EXEMPLAR-BASED LARGE VOCABULARY SPEECH RECOGNITION USING K-NEAREST NEIGHBORS

Yanbo Xu[*1], *Olivier Siohan*[2], *David Simcha*[2], *Sanjiv Kumar*[2], *Hank Liao*[2]

[1]Department of Electrical and Computer Engineering, University of Maryland College Park, MD, USA
[2]Google Inc., New York

ABSTRACT

This paper describes a large scale exemplar-based acoustic modeling approach for large vocabulary continuous speech recognition. We construct an index of labeled training frames using high-level features extracted from the bottleneck layer of a deep neural network as indexing features. At recognition time, each test frame is turned into a query and a set of k-nearest neighbor frames is retrieved from the index. This set is further filtered using majority voting and the remaining frames are used to derive an estimate of the context-dependent state posteriors of the query, which can then be used for recognition. Using an approximate nearest neighbor search approach based on asymmetric hashing, we are able to construct an index on over 25,000 hours of training data. We present both frame classification and recognition experiments on a Voice Search task.

Index Terms— acoustic modeling, exemplar-based recognition, k-Nearest Neighbor, deep neural network

1. INTRODUCTION

Over the past several decades, the most successful approaches to acoustic modeling for speech recognition have relied on data-driven techniques to estimate models such as hidden Markov models (HMM) or deep neural networks (DNNs) [1]. Improvements in recognition accuracies were for the most part driven by the use of labeled training sets of larger and larger sizes and model estimation criteria directly related to the recognition performance, such as MMI [2].

Over that period of time, the amount of speech data available to the research community has grown by 2 orders of magnitude, from tens to several thousand hours of manually transcribed data (e.g. from Resource Management to Switchboard [3]). While it had a tremendous impact on recognition accuracy, this remains a moderate increase in training resources compared to other fields such as Natural Language Processing where text corpora for language modeling grew by about 6 orders of magnitude over the same period (from 1M words in the Brown corpus [4] to 2T words used in [5]). Similarly, computing power increased by about 6 orders of magnitude over the same period, enabling compute-intensive approaches to estimate model parameters [6].

The slow increase in the amount of acoustic training data can be largely attributed to the cost of acquiring and manually transcribing large amounts of audio. In the past couple of years however, thanks to the deployment of voice services such as Google Voice Search [7] or Siri [8], it has become possible for several groups in the industry to collect large amounts of in-domain audio data, on the order of tens of thousands of hours or more, to train acoustic models using semi-supervised techniques without any need for additional manual transcriptions [9, 10].

Unfortunately, while an unprecedented amount of acoustic training data is becoming available, the computing requirements to estimate state-of-the-art models such as DNNs significantly limit the use of much larger training sets. As an illustration, training a product-grade acoustic model on a few thousand hours of audio can take over a week of computation time even when using a distributed training approach based on asynchronous stochastic gradient descent [11] running on a computer cluster. Constructing models on significantly larger training sets can lead to training times that are no longer manageable.

In this paper, we propose to exploit large and automatically transcribed training sets using an exemplar-based approach to acoustic modeling [12], by-passing the time-consuming estimation of DNN model parameters. Our approach builds on the use of high-level features extracted from the bottleneck layer of a pre-trained DNN to provide dense feature vectors of reduced dimension [13] suitable for indexing, and on some recent progress on approximate k-nearest neighbor (k-NN) search [14, 15] enabling the indexing of billions of records. The indexing of such an amount of data is enabled by a computer cluster distributing the storage of the index over a pool of servers. Unlike some of the previous works on exemplar-based acoustic modeling [16, 17, 18], our experiments are conducted with a production-grade large vocabulary speech recognition system using training sets of up to 25,000 hours of audio.

In Section 2, we describe our k-NN approach for acoustic modeling, followed in Section 3 by frame classification and recognition experiments on a Voice Search task. Section 4 concludes the paper.

