneser (1994), and Rosenfeld (1996), among others. We consider specifically the *skip-k* transition matrices, $M(w_{t-k}, w_t)$, whose predictions are conditioned on the kth previous word in the sentence. (The value of k determines how many words one "skips" back to make the prediction.) These predictions, conditioned on only a single previous word in the sentence, are inherently weaker than those conditioned on all k previous words. Nevertheless, by combining several predictions of this form (for different values of k), we can create a model that is intermediate in size and accuracy between bigram and trigram models.

Mixed-order Markov models express the predictions $P(w_t|w_{t-1}, w_{t-2}, \ldots, w_{t-m})$ as a convex combination of skip-k transition matrices, $M(w_{t-k}, w_t)$. We derive an EM algorithm to learn the mixing coefficients, as well as the elements of the transition matrices. The number of transition probabilities in these models scales as mV^2, as opposed to V^{m+1}. Mixed-order models are not as powerful as trigram models, but they can make much stronger predictions than bigram models. The reason is that quite often the immediately preceding word has less predictive value than earlier words in the same sentence.

In Section 4, we use aggregate and mixed-order models to improve the probability estimates from n-grams. This is done by interposing these models between different order n-grams in the smoothing procedure. We compare our results to a baseline trigram model that backs off to bigram and unigram models. The use of "intermediate" models is found to reduce the perplexity of unseen word combinations by over 50%.

In Section 5, we discuss some extensions to these models and some open problems for future research. We conclude that aggregate and mixed-order models provide a compelling alternative to language models based exclusively on n-grams.

2 Aggregate Markov models

In this section we consider how to construct class-based bigram models (Brown *et al.*, 1992). The problem is naturally formulated as one of hidden variable density estimation. Let $P(c|w_1)$ denote the probability that word w_1 is mapped into class c. Likewise, let $P(w_2|c)$ denote the probability that words in class c are followed by the word w_2. The class-based bigram model predicts that word w_1 is followed by word w_2 with probability

$$P(w_2|w_1) = \sum_{c=1}^{C} P(w_2|c)P(c|w_1), \qquad (1)$$

where C is the total number of classes. The hidden variable in this problem is the class label c, which is unknown for each word w_1. Note that eq. (1) represents the V^2 elements of the transition matrix $P(w_2|w_1)$ in terms of the $2CV$ elements of $P(w_2|c)$ and $P(c|w_1)$.

The Expectation-Maximization (EM) algorithm (Dempster, Laird, and Rubin, 1977) is an iterative procedure for estimating the parameters of hidden variable models. Each iteration consists of two steps: an E-step which computes statistics over the hidden variables, and an M-step which updates the parameters to reflect these statistics.

The EM algorithm for aggregate Markov models is particularly simple. The E-step is to compute, for each bigram $w_1 w_2$ in the training set, the *posterior* probability

$$P(c|w_1, w_2) = \frac{P(w_2|c)P(c|w_1)}{\sum_{c'} P(w_2|c')P(c'|w_1)}. \qquad (2)$$

Eq. (2) gives the probability that word w_1 was assigned to class c, based on the observation that it was followed by word w_2. The M-step uses these posterior probabilities to re-estimate the model parameters. The updates for aggregate Markov models are:

$$P(c|w_1) \leftarrow \frac{\sum_w N(w_1, w)P(c|w_1, w)}{\sum_{wc'} N(w_1, w)P(c'|w_1, w)}, \qquad (3)$$

$$P(w_2|c) \leftarrow \frac{\sum_w N(w, w_2)P(c|w, w_2)}{\sum_{ww'} N(w, w')P(c|w, w')}, \qquad (4)$$

where $N(w_1, w_2)$ denotes the number of counts of $w_1 w_2$ in the training set. These updates are guaranteed to increase the overall log-likelihood,

$$\ell = \sum_{w_1 w_2} N(w_1, w_2) \ln P(w_2|w_1), \qquad (5)$$

at each iteration. In general, they converge to a local (though not global) maximum of the log-likelihood. The perplexity V^* is related to the log-likelihood by $V^* = e^{-\ell/N}$, where N is the total number of words processed.

