DISTRIBUTED ACOUSTIC MODELING WITH BACK-OFF N-GRAMS

Ciprian Chelba, Peng Xu, Fernando Pereira

Google, Inc.,
1600 Amphiteatre Pkwy,
Mountain View, CA 94043, USA

Thomas Richardson

Statistics Department
Box 354322, University of Washington,
Seattle, WA 98195, USA

ABSTRACT

The paper proposes an approach to acoustic modeling that borrows from n-gram language modeling in an attempt to scale up both the amount of training data and model size (as measured by the number of parameters in the model) to approximately 100 times larger than current sizes used in ASR.

Dealing with unseen phonetic contexts is accomplished using the familiar back-off technique used in language modeling due to implementation simplicity. The new acoustic model is estimated and stored using the MapReduce distributed computing infrastructure.

Speech recognition experiments are carried out in an N-best rescoring framework for Google Voice Search. 87,000 hours of training data is obtained in an unsupervised fashion by filtering utterances in Voice Search logs on ASR confidence. The resulting models are trained using maximum likelihood and contain 20-40 million Gaussians. They achieve relative reductions in WER of 11% and 6% over first-pass models trained using maximum likelihood, and boosted MMI, respectively.

1. INTRODUCTION

The most common technique in dealing with data sparsity when estimating context-dependent HMMs in automatic speech recognition (ASR) is the well known decision tree (DT) state clustering approach [1]. To make sure the clustered states have enough data for reliable estimation, the algorithm guarantees a minimum number of frames at each context dependent state (leaf of the DT). At the other end of the spectrum, states for which there is a lot more training data than the minimum can have more mixture components. An effective way of sizing the GMM as a function of the number of frames n in training data is the *varmix* rule [2]:

$$\log(no.\ mix\ components) \quad = \quad \log(\beta) + \alpha \cdot \log(n) \quad (1)$$

Typical amounts of training data used for the acoustic model (AM) in ASR vary from 100 to 1,000 hours. The frame rate in most systems is 10 milliseconds, which means that about 360 million samples are used to train the 0.5 million or so Gaussians in an ASR system. Assuming that $n = 1,000$ frames are sufficient for robustly estimating a single Gaussian, a back-of-the-envelope calculation shows that 1,000 hours of speech would allow for a system with about 0.36 million Gaussians—quite close to values encountered in ASR practice, see Section 4.1, or Table VI in [2]. We can thus say that current AMs achieve *"estimation efficiency"*.

Recent applications have led to availability of data far beyond those used in ASR systems. Filtering Google Voice Search logged utterances at an adequate ASR confidence threshold guarantees transcriptions that are close to human annotator performance, e.g. we can obtain 87,000 hours of transcribed speech at ASR confidence of 0.8 or higher. If we are to strive for *"estimation efficiency"* then this much speech data would allow estimation of an AM whose size is about 40 million Gaussians.

As a first direction we chose to use longer context: the phonetic context for an HMM state is determined by M context independent phones to the left and right of the current phone and state. We experimented with values for $M = 1 \ldots 5$—thus reaching the equivalent of 11-phones. For such large values of M not all *M-phones* (context dependent HMM states in our model) can be reliably estimated and thus saved in the model. We deal with unseen *M-phones* by backing-off, similar to what is done in n-gram language modeling: the context for an unseen *M-phone* encountered on test data is decreased gradually until we reach an *M-phone* that is present in the model.

The next section describes our approach to increasing the state space using back-off acoustic modeling (BAM), and explains why standard DT state-tieing does not easily scale to such large amounts of data. Section 3 describes the BAM implementation using Google's distributed infrastructure. Section 4 presents our experiments in an N-best rescoring framework, followed by conclusions.

2. BACK-OFF N-GRAMS FOR ACOUSTIC MODELING

Consider a short utterance whose transcription is:
W = `<S> action </S>`, and assume the pronunciation lexicon provides the following mapping to context-

independent (CI) phones `sil eh k sh ih n sil`. `<S>`, `</S>` denote sentence boundaries, pronounced as long silence `sil`.

A typical triphone approach would model the 3 states of `ih` as `sh-ih+n_{1,2,3}` using the decision tree clustering algorithm for tieing each of the 3 states across various instances `*-ih+*_{1,2,3}`, respectively. This yields the so-called context-dependent states in the HMM.

In contrast, BAM with $M = 3$ extracts the following training data instances for the maximal order *M-phone*, as well as the back-off ones:

```
ih_1 / eh k sh ___ n sil     frames
ih_1 /    k sh ___ n sil     frames
ih_1 /       sh ___ n        frames
```

for the first HMM state of the `ih` instance in the example utterance above.