2. KNN-BASED ACOUSTIC MODELING

2.1. Overview

In a hybrid DNN-HMM system, a DNN model is used to provide an estimate of $P(\mathbf{s}|\mathbf{x})$, the posterior probabilities of a set of context-dependent (CD) state inventory $\mathbf{s}$ given an acoustic feature vector $\mathbf{x}$. This posterior distribution is then scaled using $P(\mathbf{s})$, the prior distribution of each CD state symbol estimated from the training set, turning the posterior into a likelihood $p(\mathbf{x}|\mathbf{s})$ which can then be used for recognition.

Following the procedure described in [10] which consists in automatically selecting high-confidence utterances extracted from the anonymized audio logs of Google Voice Search, we are able to construct data sets of arbitrary size. Such a semi-supervised procedure can be used to train acoustic models without relying on any additional manual transcription. However, when large training sets are used (e.g. 30M utterances, or about 25,000 hours of audio) the time it takes to train a DNN with a large CD state inventory (typically around 15k CD states) can become prohibitive as it is essentially a sequential procedure based on stochastic gradient descent (SGD). While approaches such as asynchronous SGD [11] can parallelize the training, processing large data sets still remains a time consuming procedure.

In this paper, we propose instead to use an existing DNN trained on 2,500 hours of manually transcribed utterances to compute $P(\mathbf{s}|\mathbf{x})$ on all frames of all utterances of a much larger unsupervised training

*The author performed the work as a summer intern at Google, NYC

set (up to 25,000 hours automatically obtained from the procedure described above) and store the resulting frame-level posteriors into an index, which can be constructed efficiently, even on large training sets. This index is then used to construct an estimate of $P(\mathbf{s}|\mathbf{x})$ for any test frame $\mathbf{x}$, based on the pre-computed $P(\mathbf{s}|\mathbf{x})$ of its k-Nearest Neighbor (k-NN) [19] frames extracted from the index. Since each training frame stored in the index is also labeled with its CD-state (obtained by force-aligning the training set), it is possible to apply a majority voting rule and restrict the set of k-NN frames associated to a given query to the subset of frames winning the majority vote.

Two important and related aspects when indexing the training frames are the choice of the indexing feature, or query, and of the indexing approach. The indexing feature vector should carry discriminative information to enable distinguishing between the different CD-state labels associated to each frame, without reaching an unduly large dimension that would prevent the computation of a meaningful distance between frames. For those reasons, we propose to use the activations of the bottleneck layer of a DNN as indexing features, similar to what was done in [13]. In addition, given the size of the index, a brute force search over the entire index is not feasible and we will rely instead on an approximate search method based on asymmetric hashing [15], as described in Section 2.2.

The entire approach is divided into 2 stages. In the first stage, the equivalent of training, an index of frames is constructed by first computing for each frame in the training set its indexing feature vector (using a bottleneck DNN), its CD-states posteriors (using a pre-existing DNN, not necessarily the same bottleneck DNN that was used to compute the indexing features), and its corresponding CD-state label obtained by running forced alignment against the reference transcript. Once this information is available, the index is built according to the procedure described in Section 2.2. In the second stage, corresponding to the recognition step, each frame of every test utterance is turned into a query by computing its bottleneck layer activations, and the index returns a corresponding list of nearest neighbor frames from the training set. The precomputed posteriors of the nearest neighbor frames are used to construct an estimate of the state posterior of the query frame, which can be used instead of the posteriors provided by the baseline DNN. The advantage of this approach is that the index can be constructed from a much larger training set than what was used to train the DNN. Provided that the index is able to preserve the locality of frames sharing the same CD-state label, the use of nearest neighbors could improve the estimation of the state posteriors.

2.2. Approximate Nearest Neighbor Search

We rely on Google's Scalable Matching toolkit $ScaM$ [15] to retrieve the approximate nearest neighbors of each query frame. Specifically, we use *Asymmetric Hashing* [14] to compute the approximate distance between feature vectors.