Though several algorithms (Brown *et al.*, 1992; Pereira, Tishby, and Lee, 1993) have been proposed
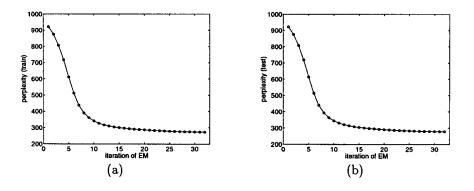
Figure 1: Plots of (a) training and (b) test perplexity versus number of iterations of the EM algorithm, for the aggregate Markov model with $C = 32$ classes.

C	train	test
1	964.7	964.9
2	771.2	772.2
4	541.9	543.6
8	399.5	401.5
16	328.8	331.8
32	278.9	283.2
V	123.6	—

Table 1: Perplexities of aggregate Markov models on the training and test sets; C is the number of classes. The case $C = 1$ corresponds to a ML unigram model; $C = V$, to a ML bigram model.
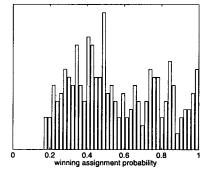

Figure 2: Histogram of the winning assignment probabilities, $\max_c P(c|w)$, for the three hundred most commonly occurring words.

for performing the decomposition in eq. (1), it is worth noting that only the EM algorithm directly optimizes the log-likelihood in eq. (5). This has obvious advantages if the goal of finding word classes is to improve the perplexity of a language model. The EM algorithm also handles probabilistic constraints in a natural way, allowing words to belong to more than one class if this increases the overall likelihood.

Our approach differs in important ways from the use of hidden Markov models (HMMs) for class-based language modeling (Jelinek *et al.*, 1992). While HMMs also use hidden variables to represent word classes, the dynamics are fundamentally different. In HMMs, the hidden state at time $t + 1$ is predicted (via the state transition matrix) from the hidden state at time t. On the other hand, in aggregate Markov models, the hidden state at time $t + 1$ is predicted (via the matrix $P(c_{t+1}|w_t)$) from the *word* at time t. The state-to-state versus word-to-state dynamics lead to different learning algorithms. For example, the Baum-Welch algorithm for HMMs requires forward and backward passes through each training sentence, while the EM algorithm we use does not.

We trained aggregate Markov models with 2, 4, 8, 16, and 32 classes. Figure 1 shows typical plots of the training and test set perplexities versus the number of iterations of the EM algorithm. Clearly, the two curves are very close, and the monotonic decrease in test set perplexity strongly suggests little if any overfitting, at least when the number of classes is small compared to the number of words in the vocabulary. Table 1 shows the final perplexities (after thirty-two iterations of EM) for various aggregate Markov models. These results confirm that aggregate Markov models are intermediate in accuracy between unigram $(C = 1)$ and bigram $(C = V)$ models.

The aggregate Markov models were also observed to discover meaningful word classes. Table 2 shows, for the aggregate model with $C = 32$ classes, the

#	Words
1	as cents made make take
2	ago day earlier Friday Monday month quarter reported said Thursday trading Tuesday Wednesday $\langle \ldots \rangle$
3	even get to
4	based days down home months up work years $\langle \% \rangle$
5	those $\langle , \rangle$ $\langle - \rangle$
6	$\langle . \rangle$ $\langle ? \rangle$
7	eighty fifty forty ninety seventy sixty thirty twenty $\langle () \rangle$ $\langle \cdot \rangle$
8	can could may should to will would
9	about at just only or than $\langle \& \rangle$ $\langle ; \rangle$
10	economic high interest much no such tax united well
11	president
12	because do how if most say so then think very what when where
13	according back expected going him plan used way
15	don't I people they we you
16	Bush company court department more officials police retort spokesman
17	former the
18	American big city federal general house military national party political state union York
19	billion hundred million nineteen
20	did $\langle " \rangle$ $\langle ' \rangle$
21	but called San $\langle : \rangle$ $\langle \text{start-of-sentence} \rangle$
22	bank board chairman end group members number office out part percent price prices rate sales shares use
23	a an another any dollar each first good her his its my old our their this
24	long Mr. year
25	business California case companies corporation dollars incorporated industry law money thousand time today war week $\langle () \rangle$ $\langle \text{unknown} \rangle$
26	also government he it market she that there which who
27	A. B. C. D. E. F. G. I. L. M. N. P. R. S. T. U.
28	both foreign international major many new oil other some Soviet stock these west world
29	after all among and before between by during for from in including into like of off on over since through told under until while with
30	eight fifteen five four half last next nine oh one second seven several six ten third three twelve two zero $\langle - \rangle$
31	are be been being had has have is it's not still was were
32	chief exchange news public service trade

