SPEEDING UP SOFTMAX COMPUTATIONS IN DNN-BASED LARGE VOCABULARY SPEECH RECOGNITION BY SENONE WEIGHT VECTOR SELECTION

Yingke Zhu and Brian Mak

Department of Computer Science & Engineering
The Hong Kong University of Science & Technology
{yzhuav,mak}@cse.ust.hk

ABSTRACT

Deep neural network has obtained significant accuracy improvement in many large vocabulary continuous speech recognition (LVCSR) tasks. Recently, it was shown that even better performance can be obtained by modeling a larger number of more discriminative senones. However, as the neural network becomes larger, the number of parameters increases greatly, resulting in greater computation cost and slower decoding process. Since in LVCSR systems, most DNN computations are done in the output softmax layer, we propose a senone weight vector selection method in this paper to speed up the DNN softmax computation while keeping the system accuracy more or less the same. We apply clustering on the weight vectors of the softmax layer and group all the senone weight vectors into several clusters. During decoding, we only compute the exact posteriors for senones in the selected clusters. For the senones in the unselected clusters, their posteriors are approximated using their cluster centers. Experimental results show that our speed-up method can reduce DNN computation time by more than 35% with negligible accuracy loss in a DNN model with 60,000 senones on Switchboard.

Index Terms— decoding speedup, large vocabulary speech recognition, deep neural networks, weight vector selection

1. INTRODUCTION

In the last decade, deep learning has been proved very successful in many areas including the field of automatic speech recognition (ASR). As of today, the acoustic models of most state-of-the-art ASR systems are hidden Markov models (HMMs) in which state posteriors are modeled by some form of deep neural network (DNN) [1, 2]. However, DNN-HMMs usually have more parameters than traditional Gaussian-mixture model-HMMs (GMM-HMMs) due to the large number of hidden units in many hidden layers in the DNN, and the direct modeling of senones in the output softmax layer. Recently, it was also found that better recognition performance can be obtained by modeling finer acoustic details with more senones. For example, [3] and [4] used 32K and 60K senones in their Switchboard (SWB) ASR systems respectively; [5] used 44K senones in the YouTube video transcription task. The better performance of these systems with large number of senones comes at a price: a significant increase in their decoding time. In these systems, most — about 80% — DNN computations are done in the output softmax layer[1]. Therefore, it will be beneficial to fast decoding if we can reduce the computation cost in the output layer without affecting the recognition performance significantly.

There are already many decoding speedup methods that make use of model quantization [6], model compression [7] and special integer and floating-point instruction sets [8]. In the filed of language modeling, there are also works aiming to reduce the computational complexity of large neural networks [9, 10, 11]. In this paper, we investigate a speedup method on the senone posterior computations that is inspired by the following two observations.

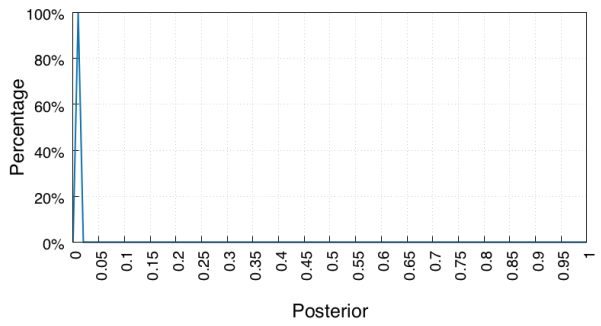
Firstly, it is well-known that although the total number of senones in a LVCSR system can be quite large, they are not all needed during decoding at each frame. In fact, the number of active states (senones) can be much smaller than the entire set of senones. For example, from our analysis, the average number of active states during Viterbi decoding in the Switchboard task is only about 2K–3K for a DNN model with 8704 senones, and for a model with 60K distinct triphone states (DTSs), the figure is about 3K–4K. Thus, we see that the number of active states increases only modestly as the number of senone grows: from about 25–30% for a typical model with 9K senones to about 5–6% in the DTS model with 60K senones[2]. This opens a possibility of evaluating only a small fraction of senone posteriors to speedup decoding.