Given an indexing set of M bottleneck layer feature vectors $\{\mathbf{v}_j\}_{j=1}^{M}$, with $\mathbf{v}_j \in R^N$, the consecutive elements of each vector $\mathbf{v}$ are divided into N/d d-dimensional subspaces or chunks $\mathbf{c}_i(\mathbf{v})$. Within each subspace i, k-means clustering is applied to the set of feature vectors $\{\mathbf{c}_i(\mathbf{v}_j)\}_{j=1}^{M}$ to construct K centroids $\{\mathbf{C}_i^k\}_{k=1}^{K}$, resulting in a total of $K \times N/d$ centroids. To index each feature vector $\mathbf{v}$, we find the closest centroid in each chunk and assign the integer index of its closest centroid as the quantization label for $\mathbf{c}_i(\mathbf{v})$ [20]. Each $\mathbf{v}$ is effectively hashed into a N/d-dimensional vector composed of integers from 1 to K. To find the approximate nearest neighbors of a query $\mathbf{u}$, we compute the approximate distance between $\mathbf{u}$ and each of the M vectors in the indexing set. The "asymmetric" term in asymmetric hashing refers to the fact that only vectors in the indexing set are hashed, not the query [14]. The approximate distance between $\mathbf{u}$ and $\mathbf{v}$ is the summation of the distances between

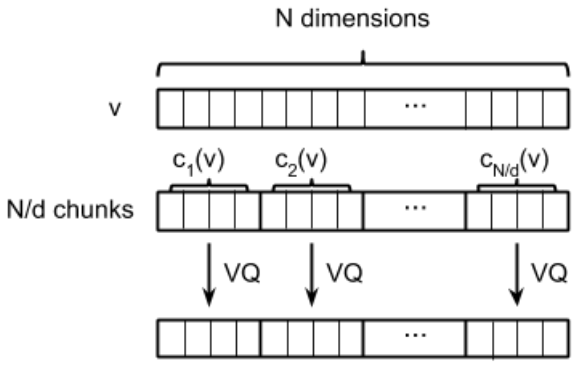


Fig. 1. Illustration of how feature vectors of dimension N are grouped into N/d chunks of dimension d and independently quantized in each chunk.

$\mathbf{c}_i(\mathbf{u})$ and $\mathbf{C}_i^{I_i(\mathbf{v})}$ where $I_i(\mathbf{v})$ is the index of the closest centroid to $\mathbf{c}_i(\mathbf{v})$. Since for each chunk of $\mathbf{u}$, there are only K unique pairs to consider for distance computation, a lookup table of size $K \times N/d$ is first constructed. For each $\mathbf{v}$, it then takes $N/d - 1$ additions to sum the corresponding N/d table lookups. Once the approximate distances to all vectors are computed, it is possible to refine the distance computation using exact reordering, which consists in computing the exact distance between the query and its top R neighbors to partially recover the loss of accuracy due to hashing.

Because of the large number of feature vectors in the indexing set, the computational complexity of asymmetric hashing is dominated by the approximate distance combination for each vector. In terms of cost in Floating Point Operations for each query vector, the cost is $O(M(N/d-1))$ with asymmetric hashing when ignoring the overhead of lookup table construction and exact reordering, while for ground truth brute force search, the cost is $O(M(3N-1))$. The speed up factor brought by asymmetric hashing is proportional to the chunk size d. Experimentally, we observe a 30x speed up with $d = 32$, and 15x speed up with $d = 16$.

2.3. Frame Classification and State Posterior Estimation

Once the index has been constructed, it can be used for both frame classification and to estimate the state-posteriors of each query frame from the test set, which can then be used instead of the DNN-provided posteriors for recognition. For each query, ScaM returns the top k approximate nearest neighbors frames from the training set. Recall that each frame in the index is associated with a tuple providing its pre-computed state posteriors as well as its corresponding ground-truth CD-state label.

For frame classification, we propose to use majority voting to predict the label of query: the most frequent label amongst the nearest neighbors is returned as the hypothesized label.

To estimate the state-posteriors of a query frame, we compute an average of the pre-computed state posteriors of each frame from its k-NN list. This contrasts with the approach in [21] where the class posteriors were computed directly from the class counts in the k-NN list, which is not applicable when dealing with a large number of classes (15k in our case) as it would require generating large k-NN lists. In one approach, called ***MAJOR***, we restricted the list of nearest neighbor frames used for the averaging to the frames belonging to the majority vote class. We contrasted this with not using the frame labels and simply averaging the posteriors from the k nearest neighbor, an approach denoted as ***NEAR***.