Table 2: Most probable assignments for the 300 most frequent words in an aggregate Markov model with $C = 32$ classes. Class 14 is absent because it is not the most probable class for any of the selected words.)

most probable class assignments of the three hundred most commonly occurring words. To be precise, for each class c^*, we have listed the words for which $c^* = \arg\max_c P(c|w)$. Figure 2 shows a histogram of the winning assignment probabilities, $\max_c P(c|w)$, for these words. Note that the winning assignment probabilities are distributed broadly over the interval $[\frac{1}{C}, 1]$. This demonstrates the utility of allowing "soft" membership classes: for most words, the maximum likelihood estimates of $P(c|w)$ do not correspond to a winner-take-all assignment, and therefore any method that assigns each word to a single class ("hard" clustering), such as those used by Brown et al. (1992) or Ney, Essen, and Kneser (1994), would lose information.

We conclude this section with some final comments on overfitting. Our models were trained by thirty-two iterations of EM, allowing for nearly complete convergence in the log-likelihood. Moreover, we did not implement any flooring constraints[1] on the probabilities $P(c|w_1)$ or $P(w_2|c)$. Nevertheless, in all our experiments, the ML aggregate Markov models assigned non-zero probability to all the bigrams in the test set. This suggests that for large vocabularies there is a useful regime $1 \ll C \ll V$ in which aggregate models do not suffer much from overfitting. In this regime, aggregate models can be relied upon to compute the probabilities of unseen word combinations. We will return to this point in Section 4, when we consider how to smooth n-gram language models.

3 Mixed-order Markov models

One of the drawbacks of n-gram models is that their size grows rapidly with their order. In this section, we consider how to make predictions based on a convex combination of pairwise correlations. This leads to language models whose size grows linearly in the number of words used for each prediction.

For each $k > 0$, the skip-k transition matrix $M(w_{t-k}, w_t)$ predicts the current word from the kth previous word in the sentence. A mixed-order Markov model combines the information in these matrices for different values of k. Let m denote the number of bigram models being combined. The probability distribution for these models has the form:

$$P(w_t|w_{t-1}, \ldots, w_{t-m}) = \tag{6}$$

---

[1] It is worth noting, in this regard, that individual zeros in the matrices $P(w_2|c)$ and $P(c|w_1)$ do not necessarily give rise to zeros in the matrix $P(w_2|w_1)$, as computed from eq. (1).

$$\sum_{k=1}^{m} \lambda_k(w_{t-k}) \, M_k(w_{t-k}, w_t) \prod_{j=1}^{k-1} [1 - \lambda_j(w_{t-j})].$$

The terms in this equation have a simple interpretation. The $V \times V$ matrices $M_k(w, w')$ in eq. (6) define the skip-k stochastic dependency of w' at some position t on w at position $t - k$; the parameters $\lambda_k(w)$ are mixing coefficients that weight the predictions from these different dependencies. The value of $\lambda_k(w)$ can be interpreted as the probability that the model, upon seeing the word w_{t-k}, looks no further back to make its prediction (Singer, 1996). Thus the model predicts from w_{t-1} with probability $\lambda_1(w_{t-1})$, from w_{t-2} with probability $[1 - \lambda_1(w_{t-1})]\lambda_2(w_{t-2})$, and so on. Though included in eq. (6) for cosmetic reasons, the parameters $\lambda_m(w)$ are actually fixed to unity so that the model never looks further than m words back.