In [6], a lazy evaluation method was proposed to take advantage of this observation. It computes DNN activations up to the last hidden layer in a dedicated thread, and the output senone posteriors of the softmax layer are computed only when needed by the decoder in a separate thread. The lazy evaluation means that common batching to produce all senone posteriors over several frames cannot be done and it also cannot be done efficiently using matrix multiplications; thus, it introduces a small fixed cost of about 22%. Moreover, lazy evaluation can lead to a small recognition delay. One solution is to use batched lazy evaluation. It makes use of the piece-wise stationary nature of speech signals. That is, if a state is active at frame t, it is very likely that it is also active at frame $t + 1$. As a result, one may assume a set of active states remain active across several consecutive frames, and compute their posteriors in a batch. Obviously, the batch size will be a trade-off between computation efficiency and recognition latency.

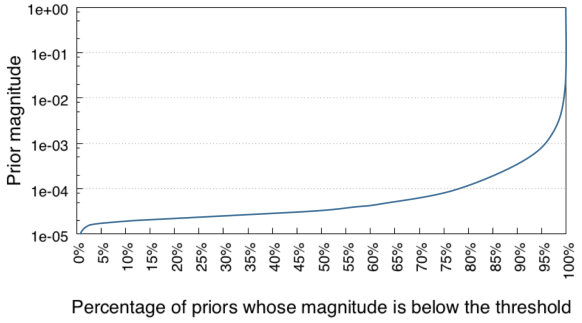
Secondly, we find that the distribution of senone posteriors produced by a DNN is usually very sharp, meaning that many senones have very low posterior probabilities. Figure 1 shows the distribution of state posteriors per frame and state priors for the Switchboard task. We can see that over 99.9% of the state posteriors have a value below 0.01. If one only looks at the posterior distribution, it may seem that one needs only compute a few senone posteriors. However, since the Viterbi decoder works with likelihoods, the senone

[1] In contrast, the amount of DNN computation in the output layer is about 50% in a typical Switchboard DNN model having about 9K senones.

[2] Note that the lower hidden layers in a DNN are shared across all the senones. Thus, they need to be computed even only a few senones/states in the output layer are active during decoding.



(a) State Posterior Distribution



(b) State Prior Distribution

Fig. 1. A sample distribution of state posteriors and priors in a DNN-HMM.

posteriors need to be converted to scaled state likelihoods by dividing them by the senone priors. Fig. 1(b) shows that the prior distribution is far from uniform, and around 80% of the state priors are also very small with a value less than 10^{-4}. It turns out that after considering the state priors, the distribution of state likelihoods will not be as sharp as the posteriors'. This also explains the large discrepancy between the number of active states and the number of very few states with 'large' posteriors.

In this paper, based on the above two observations, we propose a new method to speed up the senone posterior computation. Inspired by Gaussian selection method [12, 13] in the speedup of GMM-HMMs, we propose a softmax weight vector selection method that computes the exact posteriors only for senones in a subset of clusters selected by the inputs to the softmax units; for senones that are not selected, their posteriors are approximated.

The rest of the paper is organized as follows. Section 2 gives a brief review of DNN-HMMs as they are used in speech recognition. Section 3 describes in detail our proposed method in speeding up the senone posterior computations. Experimental results are presented in Section 4, and we conclude our work in Section 5.

2. DNN-HMM IN SPEECH RECOGNITION

The hybrid DNN-HMM combines the discriminative modeling power of DNN with the sequential modeling power of HMM. Compared with the former GMM-HMM model for ASR, the DNN in a DNN-HMM replaces the role of GMM in GMM-HMM to compute the HMM state likelihoods for decoding as follows.

