Application of Deep Belief Networks for Natural Language Understanding

Ruhi Sarikaya, Geoffrey E. Hinton, and Anoop Deoras

***Abstract*—Applications of Deep Belief Nets (DBN) to various problems have been the subject of a number of recent studies ranging from image classification and speech recognition to audio classification. In this study we apply DBNs to a natural language understanding problem. The recent surge of activity in this area was largely spurred by the development of a greedy layer–wise pretraining method that uses an efficient learning algorithm called Contrastive Divergence (CD). CD allows DBNs to learn a multi-layer generative model from unlabeled data and the features discovered by this model are then used to initialize a feed-forward neural network which is fine-tuned with backpropagation. We compare a DBN-initialized neural network to three widely used text classification algorithms: Support Vector Machines (SVM), boosting and Maximum Entropy (MaxEnt). The plain DBN-based model gives a call–routing classification accuracy that is equal to the best of the other models. However, using additional unlabeled data for DBN pre–training and combining DBN–based learned features with the original features provides significant gains over SVMs, which, in turn, performed better than both MaxEnt and Boosting.**

***Index Terms*—Call-Routing, DBN, Deep Learning, Deep Neural Nets, Natural language Understanding, RBM.**

I. Introduction

THE goal of spoken language understanding (SLU) systems is to enable communication between a human and machine. SLU systems automatically identify a user's intent from natural language by extracting the information bearing words and issuing queries to back-end databases to satisfy the user's requests. Ground-breaking advances in speech recognition technology from early 1980's to early 1990's opened the way for spoken language understanding. An early SLU task was the DARPA (Defense Advanced Research Program Agency) Airline Travel Information System (ATIS) project [1] in 1990. This project focused on building spoken understanding systems in the travel domain. These systems handled spoken queries related to flight-related information including flight booking and hotel reservation. An example utterance from this domain is *I want to fly from Seattle to Miami tomorrow morning*. Language understanding was reduced to the problem of extracting task-specific slots, such as *DestinationLocation*, *DepartureLocation* and *DepartureDate*, where the intent is *FindFlight*.

The conditional random fields (CRFs) [4] is one of the most widely used discriminative modeling technique for slot filling [2], [3] in spoken language understanding. Slot filling is cast as a sequence classification problem to obtain the most probable slot sequence:

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmax}}\ P(\mathbf{C}|\mathbf{W})$$

where $\mathbf{W} = w_1, \ldots, w_T$ is the input word sequence and $\mathbf{C} = c_1, \ldots, c_T, c_t \in \mathcal{C}$ is the sequence of associated class labels $\in \mathcal{C}$.

Motivated by the success of early commercial interactive voice response (IVR) applications used in call centers, a new SLU task evolved: that of determining the user intent. This new SLU task was framed as *classifying users' utterances into predefined categories* (called *intents* or *call-types*) [5]. For example, if the user said something related to a billing statement in an IVR setting, the automatic call routing system should direct the call to the billing department. For intent determination (for call routing or other tasks), early work on discriminative classification algorithms for the AT&T HMIHY system [5] used Boosting [6]. In this paper, we focus on the intent determination task, specifically focusing on call routing applications. We frame the problem in a probabilistic setting. More formally, given the sequence of words, $\mathbf{W}$, the most likely user intent (class label), U^* is given by:

$$U^* = \underset{U}{\operatorname{argmax}}\, P(U|\mathbf{W})$$

where $\mathbf{W} = w_1, \ldots, w_T$ is the input word sequence and $U \in \mathcal{U}$ is the user intent among the possible set of intents $\in \mathcal{U}$. We refer interested readers to [9] for a detailed history and overview on SLU.

Today, natural language call routing is one of the most widely adopted NLP technologies in the world, and there are hardly any large companies that do not use it for dealing with customers. The main advantage of call routing is the automation it provides for customer care, largely eliminating customer/agent interaction. As such, every small improvement in call routing accuracy matters since users whose goal is not identified by the system require a human agent to resolve their problems. A typical call routing system is composed of two statistical components: a speech recognition system and an action classifier. The speech recognition system transcribes the speaker's speech and sends the transcription to the action classifier, which extracts the speaker's intent embodied in different call-types. Each call-type

Manuscript received November 08, 2012; revised September 02, 2013; accepted January 11, 2014. Date of publication February 11, 2014; date of current version February 19, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Pascal Fung.