We can view eq. (6) as a hidden variable model. Imagine that we adopt the following strategy to predict the word at time t. Starting with the previous word, we toss a coin (with bias $\lambda_1(w_{t-1})$) to see if this word has high predictive value. If the answer is yes, then we predict from the skip-1 transition matrix, $M_1(w_{t-1}, w_t)$. Otherwise, we shift our attention one word to the left and repeat the process. If after $m - 1$ tosses we have not settled on a prediction, then as a last resort, we make a prediction using $M_m(w_{t-m}, w_t)$. The hidden variables in this process are the outcomes of the coin tosses, which are unknown for each word w_{t-k}.

Viewing the model in this way, we can derive an EM algorithm to learn the mixing coefficients $\lambda_k(w)$ and the transition matrices[2] $M_k(w, w')$. The E-step of the algorithm is to compute, for each word in the training set, the posterior probability that it was generated by $M_k(w_{t-k}, w_t)$. Denoting these posterior probabilities by $\phi_k(t)$, we have:

$$\phi_k(t) = \qquad\qquad\qquad\qquad\qquad (7)$$
$$\frac{\lambda_k(w_{t-k}) M_k(w_{t-k}, w_t) \prod_{j=1}^{k-1} [1 - \lambda_j(w_{t-j})]}{P(w_t | w_{t-1}, w_{t-2}, \ldots, w_{t-m})},$$

where the denominator is given by eq. (6). The M-step of the algorithm is to update the parameters $\lambda_k(w)$ and $M_k(w, w')$ to reflect the statistics in eq. (7). The updates for mixed-order Markov models are given by:

$$\lambda_k(w) \leftarrow \frac{\sum_t \delta(w, w_{t-k}) \phi_k(t)}{\sum_t \sum_{j=k}^{m} \delta(w, w_{t-k}) \phi_j(t)}, \qquad (8)$$

[2] Note that the ML estimates of $M_k(w, w')$ do not depend only on the raw counts of k-separated bigrams; they are also coupled to the values of the mixing coefficients, $\lambda_k(w)$. In particular, the EM algorithm adapts the matrix elements to the weighting of word combinations in eq. (6). The raw counts of k-separated bigrams, however, do give good initial estimates.
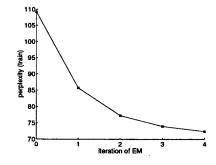

Figure 3: Plot of (training set) perplexity versus number of iterations of the EM algorithm. The results are for the $m = 4$ mixed-order Markov model.

m	train	missing
1	123.2	0.045
2	89.4	0.014
3	77.9	0.0063
4	72.4	0.0037

Table 3: Results for ML mixed-order models; m denotes the number of bigrams that were mixed into each prediction. The first column shows the perplexities on the training set. The second shows the fraction of words in the test set that were assigned zero probability. The case $m = 1$ corresponds to a ML bigram model.

$$M_k(w, w') \leftarrow \frac{\sum_t \delta(w, w_{t-k}) \delta(w', w_t) \phi_k(t)}{\sum_t \delta(w, w_{t-k}) \phi_k(t)}, \quad (9)$$

where the sums are over all the sentences in the training set, and $\delta(w, w') = 1$ iff $w = w'$.

We trained mixed-order Markov models with $2 \leq m \leq 4$. Figure 3 shows a typical plot of the training set perplexity as a function of the number of iterations of the EM algorithm. Table 3 shows the final perplexities on the training set (after four iterations of EM). Mixed-order models cannot be used directly on the test set because they predict zero probability for unseen word combinations. Unlike standard n-gram models, however, the number of unseen word combinations actually *decreases* with the order of the model. The reason for this is that mixed-order models assign finite probability to all n-grams $w_1 w_2 \ldots w_n$ for which *any* of the k-separated bigrams $w_k w_n$ are observed in the training set. To illustrate this point, Table 3 shows the fraction of words in the test set that were assigned zero probability by the mixed-order model. As expected, this fraction decreases monotonically with the number of bigrams that are mixed into each prediction.

Clearly, the success of mixed-order models depends on the ability to gauge the predictive value of each word, relative to earlier words in the same sentence. Let us see how this plays out for the

0.1 < $\lambda_1(w)$ < 0.7
$\langle$-$\rangle$ and of $\langle$"$\rangle$ or $\langle$;$\rangle$ to $\langle$,$\rangle$ $\langle$&$\rangle$ by with S. from nine were for that eight low seven the $\langle$($\rangle$ $\langle$:$\rangle$ six are not against was four between a their two three its $\langle$unknown$\rangle$ B. on as is $\langle$—$\rangle$ five $\langle$)$\rangle$ into C. M. her him over than A.