To achieve this we first compute the context-dependent state level Viterbi alignment between transcription W and speech feature frames using the transducer composition $H \circ C \circ L \circ W$, where L, C, H denote respectively the pronunciation lexicon, context dependency tree, and HMM-to-state FST transducers [3]. From the alignment we then extract modeling units called *M-phones* along with the corresponding speech feature `frames`. Each *M-phone* is uniquely identified by its *key*, e.g. `ih_1 / eh k sh ___ n sil`. The key is a string representation obtained by joining on `/` the CI-state, i.e. `ih_1` above and the surrounding phonetic context, in this case `eh k sh ___ n sil`; `___` is a placeholder marking the position where the central CI-state `ih_1` occurs in the context.

Besides the maximal order *M-phones* we also collect *back-off M-phones*, as outlined above. There are other possible back-off strategies, but we currently implement only the one above: if the *M-phone* is symmetric (same left and right context length) back-off at both ends; if not, then back-off from the longer end until the *M-phone* is symmetric, and proceed with symmetric back-offs from there on. With each back-off we clone the speech feature frames from the maximal order *M-phone* to the back-off one. We found it useful to augment the CI-phones with word boundaries, which has its own symbol, and occupies its own context position.

2.1. Comparison with Existing Approaches

BAM can be viewed as a simplified DT state clustering algorithm that uses question sets consisting of atomic context independent phones, queried in a pre-defined order. The approach is not novel, see [4], and it is likely suboptimal but we prefer it to DT AMs for ease of implementation in MapReduce.

Our approach to obtaining large amounts of training data is very similar to [2]. Table VI there highlights the gains from using increasing amounts of training data (from 375 hours up to 2,210 hours), and shows that past 1,350 hours a system with 9K states and about 300k Gaussians gets diminishing returns in recognition accuracy. Our approach allows both the use of significantly more training data and estimation of much larger models: in our experiments we used 87,000 hours of training data and built models of up to 1.1 million states and 40 million Gaussians.

3. DISTRIBUTED ACOUSTIC MODELING

3.1. BAM Estimation Using MapReduce

BAM estimation and run-time are implemented using MapReduce [5] and SSTable (immutable persistent B-tree[1]), and draws heavily from the large language modeling approach for statistical machine translation [6].

Each *Mapper* instance processes a chunk of the input data, one record at a time. Each record consists of a key-value pair; the value stores the waveform, the word level transcript for the utterance, and other elements. For each record arriving at the *Mapper* we: generate context-dependent-state level Viterbi alignment after composing $H \circ C \circ L$ with the transcript W; extract maximal order *M-phones* and output $< M-phone\ key,\ frames >$ pairs; compute back-off *M-phones* and output $< M-phone\ key,\ < empty >>$ pairs.

After shuffling, *M-phones* have their frame data (if they carry any) collated and presented to the *Reducer* along with the *M-phone* key. Every time a maximal order *M-phone* arrives at the reducer we estimate and output a GMM from its data (assuming the number of frames is above the lower threshold), and also accumulate its data in the reservoirs for all of its back-off *M-phones*—which are cached in a stack until all instances of that back-off *M-phone* have arrived at the *Reducer* and we can estimate and output its GMM.

For each *M-phone* that meets a lower threshold on the number of frames aligned against it we estimate a GMM using the standard splitting algorithm [7], following the *varmix* rule to size the GMM. The *M-phones* that have more frames than an upper threshold on the number frames (256k) are estimated using reservoir sampling. Variances that become too low are floored to a small value (0.00001). The resulting SSTable stores the BAM, a distributed (partitioned) associative array $< M-phone\ key,\ GMM >$.

3.2. BAM Test Run-time

At test time we rescore 10-best lists for each utterance using BAM. We load the model into an in-memory key-value serving system (SSTable service) with S servers each holding $1/S$-th of the data. For each hypothesis in the 10-best list for an input utterance, we: generate context-dependent state level Viterbi alignment after composing $H \circ C \circ L$ with the transcript W; extract maximal order *M-phones*; compute back-

[1] A format similar to SSTable has been open-sourced as part of the LevelDB project http://code.google.com/p/leveldb/

off *M-phones*; add all *M-phones* to a pool initialized once for each input record (utterance).

Once the pool is finalized, it is sent as a batch request to the SSTable service. The *M-phones* that are actually stored in the model are returned to the *Mapper*, and are used to rescore the alignment for each of the hypotheses in the 10-best list. For each segment in the alignment we use the highest order *M-phone* that was retrieved from the BAM SSTable. If no back-off *M-phones* are retrieved for a given segment, we back-off to the first pass AM score for that segment—already computed during Viterbi alignment.