R. Sarikaya and A.Deoras are with Microsoft Corporation, Redmond, WA 98052 USA (e-mail: ruhi.sarikaya@microsoft.com; anoop.deoras@microsoft.com).

G. Hinton is with the Department of Computer Science, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: hinton@cs.toronto.edu).

Digital Object Identifier 10.1109/TASLP.2014.2303296

2329-9290 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

triggers a different action in the system back-end. There are numerous machine learning techniques such as Boosting [6], Maximum Entropy Modeling (MaxEnt) [21], [20] and Support Vector Machines (SVM) [7], [8], which are used as action classifiers. All of these techniques require labeled data to train a model. Quantity and quality of labeled data are the determining factors in building and deploying such systems. The complexity of the call routing task largely determines how much labeled data is needed to achieve a reasonable performance level. As the complexity of the task increases the amount of training data required for a reasonable performance level can become large. Therefore, there are several key areas for technology improvement: 1) minimizing the amount of labeled data to achieve a given performance level, 2) improving the machine learning algorithms to achieve the best performance for a given amount of labeled data, and 3) exploiting unlabeled data, which are typically available in much larger quantities than labeled data, to improve the performance for a given amount of labeled data.

Neural Networks (NNets) are not new to the speech and language processing field. There have been numerous applications of NNets to speech recognition and natural language processing problems during the past two decades. Even though NNets, particularly deep nets with many hidden layers, appeared capable of modeling complex structures and dependencies in the data, they failed to live up to the expectations because of the lack of effective training algorithms for training such networks. Consequently, until very recently, NNets lost the battle against GMMs/HMMs for speech recognition due to larger computational demands and difficulty in parallelizing the model training compared to the GMM/HMM approach. In the NLP area, where the primary problems can be cast as classification problems, NNets fared better, but they still were not the preferred modeling approach compared to maximum entropy models, support vector machines, and boosting techniques partly due to the difficulty in training deep networks. Moreover, SVM and boosting have maximum margin properties with faster training algorithms. Recently, however, there has been increasing interest in Deep Belief Networks (DBNs) because of the invention of an efficient layer-by-layer learning technique. The building block of a DBN is a probabilistic model called a Restricted Boltzmann Machine (RBM), which is used to discover one layer of features at a time. To learn a DBN, RBMs are applied recursively with the feature activations produced by one RBM acting as the data for training the next RBM in the stack. DBNs have been used as generative models of many different forms of data in such diverse areas as image classification, speech recognition and information retrieval [10], [11], [12]. Deep networks typically have higher modeling capacity than shallow networks with the same number of parameters, but they are harder to train, both as stochastic top-down generative models and as deterministic bottom-up discriminative models. For generative training, it is generally very difficult to infer the posterior distribution over the multiple layers of latent (hidden) variables. For discriminative training using backpropagation, learning can be very slow with multiple hidden layers and overfitting can also be a serious problem. The recursive training method for DBNs solves the inference problem. The use of features found by the DBN to initialize a multilayer feed-forward neural network significantly decreases both the time taken for discriminative training and the amount of overfitting [13].

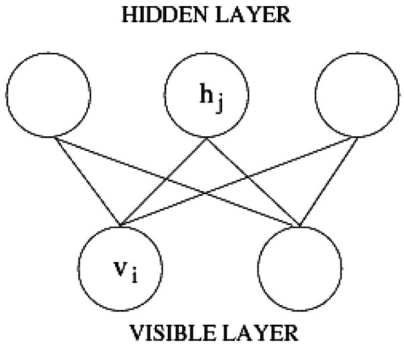

Fig. 1. RBM Architecture.

RBMs can be trained using unlabeled data and they can learn stochastic binary features which are good for modeling the higher-order statistical structure of a dataset. Even though these features are discovered without considering the discriminative task for which they will be used, some of them are typically very useful for classification as well as for generation. A subsequent stage of discriminative fine-tuning can then slightly change the feature weights to make the network even more useful for discrimination with much less overfitting, which otherwise can be a serious problem with purely discriminative training. This is particularly helpful when the number of labeled training examples is relatively small. In this regime, it has been shown that classifiers based on generative models can outperform discriminative classifiers, even without making use of additional unlabeled data [14].

Part of the work in this paper is presented in [15]. In this paper we pursue two lines of research suggested as future work in [15]: a) investigating the effect of using unlabeled data to train RBMs, and b) treating the DBN as a feature generator and using a separate classifier such as an SVM to perform the actual classification task. These techniques lead to clear performance improvements both over the baseline DBN and SVM, which are largely equivalent in terms of the performance figures.