0.96 < $\lambda_1(w)$ ≤ 1
officials prices which go way he last they earlier an Tuesday there foreign quarter she former federal don't days Friday next Wednesday $\langle$%$\rangle$ Thursday I Monday Mr. we half based part United it's years going nineteen thousand months $\langle$·$\rangle$ million very cents San ago U. percent billion $\langle$?$\rangle$ according $\langle$.$\rangle$

Table 4: Words with low and high values of $\lambda_1(w)$ in an $m = 2$ mixed order model.

C	validation	test	unseen
1	163.615	167.112	293175
2	162.982	166.193	259360
4	161.513	164.363	200067
8	161.327	164.104	190178
16	160.034	162.686	164673
32	159.247	161.683	150958

Table 5: Perplexities of bigram models smoothed by aggregate Markov models with different numbers of classes (C).

second-order ($m = 2$) model in Table 3. In this model, a small value for $\lambda_1(w)$ indicates that the word w typically carries less information that the word that precedes it. On the other hand, a large value for $\lambda_1(w)$ indicates that the word w is highly predictive. The ability to learn these relationships is confirmed by the results in Table 4. Of the three-hundred most common words, Table 4 shows the fifty with the lowest and highest values of $\lambda_1(w)$. Note how low values of $\lambda_1(w)$ are associated with prepositions, mid-sentence punctuation marks, and conjunctions, while high values are associated with "contentful" words and end-of-sentence markers. (A particularly interesting dichotomy arises for the two forms "a" and "an" of the indefinite article; the latter, because it always precedes a word that begins with a vowel, is inherently more predictive.) These results underscore the importance of allowing the coefficients $\lambda_1(w)$ to depend on the context w, as opposed to being context-independent (Ney, Essen, and Kneser, 1994).

4 Smoothing

Smoothing plays an essential role in language models where ML predictions are unreliable for rare events. In n-gram modeling, it is common to adopt a recursive strategy, smoothing bigrams by unigrams, trigrams by bigrams, and so on. Here we adopt a similar strategy, using the $(m - 1)$th mixed-order model to smooth the mth one. At the "root" of our smoothing procedure, however, lies not a unigram model, but an aggregate Markov model with $C > 1$ classes. As shown in Section 2, these models assign finite probability to all word combinations, even those that are not observed in the training set. Hence, they can legitimately replace unigrams as the base model in the smoothing procedure.

Let us first examine the impact of replacing unigram models by aggregate models at the root of the smoothing procedure. To this end, a held-out interpolation algorithm (Jelinek and Mercer, 1980) was used to smooth an ML bigram model with the aggregate Markov models from Section 2. The smoothing parameters, one for each row of the bigram transition matrix, were estimated from a validation set the same size as the test set. Table 5 gives the final perplexities on the validation set, the test set, and the unseen bigrams in the test set. Note that smoothing with the $C = 32$ aggregate Markov model has nearly halved the perplexity of unseen bigrams, as compared to smoothing with the unigram model.

Let us now examine the recursive use of mixed-order models to obtain smoothed probability estimates. Again, a held-out interpolation algorithm was used to smooth the mixed-order Markov models from Section 3. The mth mixed-order model had mV smoothing parameters $\sigma_k(w)$, corresponding to the V rows in each skip-k transition matrix. The mth mixed-order model was smoothed by discounting the weight of each skip-k prediction, then filling in the leftover probability mass by a lower-order model. In particular, the discounted weight of the skip-k prediction was given by

$$[1 - \sigma_k(w_{t-k})]\lambda_k(w_{t-k}) \prod_{j=1}^{k-1}[1 - \lambda_j(w_{t-j})] \quad , \quad (10)$$

leaving a total mass of

$$\sum_{k=1}^{m} \sigma_k(w_{t-k})\lambda_k(w_{t-k}) \prod_{j=1}^{k-1}[1 - \lambda_j(w_{t-j})] \quad (11)$$

for the $(m - 1)$th mixed-order model. (Note that the $m = 1$ mixed-order model corresponds to a ML bigram model.)