3. EXPERIMENTS AND RESULTS

3.1. Datasets

Our typical training set consists of about 3 million manually transcribed Voice Search queries (about 2,500 hours), denoted as $3M\_supervised$. In this work we focus on constructing a system on a much larger training set consisting of 30 million utterances whose transcriptions are automatically generated by our production ASR system, denoted as $30M\_unsupervised$. To investigate the impact of the size of the training set on performance, the $30M\_unsupervised$ set is randomly downsampled to construct a 3M utterances subset ($3M\_unsupervised$) and a 300K utterances subset ($0.3M\_unsupervised$). To mitigate the CD-state labels imbalance that is typical of a speech data set (silence frames are the dominant classes), we also downsampled the $30M\_unsupervised$ set to construct a data set having as many frames as in a 3M utterances set, but downsampled to provide a near-uniform distribution of the CD-state labels. This set is denoted $flattened\_unsupervised$ and matches $3M\_unsupervised$ in size. In the following, these data sets are also called indexing sets. Last, we use $23K$ Voice Search queries as a test set.

3.2. DNN Bottleneck Layer Features

The DNN that we use to compute both the indexing features and the state posteriors of all training frames has 8 hidden rectified linear units layers each with 2560 units, an input layer of 1040 units corresponding to 26 stacked frames of 40-dimensional mel-spectrum log-filterbank energies with 20 frames to the left and 5 frames to the right of the center frame, and the output softmax layer has an inventory of 14247 CD states. The connections between the last hidden layer and the output layers are factored into a low rank representation [22] forming a bottleneck layer of 256 units with linear activations. A similar architecture could be found in [23]. The activations of the bottleneck layer are a non-linear and dimensionality reducing transformation of the original input features. This DNN was trained on the $3M\_supervised$ training set and defines our baseline system.

For each frame $\mathbf{x}$, a forward pass is applied to obtain the bottleneck layer activations $\mathbf{v}(\mathbf{x})$ and state posteriors $P(\mathbf{s}|\mathbf{x})$ from the output layer. Note that we could have used two distinct DNNs, one to compute the low-dimension activations needed for indexing and another to compute the state posteriors needed for recognition. In particular, since the state posteriors are only computed on the training set, it is possible to use an expensive procedure (e.g. large model or ensemble of models) to derive those posteriors. In this work, for simplicity, the same bottleneck DNN is used to provide both the activation features for indexing and the CD-state posteriors. The ground truth label $l(\mathbf{x})$ corresponding to each frame is obtained by forced alignment. Thus the information associated to each frame is a tuple $< \mathbf{x}, \mathbf{v}(\mathbf{x}), P(\mathbf{s}|\mathbf{x}), l(\mathbf{x}) >$. We use the bottleneck layer feature vectors $v(\mathbf{x})$ for the nearest neighbors search and the squared $L2$ distance to measure the distance between those feature vectors.

3.3. ScaM Setup

The unsupervised indexing set $30M\_unsupervised$ contains around 7 billion frames. For indexing purposes, the number of centroids used in each chunk is set to $K = 256$. All our experiments are carried out in an offline batch mode, parallelizing the search computation using MapReduce [24]. A total of $5,000$ mapper jobs are allocated to handle the approximate nearest neighbor search, each independently handing an indexing set of around 1.6 million frames. To obtain the top 100 neighbors for each query, each mapper should return the top 100 neighbors, which are then sent to the reducer for combination and exact reordering. However, to process a query set as large as $23K$ utterances, the whole MapReduce pipeline can generate intermediate data exceeding $200TB$. To mitigate this issue, an additional approximation is made to restrict the list of nearest-neighbors from each mapper to 5 neighbors. Since the frames are pseudo-randomly distributed across mappers, the probability of having a single mapper out of $5,000$ hold over 5 ground truth frames when querying for the 100-nearest neighbors is extremely low and was shown not to impact performance while significantly reducing intermediate data storage requirements.