To penalize the use of lower order *M-phones*, we incur a per-frame back-off cost $fbo \times (M - o) > 0.0$ when rescoring a segment with an *M-phone* of lower order $o \geq 0$ than the maximum one M; the order of an asymmetric *M-phone* is computed as the maximum of the left and right context lengths. When the model backs-off all the way to using the first pass AM (clustered state), $o = 0$.

The final AM score for each hypothesis is then computed by log-linear interpolation between the first pass AM, and the second pass one (BAM, or first pass AM if running sanity checks, see Table 1):

$$\begin{aligned} \log P_{AM}(A|W) &= \lambda \cdot \log P_{first\ pass}(A|W) + \\ &\quad (1.0 - \lambda) \cdot \log P_{second\ pass}(A|W) \end{aligned} \tag{2}$$

where A denotes the acoustic features, and W denotes the word sequence.

4. EXPERIMENTS

We ran our experiments on Google Voice Search training and test data. There are two training sets that we used in our experiments:

- *ML baseline*: 1 million manually transcribed Voice Search spoken queries—approx. 1,000 hours of speech
- *filtered logs*: 110 million Voice Search spoken queries + 1-best ASR transcript, filtered by confidence at 0.8 threshold—approx. 87,000 hours of speech. The whole query-level confidence measure used for filtering is derived using standard lattice-based word posteriors.

As development/test data we used two sets of manually transcribed data that do not overlap with the training data (the utterances originate from non-overlapping time periods in our logs). Let's denote them as data sets T9b/T9a, consisting of 27,273/26,722 spoken queries (87,360/84,918 words), respectively. All query data used in the experiments (training/development/test) is anonymized.

4.1. First Pass AMs

The first pass AM is estimated on the *ML baseline* data in the usual staged approach after extracting 39 dimensional features for every frame:

1. 3-state, context independent phone HMMs with single Gaussian, diagonal covariance output distributions
2. standard decision tree clustering for triphones, 8k context-dependent states
3. GMM splitting up to 0.33 million diagonal covariance Gaussians; the minimum/maximum number of frames for a given context-dependent state is 18k/256k respectively; states with more than the maximum number of frames are estimated by sampling randomly down to 256k frames; *varmix* estimation is used to determine the number of mixtures according to the amount of training data
4. boosted-MMI training [8] on the *ML baseline* data augmented with another 10 million Voice Search spoken queries + 1-best ASR transcript, filtered by confidence.

4.2. N-best Rescoring Experiments

The T9b development data is used to optimize the model order $M = 1 \ldots 5$ (triphones to 11-phones); AM weight in log-linear mixing of first pass AM scores with the rescoring AM, λ; language model weight: lmw; and the per frame back-off weight: fbo. Across all experiments we kept constant the baseline AM (in all cases the ML one trained on the *ML baseline* data) and the maximum number of frames for an *M-phone* state (256k). For the $\alpha/\beta = 0.3/2.2$ setting this means a maximum number of 92 mixture components per state.

Table 1 shows the results when rescoring 10-best with BAM, along with the best settings as estimated on development data. We built models for $M = 1, \ldots 5$ but as the results show there was no gain in performance for values of $M > 2$.

The first three rows show the performance, and size (in number of Gaussians) of the maximum likelihood AM baseline (stage 3 in Section 4.1) on the test set T9a. Somewhat surprisingly, there is a small gain (0.3% absolute) obtained by interpolating the first and second pass scores produced by the ML baseline AM for the same utterance. We point out this oddity because the same second pass alignments are rescored with the BAM, and hence this small improvement should not be credited to better modeling using BAM, but rather to the re-computation of alignments in the second pass.

The first training regime for BAM used the same data as that used for the ML part of the baseline AM training sequence. When matching the threshold on number of frames with the one used for the baseline AM (18k), BAM ends up with fewer Gaussians than the baseline AM—223k vs. 327k. This is not surprising, since no decision tree clustering is done, and the data is not used as effectively—many triphones/*1-phones* are discarded, along with their data. However, its performance matches that of the baseline AM—in a 10-best rescoring setup; no claims are made about the performance in the first pass. Lowering the threshold on the number