The DNN in the hybrid model takes an observation $\mathbf{x}$, which usually consists of several contextual frames of acoustic features as input, and performs a series of nonlinear transformations when it is propagated forward through L layers of perceptrons. At the lth hidden layer, where $1 \leq l \leq L-1$, we have

$$h_i^l = \sigma(z_i^l) = \sigma\left((\mathbf{w}_i^l)' \cdot \mathbf{v}^l + b_i^l\right), \tag{1}$$

where b_i^l, z_i^l, and h_i^l are the bias, excitation and output of its ith neuron; $\mathbf{v}^l = \mathbf{h}^{l-1}$ is the input vector to the lth hidden layer; $\mathbf{w}_i^l$ is the weight vector associated with the ith neuron; $\sigma(x) = 1/\left(1+e^{-x}\right)$ is the sigmoid function. A softmax layer, the Lth layer, is then stacked onto the $L-1$ hidden layers. The softmax layer consists of J units, representing the set of HMM senones $s_j \in \{s_1, s_2, \ldots, s_J\}$. The softmax layer outputs are computed as estimates of the posterior probabilities of the HMM senones:

$$p(s_j|\mathbf{x}) = p(s_j|\mathbf{v}^L) = \frac{\exp(\mathbf{w}_j'\mathbf{v}^L)}{\sum_{i=1}^{J}\exp(\mathbf{w}_i'\mathbf{v}^L)}, \tag{2}$$

where $\mathbf{w}_j$ and $\mathbf{w}_i$ are the weight vector associated with senone s_j and s_i respectively.

DNN training is typically done by minimizing the cross entropy between the state distributions given by the state alignments of the training data and the DNN outputs using the backpropagation algorithm. Using the Bayes' rule, the senone posterior probabilities are then converted to senone likelihoods as follows:

$$p(\mathbf{x}|s_j) = \frac{p(s_j|\mathbf{x})}{p(s_j)} \cdot p(\mathbf{x}). \tag{3}$$

Since $p(\mathbf{x})$ is the same for all senones, only the scaled senone likelihoods $\frac{p(s_j|\mathbf{x})}{p(s_j)}$ are actually computed and used in Viterbi decoding.

3. SENONE WEIGHT VECTOR SELECTION

The big difference between the number of senones and the number of active states opens the possibility to speed up state likelihood computations. The basic idea is that if the active states in the next frame are known, one may save a lot of computations by computing the exact likelihoods only for those active states and ignore the remaining states. However, since the following active states are not known in advance, one needs an evaluation function for their prediction. In the past, when GMM-HMMs were used in ASR, Gaussian selection is a common technique for fast likelihood computations. In Gaussian selection, all Gaussians in a GMM-HMM system are clustered into a number of Gaussian clusters represented by their cluster means. During recognition, an input acoustic vector is compared with the cluster means to find the one that is closest to it. Only the likelihoods of those Gaussians inside or in some neighborhood of the closest cluster are computed. By carefully tuning the number of Gaussian clusters and the size of their neighborhoods, the number of Gaussians that are actually evaluated can be much smaller than the entire set of Gaussians in the HMMs, and the method leads to a great speedup in likelihood computation.

In DNN-HMMs, the likelihood of state/senone s_j is computed from the posterior probability of the corresponding DNN softmax output unit, which is proportional to the exponential value of the dot product $z_j = \mathbf{w}_j'\mathbf{h}^{L-1}$ or $\mathbf{w}_j'\mathbf{v}^L$, where $\mathbf{w}_j$ is the weight vector of senone s_j and $\mathbf{h}_t^{L-1} \equiv \mathbf{v}^L$ is the output vector of the last hidden

layer. If we ignore the state priors, senones with similar weight vectors $\mathbf{w}_j$ will produce similar posterior probabilities. Therefore, we propose to use k-means clustering to cluster all the senone weight vectors into a set of k weight vector clusters $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$, which are represented by their centroid vectors $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$. During decoding, dot products between the last hidden layer output $\mathbf{v}^L$ and the cluster centroids $\{\mathbf{c}_j\}$ are computed to rank the clusters in descending order. For senones in the top N nearest clusters, their exact posterior probabilities are computed using Eqn. (2). On the other hand, for the senones in an unselected cluster Ω_i, their posterior probabilities are approximated using the dot product $\mathbf{c}_i'\mathbf{v}^L$.