3.4. Approximate Nearest Neighbor Search Accuracy

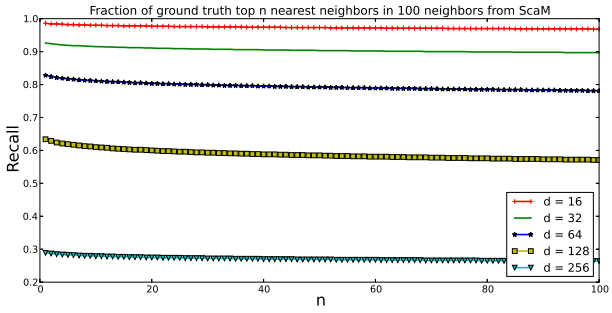


Fig. 2. Effect of chunk size d on approximate nearest neighbor search accuracy

The quality of the approximate nearest neighbor search is crucial to our work. Since the chunk size d largely determines the speed up of the approximate search compared to the ground truth brute force search at the expense of accuracy, we are also interested in how the value of d affects performance. We use 'recall at 100' which represents the fraction of the ground-truth top n-nearest neighbors which are present in the returned 100 neighbors, as a quality measure. For those experiments, the $3M\_supervised$ is used as the indexing set since its labels are considered to be more accurate than those of the unsupervised training sets and the difference we observe can be more attributed to the value of d. 5% of test set is used in the evaluation during which both ground truth top-100 nearest neighbors and approximate ones are found for each query. From Fig.2, we can see that good recall can be obtained by setting $d = 16$, and the accuracy drops quickly as d increases. It is also interesting to see that the line is almost flat with increasing n. This phenomenon is due to quantization and exact reordering: most of the time the ground truth top-100 nearest neighbors are either returned by ScaM together or totally missing.

3.5. Frame Classification Error Rate

We also investigate how the intrisic quality of the approximate nearest neighbor search procedure, as measured by different recall levels controlled by the chunk size, impacts the frame classification performance. Using the same indexing and test sets as above, each test frame is classified based on majority voting within the k-NN list returned by the index. Results are given in Fig.3 for the same indexing configuration as in the previous section, with the chunk size d varying from 16 (high recall configuration) to 256 (low recall). All curves go down quickly and converge around 50 neighbors. It is worth noting that the plots associated with d ranging from 16 to 64 coincide with each other. Even though recall at 100 with $d = 16$ is around 97% while only 80% with $d = 64$, there is no significant difference in terms of frame classification error rate. This indicates that neighbors nearest to the query have no advantage over the ones that are not

within the top-100 nearest neighbors in providing useful information for label prediction.

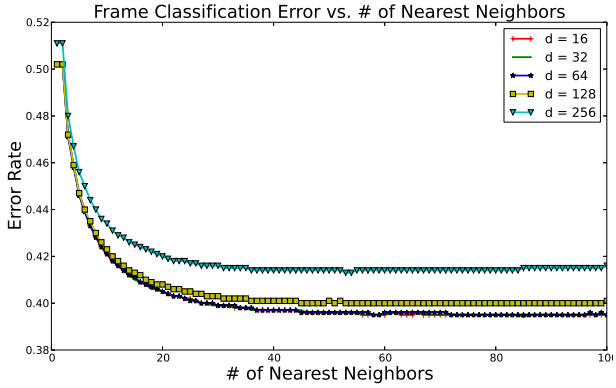
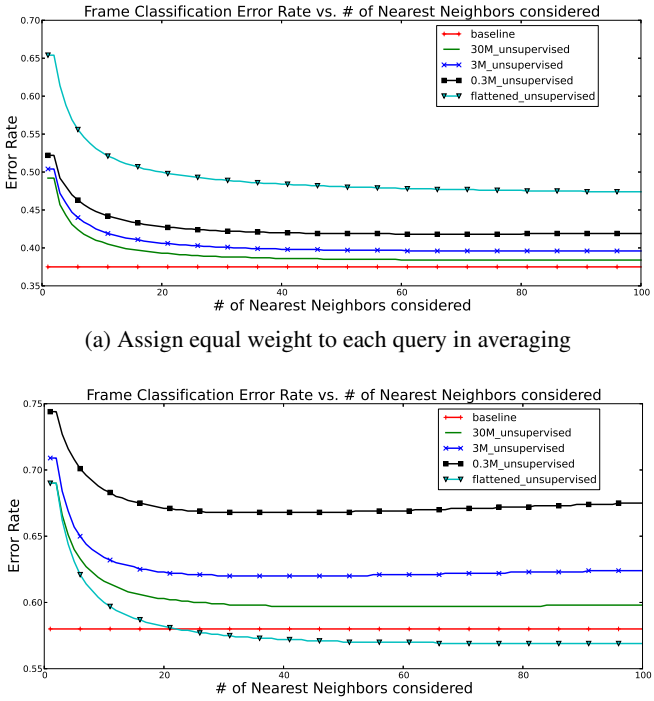