Model	WER (Sub/Del/Ins)	No. Gaussians (39 dim, diag cov)
TRAINING DATA = ML baseline data (1k hours)		
ML baseline AM, $\lambda = 0.0$, $lmw = 17$	12.4 (1.3/2.5/8.6)	327,438
ML baseline AM, $\lambda = 0.6$, $lmw = 17$	*11.6* (1.2/2.3/8.1)	327,438
ML baseline AM, $\lambda = 1.0$ (1-st pass), $lmw = 17$	11.9 (1.2/2.4/8.3)	327,438
TRAINING DATA = ML baseline data (1k hours)		
BAM (min no. frames=18k, $M = 1$, $\lambda = 0.8$, $lmw = 17$, $fbo = 0.0$)	11.6 (1.2/2.2/8.2)	223,211
BAM (min no. frames=4k, $M = 1$, $\lambda = 0.8$, $lmw = 17$, $fbo = 0.0$)	11.5 (1.2/2.2/8.1)	489,640
TRAINING DATA = 1% filtered logs data (1k hours)		
BAM (min no. frames=4k, $M = 2$, $\lambda = 0.8$, $lmw = 17$, $fbo = 1.0$)	11.3 (1.2/2.2/7.9)	600,291
TRAINING DATA = 10% filtered logs data (9k hours)		
BAM (min no. frames=4k, $M = 2$, $\lambda = 0.6$, $lmw = 17$, $fbo = 0.4$)	10.9 (1.1/2.2/7.7)	3,974,917
TRAINING DATA = 100% filtered logs data (87k hours)		
BAM (min no. frames=4k, $M = 2$, $\lambda = 0.6$, $lmw = 17$, $fbo = 0.0$)	**10.6 (1.0/2.2/7.4)**	22,210,429

Table 1. Maximum Likelihood Back-off Acoustic Model (BAM) Results on the Test Set T9a, 10-best Rescoring

of frames to 4k (26 mixture components at $\alpha/\beta = 0.3/2.2$) does increase the number of Gaussians in the model to 490k.

The second training regime for BAM uses the *filtered logs* data, in varying amounts: 1%, 10%, 100%, respectively. A surprising result is that switching from manually annotated data to the same amount of confidence filtered data provides a small absolute WER gain of 0.1-0.2%. This shows that the confidence filtered data is just as good as the manually annotated data for training AMs to be used in N-best rescoring.

From then on, BAM steadily improves as we add more *filtered logs* data: the first 10X increase brings a 0.4-0.5% absolute WER reduction, and the second 10X increase brings a 0.3% absolute WER reduction. This amounts to 1.3% absolute reduction (11% relative) on the one-pass baseline of 11.9% WER.

When switching to using the boosted MMI AM (stage 4 in Section 4.1) as the first-pass AM in both training and test, the baseline result is significantly better at 9.8% WER. Despite the fact that it is not discriminatively trained BAM provides a 0.6% (6% relative) reduction in WER over the MMI baseline.

5. CONCLUSIONS AND FUTURE WORK

We find these results very encouraging, and think that distributed acoustic modeling is something to look into for improving ASR. Expanding phonetic context is not really productive: "more model" by increasing $M > 2$ yields no gain in accuracy, so we still need to find good ways of using large amounts of data.

Obvious future work items that are perfectly feasible at this scale include: DT state tieing, re-computing alignments in ML training, and discriminative GMM training. On the more exploratory side, non-parametric modeling techniques hold promise with such large amounts of training data.

Acknowledgments

Many thanks to my colleagues that helped with comments, suggestions, and solving various speech infrastructure issues, in particular: Alex Gruenstein, Brian Strope, Doug Beeferman, Erik McDermott, Jeff Dean, Johan Schalkwyk, Michiel Bacchiani, Thorsten Brants, Vincent Vanhoucke, and Will Neveitt. Special thanks go to Olivier Siohan for help with prompt code reviews and detailed comments.

6. REFERENCES

[1] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings ARPA Workshop on Human Language Technology*, Berlin, 1994, pp. 307–312.

[2] M.J.F. Gales et al., "Progress in the CU-HTK broadcast news transcription system," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 5, pp. 1513 –1525, September 2006.

[3] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[4] Rich Schwartz et al., "Improved Hidden Markov modeling of phonemes for continuous speech recognition," in *Proceedings of ICASSP*, 1984, vol. 9, pp. 21–24.

[5] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008.

[6] Thorsten Brants et al., "Large language models in machine translation," in *Proceedings of the Joint Conference EMNLP-CoNLL*, 2007, pp. 858–867.

[7] Steve Young et al., *The HTK Book*, Cambridge University Engineering Department, Cambridge, England, December 2002.

[8] Dan Povey et al., "Boosted MMI for model and feature space discriminative training," in *Proceedings of ICASSP*, April 2008, pp. 4057 –4060.