The rest of the manuscript is organized as follows: Section 2 provides a brief introduction to RBMs. Section 3 describes how to train a stack of RBMs recursively and how to use the resulting DBN to initialize a feed-forward neural network that can be discriminatively fine-tuned to optimize classification. Section 4 summarizes the other widely used discriminative classifiers. Section 5 presents the experimental results and discussion followed by the conclusions in Section 6.

II. Restricted Boltzmann Machines

A restricted Boltzmann machine [16] is a two-layer, undirected, bipartite graphical model where the first layer consists of observed data variables (or visible units), and the second layer consists of latent variables (or hidden units). The visible and hidden layers are fully connected via symmetric undirected weights, and there are no intra-layer connections within either the visible or the hidden layer. A typical RBM model topology is shown in Fig. 1.

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

The weights and biases of an RBM determine the energy of a joint configuration of the hidden and visible units $E(v, h)$,

$$E(v, h; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_i \quad (1)$$

with model parameters $\theta = \{W, b, a\}$ and $v_i, h_j \in \{0, 1\}$. W are the symmetric weight parameters with $V \times H$ dimensions, b are the visible unit bias parameters, a are the hidden unit bias parameters. The network assigns a probability to every possible visible-hidden vector pair via the energy function,

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \quad (2)$$

The normalization term or partition function, Z, is obtained by summing over all possible pairs of visible and hidden vectors.

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (3)$$

The probability that the model assigns to a visible vector, $\mathbf{v}$, is obtained by marginalizing over the space of hidden vectors,

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (4)$$

The simplest RBMs use Bernouilli-distributed units (*i. e.* stochastic binary units), but they can be generalized to any distribution in the exponential family [12]. However, some combinations of distributions for the visible and hidden units are very hard to train (see [17] for more details). In this paper, we restrict ourselves to binary units for all of the experiments.

The derivative of the log probability of a visible vector, $\mathbf{v}$ with respect to the weights is given by:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{\mathbf{v}} - \langle v_i h_j \rangle_{model} \quad (5)$$

where the angle bracket denotes the expectation with respect to the distribution specified in the subscript. Following the gradient of the log likelihood we obtain the update rule for the weights as,

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \quad (6)$$

where ϵ is the learning rate. The lack of hidden–hidden connections makes the first expectation easy to compute. Given a visible vector, $\mathbf{v}$, the hidden units are conditionally independent and the conditional distribution of hidden unit j is given by:

$$p(h_j = 1|\mathbf{v}) = \sigma(a_j + \sum_i v_i w_{ij}) \quad (7)$$

where σ is the logistic sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. It is therefore easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. Similarly, because there are no visible-visible connections, we can easily get an unbiased sample of the state of a visible unit, i, given a hidden vector, $\mathbf{h}$:

$$p(v_i = 1|\mathbf{h}) = \sigma(b_i + \sum_j h_j w_{ij}) \quad (8)$$

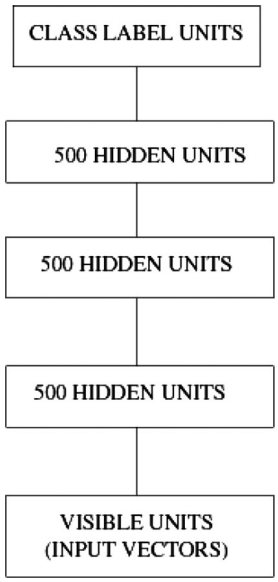

Fig. 2. Stacking RBMs to create a deep network. This architecture is used in our experiments.

Unfortunately, it is exponentially expensive to compute $\langle v_i h_j \rangle_{model}$ exactly so the contrastive divergence (CD) approximation to the gradient is used by replacing $\langle v_i h_j \rangle_{model}$ with $\langle v_i h_j \rangle_{recon}$, which is a lot easier and faster to compute [18]. $\langle v_i h_j \rangle_{recon}$ is computed by setting the visible units to a random training vector. Then the binary states of the hidden units are computed using Eqn. (7), followed by computing the binary states of the visible units using Eqn. (8). The computed visible states are a 'reconstruction' of the original visible vector. Finally, Eqn. (7) is used once more to compute the states of the hidden units from the reconstruction. The new learning rule is a crude approximation to following the gradient of the log probability of the training data, but it works well in practice and is adequate for discovering good features.