Fig. 3. Effect of chunk size d on frame classification error rate

We then perform frame classification experiments using the unsupervised indexing sets and the entire test set. The chunk size was set to $d = 32$ for ScaM. A characteristic of speech data sets is that the different CD state labels are very unbalanced. In Fig.4(a), each frame in the test set is assigned equal weight when computing the average frame classification error rate, which is consistent with the goal of frame-wise cross entropy training for DNN. However, the averaged result is inevitably dominated by the frequent CD states. In Fig.4(b), we assign equal weight to each CD state in averaging which provides a more informative result on the classification performance for most of the CD states. In Fig.4(a), all curves quickly decrease as the number of nearest neighbors considered increases, and then converge. With larger indexing sets, the result of k-NN gradually approaches the baseline DNN performance. We observe a similar trend in Fig.4(b). What's different however is that curves slightly bounce back for large number of neighbors which indicates that frame samples are not dense enough for less frequent CD states, especially for smaller indexing sets. In Fig.4, different from other unsupervised indexing set, $flattened\_unsupervised$ achieves the best performance when equal weight is assigned for each CD state at the expense of much worse performance for frame error rate sample-wise averaged. Since sample-wise averaging is more related with recognition performance than state-wise averaging, this flattened indexing set also failed to generate competitive result in our ASR experiments.

3.6. Automatic Speech Recognition Experiments

We finally investigate the performance of our framework on ASR experiments. The baseline DNN system trained on $3M\_supervised$ operates at 11.2% WER. Our proposed approach using an index constructed on $30M\_unsupervised$ reaches 11.7% WER at best, 5% relative worse than the DNN baseline. The use of majority voting (***MAJOR***) to limit the set of training frames used to estimate the CD-state posteriors does not outperform not using any class-labels (***NEAR***). Similar results were obtained when restricting the majority voting to context-independent state labels or phone labels. We also experimented with using the distance between the query and each of its nearest neighbors to derive a weighted average of the CD-state posteriors but did not observe any gain in performance. From Table.1, we can also observe the trend that larger indexing sets lead to better performance. Focusing on ***NEAR***, we see that 3 to 5 neighbors seem to be a good choice for the state posteriors estimation in both accuracy and efficiency. While our best system did not outperform a well tuned DNN system trained on manually labeled data, it is still promising to see that with a large unsupervised data set, our framework can generate competitive result from a totally different perspective.



(a) Assign equal weight to each query in averaging

(b) Assign equal weight to each CD state in averaging

Fig. 4. Frame classification error rate with various sizes of indexing sets

Indexing set	number of neighbors			
	1	3	5	10
$30M\_unsupervised$ ***MAJOR***	12.1	11.9	11.9	11.9
$30M\_unsupervised$ ***NEAR***	11.9	11.7	11.7	11.7
$3M\_unsupervised$ ***NEAR***	12.2	12.1	12.1	12.1
$0.3M\_unsupervised$ ***NEAR***	12.7	12.5	12.5	12.6
Hybrid DNN baseline	11.2			

Table 1. WER(%) of ASR experiments with various sizes of indexing sets and numbers of neighbors for *SP*s averaging

4. CONCLUSION

In this paper we have explored the use of k-nearest neighbors over a large index of frames to compute for each test frame an estimate of its CD-state posteriors. We have shown that the activations of the bottleneck layers of a DNN can be used to construct discriminative features for indexing purposes. In addition, we have illustrated that an index of over 7B frames or 25,000 hours of speech can be constructed and searched using an approximate nearest neighbor search approach based on asymmetric hashing [14]. As it becomes possible to collect very large speech databases, we are hoping that advances in approximate nearest neighbor search can support the indexing of longer acoustic units and provide alternative approaches to acoustic modeling.