From a predictive perspective, we are predicting the active states in the current frame by the dot products $z_j = \mathbf{w}_j'\mathbf{v}^L$ for $j = 1, \ldots, J$, and only compute their exact values for senones that are likely to be active.

3.1. Remarks

In the actual implementation, the column vectors in the weight matrix of the softmax layer are re-arranged so that the k weight vector clusters are represented as k sub-matrices as shown in Fig. 2, to make use of better computational efficiency in matrix multiplication. After the re-arrangement, each sub-matrix is a weight vector cluster Ω_i, containing n_i members $\{\mathbf{w}_i^1, \cdots, \mathbf{w}_i^{n_i}\}$.

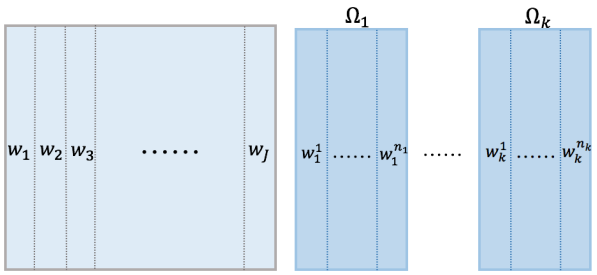


Fig. 2. Structure of the softmax layer weight matrix.

Also note that although Fig. 1 shows that the distribution of senone posteriors are very sharp, one should not simply set the posterior probability of senones from the unselected senone weight clusters to zero or a very small constant. The reason is that Viterbi decoding depends on the state likelihoods not state posterior probabilities. Since many state priors are also small, after the application of the Bayes' rule, small senone posteriors may produce state likelihoods that are 'not so small'. In this paper, we choose to approximate these small posteriors using the approximate dot products $\mathbf{c}_i'\mathbf{v}^L$ where Ω_i are the unselected weight clusters the senones belong to. The approximation does not introduce extra computation as it is already performed during the weight vector cluster selection process.

Table 1. *Recognition performance of the baseline systems. The run-times of the systems are marked 100 units for ensuing comparisons.*

System	Run-time (%)	WER (%)
with 8,704 senones	100	14.6
with 60,000 senones	100	14.0

4. EXPERIMENTS

Evaluation was performed using two different LVCSR models built on 310-hours Switchboard dataset. We first used a smaller but conventional DNN-HMM model consisting of 8,704 senones to investigate various configurations of our new speed-up method. The DTS system with 60,000 senones was then used to evaluate the final performance. Recognition results are reported on the NIST 2000 Hub5e test set.

13-dimensional MFCC features with conversation side-based cepstral mean and variance normalization were extracted. Seven consecutive feature vectors were concatenated and were then reduced to 40-dimensional feature vectors using linear discriminative analysis (LDA) [14], followed by maximum likelihood linear transform (MLLR) [15]. Finally, speaker adaptive training (SAT) [16] were applied. Both DNN systems had 7-hidden layers with 2,048 nodes per layer.

Training and decoding were performed with a dictionary with 30,000 pronunciation entries. A trigram language model was trained on all transcription texts of the training set using the SRILM toolkit [17]. Recognition performances were reported in word error rate (WER) that was computed using the NIST Scoring Toolkit. To report the recognition running times, all recognitions were performed on an Intel Xeon E5-2620 single-CPU machine running the CentOS operating system. Each experiment were repeated 5 times, and their average running time is reported. The recognition performances of the two baseline systems are shown in Table 1.