III. Learning and Using Deep Belief Networks

After training the network consisting of the visible layer and the first hidden layer, which we will refer to as RBM_1, its learned parameters, θ_1, define $p(\mathbf{v}, \mathbf{h}|\theta_1)$, $p(\mathbf{v}|\theta_1)$, $p(\mathbf{v}|\mathbf{h}, \theta_1)$, and $p(\mathbf{h}|\mathbf{v}, \theta_1)$ via Eqns. (7) and (8). The parameters of RBM_1 also define a prior distribution over hidden vectors, $p(\mathbf{h}|\theta_1)$, which is obtained by marginalizing over the space of visible vectors. This allows $p(\mathbf{v}|\theta_1)$ to be written as:

$$p(\mathbf{v}|\theta_1) = \sum_{\mathbf{h}} p(\mathbf{h}|\theta_1) p(\mathbf{v}|\mathbf{h}, \theta_1) \quad (9)$$

The idea behind training a DBN by training a stack of RBMs (as shown in Fig. 2) is to keep the $p(\mathbf{v}|\mathbf{h}, \theta_1)$ defined by RBM_1, but to improve $p(\mathbf{v})$ by replacing $p(\mathbf{h}|\theta_1)$ by a better prior over the hidden vectors. To improve $p(\mathbf{v})$, this better prior must have a smaller KL divergence than $p(\mathbf{h}|\theta_1)$ from the "aggregated posterior", which is the equally weighted mixture of the posterior distributions over the hidden vectors of RBM_1 on all N of the training cases:

$$\frac{1}{N} \sum_{\mathbf{v} \in \mathrm{train}} p(\mathbf{h}|\mathbf{v}, \theta_1) \quad (10)$$

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

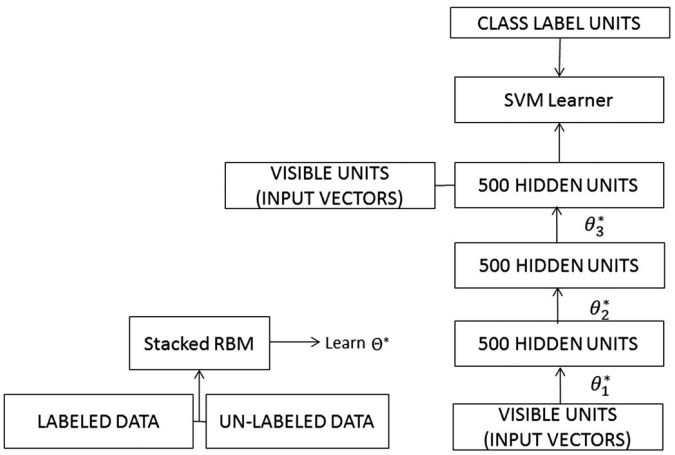

Fig. 3. Stacked RBMs (see Fig. 2) are first trained using labeled and unlabeled data and then the learned parameters are used to obtain higher level features. These higher level features in conjunction with original input feature vector are used to train a SVM classifier. This classifier is then used during evaluation.

The analogous statement for Gaussian mixture models is that the updated mixing proportion of a component should be closer to the average posterior probability of that component over all training cases.

Now consider training RBM_2, which is the network formed by using the samples from the aggregated posterior of RBM_1 as training data. It is easy to ensure that the distribution which RBM_2 defines over its visible units is identical to $p(\mathbf{h}|\theta_1)$: we simply initialize RBM_2 to be an upside-down version of RBM_1 in which the roles of visible and hidden units have been swapped. So RBM_2 has $\mathbf{h}$ as a visible vector and $\mathbf{h_2}$ as a hidden vector. Then we train RBM_2 which makes $p(\mathbf{h}|\theta_2)$ be a better model of the aggregated posterior than $p(\mathbf{h}|\theta_1)$.

After training RBM_2, we can combine the two RBMs to create a hybrid of a directed and an undirected model. $p(\mathbf{h}|\theta_2)$ is defined by the undirected RBM_2, but $p(\mathbf{v}|\mathbf{h},\theta_1)$ is defined by directed connections from the first hidden layer to the visible units. In this hybrid model, which we call a deep belief net, exact inference of $p(\mathbf{h}|\mathbf{v},\theta_1,\theta_2)$ is no longer easy because the prior over the hidden vectors is no longer defined by θ_1. However, it is proved in [19] that if we perform approximate inference for the first hidden layer by using Eqn. (7), there is a variational lower bound on the log probability of the training data that is improved every time we add another layer to the DBN, provided we add it in the appropriate way.