Table 6 shows the perplexities of the smoothed mixed-order models on the validation and test sets. An aggregate Markov model with $C = 32$ classes was used as the base model in the smoothing procedure. The first row corresponds to a bigram model smoothed by a aggregate Markov model; the second row corresponds to an $m = 2$ mixed-order model, smoothed by a ML bigram model, smoothed by an aggregate Markov model; the third row corresponds

86

m	validation	test
1	160.1	161.3
2	135.3	136.9
3	131.4	133.5
4	131.2	133.7

Table 6: Perplexities of smoothed mixed-order models on the validation and test sets.

to an $m = 3$ mixed-order model, smoothed by a $m = 2$ mixed-order model, smoothed by a ML bigram model, etc. A significant decrease in perplexity occurs in moving to the smoothed $m = 2$ mixed-order model. On the other hand, the difference in perplexity for higher values of m is not very dramatic.

Our last experiment looked at the smoothing of a trigram model. Our baseline was a ML trigram model that backed off [3] to bigrams (and when necessary, unigrams) using the Katz backoff procedure (Katz, 1987). In this procedure, the predictions of the ML trigram model are discounted by an amount determined by the Good-Turing coefficients; the leftover probability mass is then filled in by the backoff model. We compared this to a trigram model that backed off to the $m = 2$ model in Table 6. This was handled by a slight variant of the Katz procedure (Dagan, Pereira, and Lee, 1994) in which the mixed-order model substituted for the backoff model.

One advantage of this smoothing procedure is that it is straightforward to assess the performance of different backoff models. Because the backoff models are only consulted for unseen word combinations, the perplexity on these word combinations serves as a reasonable figure-of-merit.

Table 7 shows those perplexities for the two smoothed trigram models (baseline and backoff). The mixed-order smoothing was found to reduce the perplexity of unseen word combinations by 51%. Also shown in the table are the perplexities on the entire test set. The overall perplexity decreased by 16%—a significant amount considering that only 24% of the predictions involved unseen word combinations and required backing off from the trigram model.

The models in Table 7 were constructed from all n-grams $(1 \leq n \leq 3)$ observed in the training data. Because many n-grams occur very infrequently, a natural question is whether truncated models, which omit low-frequency n-grams from the training set, can perform as well as untruncated ones. The advantage of truncated models is that they do not need to store nearly as many non-zero parameters as untruncated models. The results in Table 8 were ob-

---

[3]We used a backoff procedure (instead of interpolation) to avoid the estimation of trigram smoothing parameters.

backoff	test	unseen
baseline	95.2	2799
mixed	79.8	1363

Table 7: Perplexities of two smoothed trigram models on the test set and the subset of unseen word combinations. The baseline model backed off to bigrams and unigrams; the other backed off to the $m = 2$ model in Table 6.

t	baseline	mixed	trigrams($\times 10^6$)	missing
1	95.2	79.8	25.4	0.24
2	98.6	78.3	6.1	0.32
3	101.7	79.6	3.3	0.36
4	104.2	81.1	2.3	0.38
5	106.2	82.4	1.7	0.41

Table 8: Effect of truncating trigrams that occur less than t times. The table shows the baseline and mixed-order perplexities on the test set, the number of distinct trigrams with t or more counts, and the fraction of trigrams in the test set that required backing off.

tained by dropping trigrams that occurred less than t times in the training corpus. The $t = 1$ row corresponds to the models in Table 7. The most interesting observation from the table is that omitting very low-frequency trigrams does not decrease the quality of the mixed-order model, and may in fact slightly improve it. This contrasts with the standard backoff model, in which truncation causes significant increases in perplexity.

5 Discussion

Our results demonstrate the utility of language models that are intermediate in size and accuracy between different order n-gram models. The two models considered in this paper were hidden variable Markov models trained by EM algorithms for maximum likelihood estimation. Combinations of intermediate-order models were also investigated by Rosenfeld (1996). His experiments used the 20,000-word vocabulary *Wall Street Journal* corpus, a predecessor of the NAB corpus. He trained a maximum-entropy model consisting of unigrams, bigrams, trigrams, skip-2 bigrams and trigrams; after selecting long-distance bigrams (word triggers) on 38 million words, the model was tested on a held-out 325 thousand word sample. Rosenfeld reported a test-set perplexity of 86, a 19% reduction from the 105 perplexity of a baseline trigram backoff model. In our experiments, the perplexity gain of the mixed-order model ranged from 16% to 22%, depending on the amount of truncation in the trigram model.