Table 2. *Recognition WER (%) of the DNN system with 8,704 senones using different number of weight vector clusters k and different number of top nearest clusters N.*

#Clusters k	#Nearest Clusters N				
	1	2	3	4	5
2	20.1	14.6	-	-	-
4	25.6	18.0	15.3	14.6	-
8	27.6	20.4	17.8	16.1	15.5
16	26.0	20.4	18.2	16.9	16
32	29.5	23.1	20.2	18.7	17.7
64	28.6	23.3	21.2	19.8	18.6
128	27.2	23.0	21.1	19.9	19.2
256	26.3	23.0	21.3	20.3	19.4
512	25.0	22.6	21.3	20.3	19.5
1024	22.6	20.7	19.7	19.0	18.5
2048	20.2	19.3	18.6	18.0	17.8
4096	17.1	16.8	16.6	16.5	16.4

4.1. Results on a system with 8,704 senones

We first investigated different configurations and their effect on the DNN system with 8,704 senones. Since we used the k-means clustering method to derive the senone weight vector clusters, there are two variables in each configuration: the total number of clusters k and the choice of the top N nearest clusters.

Table 2 shows the recognition performance of the DNN system with 8,704 senones for different choices of k and N.

Figure 3 plots the system WERs for different choices of N. The dotted line represents the baseline performance. For any given number of clusters k, the WER drops sharply in the beginning and then gradually drops as N increases. This suggests that the first few nearest clusters probably cover more active states than the later ones.

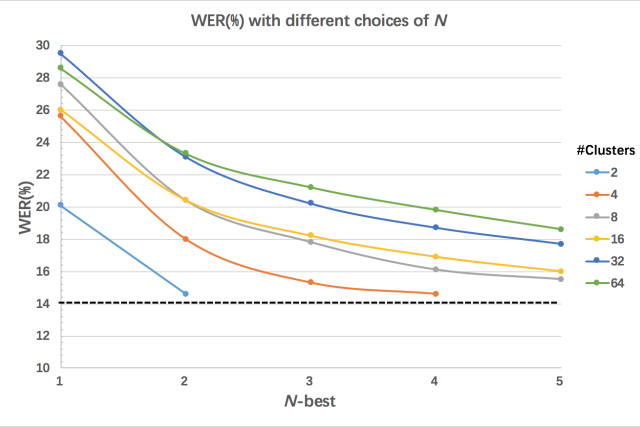


Fig. 3. Recognition WER(%) with different choices of N.

If we look at a column of Table 2, we may find that for any choice of N, WER first increases and then decreases. This reveals two factors that jointly influence the system accuracy:

- amount of exactly computed senone posteriors
- accuracy of the approximated senone posteriors

In our speed-up method, posteriors of senones in the selected clusters are exactly computed while the others are approximated. When the number of cluster k increases, the average size of each cluster decreases, so fewer senone posteriors are exactly computed. On the other hand, the approximation for senones in the unselected clusters becomes more accurate as the clusters become smaller.

Table 3 concludes the speed performance of difference k when choosing one or five nearest clusters. Computing partial senone posteriors saves much computation time, while the proposed speed-up method also introduces inefficiency in the following aspect:

- Since each frame has different active states, we cannot apply batching and the posterior vectors have to be computed frame by frame.
- The speed-up method has an additional cluster selection cost.

Considering both system accuracy and speed, we further conducted experiments with different choices of N when k was 128, 256 and 512. The best result is achieved when $k = 512$ and $N = 80$, and the speed improves by 10.1% while the WER is 14.8%.

4.2. Results on a system with 60,000 senones

According to the performance obtained from the system with 8,704 senones, we find that when the cluster number k is 512 with an average cluster size of 17, the computation speed improves while the system performance is just slightly worse than the baseline. So we repeated the speedup investigation on a system with 60,000 senones using a similar cluster size. We evaluated the performance when $k = 4,096$, which gives an average cluster size of about 15. Results are reported in Table 4.

We can see that the accuracy improves as we select more clusters. When $N = 140$, on average, about 2,100 senone posteriors are exactly computed and the remaining posteriors are approximated, and the proposed method can achieve over 35% speed up with less than 1% relative accuracy loss.