After training a stack of RBMs, the bottom up recognition weights of the resulting DBN can be used to initialize the weights of a multi-layer feed-forward neural network, which can then be discriminatively fine-tuned by backpropagating error derivatives. The feed-forward network is given a final "softmax" layer that computes a probability distribution over class labels and the derivative of the log probability of the correct class is backpropagated to train the incoming weights of the final layer and to discriminatively fine-tune the weights in all lower layers.

Deep belief networks (DBNs) have yielded impressive classification performance on several benchmark classification tasks, beating the state-of-the-art in several cases [11]. In principle, adding more layers improves modeling power, unless the DBN already perfectly models the data. In practice, however, little is gained by using more than about 3 hidden layers. We use the architecture shown in Fig. 3. It has three hidden layers that are pre-trained, one at a time, as the hidden layers in a stack of three RBMs without making any use of the class labels.

It is worth mentioning that the softmax output layer of a neural network is the same as a MaxEnt classifier: in other words, a neural network is a MaxEnt classifier in which the feature functions are learned.

IV. Traditional Classifiers

A. Maximum Entropy

The Maximum Entropy (MaxEnt) method is a flexible statistical modeling framework that has been widely used in many areas of natural language processing [20]. MaxEnt based classifiers do not assume statistical independence of the features that are used as predictors. As such, they allow the combination of multiple overlapping information sources [21], [20]. The information sources are combined as follows:

$$P(C|W) = \frac{e^{\sum_i \lambda_i f_i(C,W)}}{\sum_{C'} e^{\sum_j \lambda_j f_j(C',W)}}, \tag{11}$$

which describes the probability of a particular class C (e.g. call-types) given the word sequence W spoken by the caller. Notice that the denominator includes a sum over all classes C', which is essentially a normalization factor for probabilities to sum to 1. The f_i are indicator functions, or *features*, which are "activated" based on computable features on the word sequence, for example if a particular word or word pair appears, or if the parse tree contains a particular tag, etc. The MaxEnt models are trained using the improved iterative scaling algorithm [21] with Gaussian prior smoothing [20] using a single universal variance parameter of 2.0.

B. Boosting

Boosting is a method that can be used in conjunction with many learning algorithms to improve the accuracy of the learning algorithm. The idea of Boosting is to produce an accurate prediction rule by combining many moderately inaccurate (weak) rules into a single classifier. At each iteration, boosing adds a new (weak) prediction rule that focuses on samples that are incorrectly classified by the current combined predictor. Even though Boosting is known to be sensitive to noisy data and outliers, in some problems, it is less susceptible to overfitting than most machine learning algorithms. We used a specific implementation of Boosting, AdaBoost using decision stumps, which is described in [6]. Boosting has been applied to a number of natural language processing tasks in the past [9].

C. Support Vector Machines

Support vector machines (SVMs) are supervised learning methods used for classification. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary classifier.

SVMs are derived from the theory of structural risk minimization [7]. SVMs learn the boundaries between samples of

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

the two classes by mapping these sample points into a higher dimensional space. SVMs construct a hyperplane or a set of hyperplanes in a high-dimensional space, which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (the "functional margin"), since in general the larger the margin the lower the generalization error of the classifier. The hyperplane separating these regions is found by maximizing the margin between closest sample points belonging to competing classes. In addition to performing linear classification, SVMs can efficiently perform non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Much of the flexibility and classification power of SVMs resides in the choice of kernel. Some of the commonly used kernels are linear, polynomial and radial basis functions. In this work, we chose linear kernels to train the SVM since computationally it is faster compared to other kernels, yet there is no significant difference in performance for the current task. This is a fairly standard result for applying SVMs in natural language processing since we are already using a high-dimensional feature vector.

V. Experimental Results and Discussion

The call-routing task considered in this paper is from a call-center customer hotline that gives technical assistance for a Fortune–500 company [22]. The call-routing system selects one of 35 call–types. The training data has 27 K automatically transcribed utterances amounting to 178 K words. This data is split into sets containing {1 K, 2 K, 3 K, 4 K, 5 K, 6 K, 7 K, 8 K, 9 K, 10 K} and 27 K utterances respectively. These sets will be referred to in a similar fashion. The purpose of this split is to investigate various training data sizes and their effects on the learning methods. We also have two separate datasets containing about 3.2 K and 5.6 K sentences that are used as development and test data, respectively. All of these datasets are hand–labeled with call–types. In all the classification methods employed here we used vectors of individual word counts as the inputs to the models. For the DBNs, the counts were clipped at 1 to allow them to be modeled by binary units.