While Rosenfeld's results and ours are not di-

rectly comparable, both demonstrate the utility of mixed-order models. It is worth discussing, however, the different approaches to combining information from non-adjacent words. Unlike the maximum entropy approach, which allows one to combine many non-independent features, ours calls for a careful Markovian decomposition. Rosenfeld argues at length against naïve linear combinations in favor of maximum entropy methods. His arguments do not apply to our work for several reasons. First, we use a large number of context-dependent mixing parameters to optimize the overall likelihood of the combined model. Thus, the weighting in eq. (6) ensures that the skip-k predictions are only invoked when the context is appropriate. Second, we adjust the predictions of the skip-k transition matrices (by EM) so that they match the contexts in which they are invoked. Hence, the count-based models are interpolated in a way that is "consistent" with their eventual use.

Training efficiency is another issue in evaluating language models. The maximum entropy method requires very long training times: e.g., 200 CPU-days in Rosenfeld's experiments. Our methods require significantly less; for example, we trained the smoothed $m = 2$ mixed-order model, from start to finish, in less than 12 CPU-hours (while using a larger training corpus). Even accounting for differences in processor speed, this amounts to a significant mismatch in overall training time.

In conclusion, let us mention some open problems for further research. Aggregate Markov models can be viewed as approximating the full bigram transition matrix by a matrix of lower rank. (From eq. (1), it should be clear that the rank of the class-based transition matrix is bounded by the number of classes, C.) As such, there are interesting parallels between Expectation-Maximization (EM), which minimizes the approximation error as measured by the KL divergence, and singular value decomposition (SVD), which minimizes the approximation error as measured by the L_2 norm (Press et al., 1988; Schütze, 1992). Whereas SVD finds a global minimum in its error measure, however, EM only finds a local one. It would clearly be desirable to improve our understanding of this fundamental problem.

In this paper we have focused on bigram models, but the ideas and algorithms generalize in a straightforward way to higher-order n-grams. Aggregate models based on higher-order n-grams (Brown et al., 1992) might be able to capture multi-word structures such as noun phrases. Likewise, trigram-based mixed-order models would be useful complements to 4-gram and 5-gram models, which are not uncommon in large-vocabulary language modeling.

A final issue that needs to be addressed is scaling—that is, how the performance of these models depends on the vocabulary size and amount of training data. Generally, one expects that the sparser the data, the more helpful are models that can intervene between different order n-grams. Nevertheless, it would be interesting to see exactly how this relationship plays out for aggregate and mixed-order Markov models.

Acknowledgments

References

P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18(4):467–479.

S. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Meeting of the Association for Computational Linguistics*.

I. Dagan, F. Pereira, and L. Lee. 1994. Similarity-based estimation of word co-occurrence probabilities. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* B39:1–38.

X. Huang, F. Alleva, H. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld. 1993. The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 2:137–148.

F. Jelinek and R. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*.

F. Jelinek, R. Mercer, and S. Roukos. 1992. Principles of lexical language modeling for speech recognition. In S. Furui and M. Sondhi, eds. *Advances in Speech Signal Processing*. Mercer Dekker, Inc.

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on ASSP* 35(3):400–401.

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

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics.*

W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. 1988. *Numerical Recipes in C.* Cambridge University Press: Cambridge.

R. Rosenfeld. 1996. A Maximum Entropy Approach to Adaptive Statistical Language Modeling. *Computer Speech and Language*, 10:187–228.

H. Schütze. 1992. Dimensions of Meaning. In *Proceedings of Supercomputing*, 787–796. Minneapolis MN.

Y. Singer. 1996. Adaptive Mixtures of Probabilistic Transducers. In D. Touretzky, M. Mozer, and M. Hasselmo (eds). *Advances in Neural Information Processing Systems* 8:381–387. MIT Press: Cambridge, MA.