It is also observed that when more clusters are selected, the recognition accuracy will converge but still with a small loss. There

Table 3. *Relative Run-time (%) of the DNN system with 8,704-senone under different configurations.*

#Clusters k	Run-time with N Nearest Clusters (%)	
	$N = 1$	$N = 5$
32	71.5	115.4
64	66.9	89.8
128	65.7	75.7
256	66.0	72.6
512	71.1	71.2
1024	75.1	78.6
2048	89.8	93.8

Table 4. *Performance with 60,000 senones when $k = 4,096$.*

#Nearest Clusters N	Performance Metric	
	WER(%)	Run-time(%)
20	16.6	53.5
40	15.5	55.6
60	15.0	58.1
80	14.7	59.6
100	14.5	60.8
120	14.3	62.7
140	14.1	64.9
160	14.1	66.2

are several possible reasons for this problem. First, the approximation used for the posteriors of senones in the unselected clusters may not be good enough. Second, as $N = 160$ gives the same accuracy as $N = 140$, it may mean that the current N nearest clusters may not be the optimal choice. There may be better ways to select the best subsets of senones.

5. CONCLUSION

We report our work on speeding up softmax computation in DNN-based LVCSR systems. We show that the posteriors produced by DNNs can be used to predict the active states during Viterbi decoding. We propose a senone weight vector selection method that divides the weight matrix of the softmax layer into several clusters based on k-means clustering, and only computes the exact posteriors from selected clusters. On a system with 8,704 senones, we can improve the softmax computation speed by 10% with 2% relative accuracy loss. On a system with 60,000 senones, the proposed method can speed up the softmax computation by 35.1% with less than 1% relative accuracy loss. In general, the proposed method provides greater speedup for systems with more softmax outputs.

6. ACKNOWLEDGEMENTS

The work described in this paper was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. HKUST616513, HKUST16206714 and HKUST16215816), and partially by a grant from WeChat (Project No. 1516144-0).

7. REFERENCES

[1] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[2] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proceedings of Interspeech*, 2011, pp. 437–440.

[3] George Saon, Jeff Kuo, Steven Rennie, and Michael Picheny, "The IBM 2015 English conversational telephone speech recognition system," in *Proceedings of Interspeech*, 2015, pp. 3140–3143.

[4] Dongpeng Chen and Brian Mak, "Distinct triphone acoustic modeling using deep neural networks," in *Proceedings of Interspeech*, 2015, pp. 2645–2649.

[5] Hank Liao, Erik McDermott, and Andrew Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 2013, pp. 368–373.

[6] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao, "Improving the speed of neural networks on CPUs," in *Proceedings of the Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011.

[7] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition.," in *Proceedings of Interspeech*, 2013, pp. 2365–2369.

[8] "Intel C++ Intrinsics Reference," `https://software.intel.com/sites/default/files/a6/22/18072-347603.pdf`.

[9] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2011, pp. 5528–5531.

[10] Andriy Mnih and Yee Whye Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the International Conference on Machine Learning*, 2012.

[11] Xie Chen, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2015, pp. 5411–5415.

[12] Enrico Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1993, vol. 2, pp. 692–695.

[13] Kate M Knill, Mark J. F. Gales, and Steve J Young, "Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs," in *Proceedings of the International Conference on Spoken Language Processing*, 1996, vol. 1, pp. 470–473.

[14] Reinhold Haeb-Umbach and Hermann Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, vol. 1, pp. 13–16.

[15] Mark J. F. Gales, "Semi-tied covariance matrices," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, vol. 2, pp. 657–660.

[16] Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul, "A compact model for speaker-adaptive training," in *Proceedings of the International Conference on Spoken Language Processing*, 1996, vol. 2, pp. 1137–1140.

[17] Andreas Stolcke et al., "SRILM—an extensible language modeling toolkit," in *Proceedings of Interspeech*, 2002.