5. REFERENCES

[1] Xuedong Huang, James Baker, and Raj Reddy, "A historical perspective of speech recognition," *Commun. ACM*, vol. 57, no. 1, pp. 94–103, Jan. 2014.

[2] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *Proc. ICASSP*, 1986, pp. 49–52.

[3] Linguistic Data Consortium, "LDC speech corpura," https://catalog.ldc.upenn.edu.

[4] Brown University, "Brown university standard corpus of present-day american english," http://http://en.wikipedia.org/wiki/Brown_Corpus.

[5] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, Jeffrey Dean, and Google Inc, "Large language models in machine translation," in *In EMNLP*, 2007, pp. 858–867.

[6] Wikipedia, "Moore's law," https://http://en.wikipedia.org/wiki/Moore

[7] Johan Schalkwyk, Doug Beeferman, Francoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope, "Your word is my command: Google search by voice: A case study," in *Advances in Speech Recognition*, Amy Neustein, Ed., pp. 61–90. Springer US, 2010.

[8] Apple, "Siri," http://www.apple.com/ios/siri/.

[9] Yan Huang, Dong Yu, Yifan Gong, and Chaojun Liu, "Semi-supervised GMM and DNN acoustic model training with multi-system combination and confidence re-calibration," in *Proceedings of the 14th Annual Conference of the International Speech Communication Association (Interspeech)*, Lyon, France, 2013.

[10] Olga Kapralova, John Alex, Eugene Weinstein, Pedro Moreno, and Olivier Siohan, "A big data approach to acoustic model training corpus selection," in *INTERSPEECH*, 2014.

[11] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng, "Large scale distributed deep networks," in *NIPS*, 2012.

[12] Tara N. Sainath, Bhuvana Ramabhadran, David Nahamoo, Dimitri Kanevsky, Dirk Van Compernolle, Kris Demuynck, Jort F Gemmeke, Jerome R Bellegarda, and Shiva Sundaram, "Exemplar-based processing for speech recognition: An overview," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 98–113, 2012.

[13] Li Deng and Jianshu Chen, "Sequence classification using the high-level features extracted from deep neural networks," in *Proc. ICASSP*, May 2014.

[14] Hervé Jégou, Matthijs Douze, and Cordelia Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 33, no. 1, pp. 117–128, jan 2011.

[15] Dan Vanderkam, Rob Schonberger, Henry Rowley, and Sanjiv Kumar, "Nearest neighbor search in Google correlate," Tech. Rep., Google, 2013.

[16] Thomas Deselaers, Georg Heigold, and Hermann Ney, "Speech Recognition with State-based Nearest Neighbour Classifiers," in *Interspeech*, Antwerp, Belgium, Aug. 2007, pp. 2093–2096.

[17] Afsaneh Asaei, Hervé Bourlard, and Benjamin Picart, "Investigation of kNN classifier on posterior features towards application in automatic speech recognition," Idiap-RR Idiap-RR-11-2010, Idiap, 6 2010.

[18] Mathias De Wachter, Kris Demuynck, Dirk Van Compernolle, and Patrick Wambacq, "Data driven example based continuous speech recognition," in *INTERSPEECH*, 2003.

[19] Thomas Cover and Peter Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[20] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[21] Natasha Singh-Miller and Michael Collins, "Learning label embeddings for nearest-neighbor multi-class classification with an application to speech recognition," in *Advances in Neural Information Processing Systems*, 2009, pp. 1678–1686.

[22] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6655–6659.

[23] Hank Liao, Erik McDermott, and Andrew Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 368–373.

[24] Jeffrey Dean and Sanjay Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.