In our experiments with the development data we found that hidden layers of $500 \rightarrow 500 \rightarrow 500$ provided slightly better results than the other hidden layer sizes that we tried. The model architecture is shown in Fig. 3. The individual RBM models were trained in an unsupervised fashion using contrastive divergence learning with 1 step of Gibbs sampling (CD-1). The training phase made 100 passes (epochs) through the training dataset. The weights of each RBM were initialized with small random values sampled from a zero-mean normal distribution with standard deviation 0.01 and updated using a learning rate of 0.01/batch-size, momentum of 0.9, and a weight decay of 0.001.

For the discriminative fine-tuning, we use stochastic gradient descent (SGD) and we also set the number of iterations by using early stopping according to the validation set classification error. To reduce computation time, we select the SGD learning rate, momentum parameter and other parameters by maximizing the accuracy on the development set.

TABLE I
Package Shipment Task: Accuracy for traditional and DBN based classifiers

Action Classification Accuracy (%)							
Labeled Data	MaxEnt	SVM	Boosting	DBN	DBN-1	DBN-2	DBN-3
1K	76.0	77.8	**79.6**	78.1	78.4	78.3	78.6
2K	80.4	82.2	83.6	82.6	83.7	83.4	**84.1**
3K	82.2	84.3	85.1	84.4	85.3	84.8	**85.6**
4K	83.5	85.3	84.6	85.5	86.4	85.9	**86.6**
5K	84.6	86.2	85.9	86.2	86.9	86.7	**87.4**
6K	85.5	87.0	86.3	87.0	87.4	87.3	**87.8**
7K	86.2	87.7	86.3	87.8	88.0	88.2	**88.4**
8K	86.5	88.0	87.2	88.0	88.0	88.1	**88.1**
9K	87.2	88.5	87.5	88.7	88.8	88.8	**88.9**
10K	87.6	88.5	87.7	88.7	88.7	**88.9**	**88.9**
27K	89.7	90.3	88.1	90.3	90.3	**90.8**	**90.8**

In Table I, we present the results on the test data for SVMs, MaxEnt, Boosting and DBNs. Various classifier parameters (e.g. smoothing priors for MaxEnt learning, and kernel selection for SVMs) are tuned on the development data. Each classifier is trained using the amount of labeled data given in the first column. Looking first at the traditional classifiers, we notice that the SVM classifier obtained 77.8% accuracy using 1 K labeled data. The corresponding figures for the MaxEnt classifier and the Boosting based classifier are 76.0% and 79.6% respectively. Not only for 1 K labeled data but also for 2 K and 3 K data, Boosting provides the best performance. However, for larger amounts of training data, the SVM consistently outperformed both MaxEnt and Boosting, which is in agreement with other studies [22]. The DBN (4th column) performed as well as or slightly better than SVMs for all sizes of training set. When trained on all of the training data, they had identical performance, achieving 90.3% accuracy.

In this paper we pursued two of the three future research directions suggested in [15]. The first extension was using additional unlabeled data to train the RBMs, since typically there is a lot more unlabeled data available than labeled data. In our experiments, for smaller chunks of labeled data, the entire 27 K labeled data is treated as unlabeled data to train the DBN. For example, when 1 K labeled data is used to train the DBN, we used 27 K to train the corresponding RBMs. We have repeated the same steps with different amounts of labeled data given in Table I. The second direction of research was to treat the DBN as a feature extractor and use these features as input to a separate classifier. We first trained a DBN and then for each utterance, we generated the activity at the top layer. This activity along with the original features were concatenated and used as input to an SVM classifier. Fig. 3 shows the schematics of the setup.

We provided additional experimental results for three scenarios: a) using additional unlabeled data to train the RBMs (DBN-1), b) using DBN learned features as input additional features to SVM classifier (DBN-2), and c) combining the previous two scenarios (DBN-3). Using additional unlabeled data provided large gains when the ratio of unlabeled to labeled data size is large, as shown in the column of DBN-1 column in Table I. For example, when we have 27 K unlabeled data to train RBMs but only 2 K labeled data to fine tune the DBNs the gain is 1.1%. Likewise, when the labeled data is 3 K the gain is 0.9%. However, as the ratio of the labeled data to unlabeled data gets larger

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

we do not observe gains from using additional unlabeled data. We note that the amount of unlabeled data considered here is fairly small. In many applications however, the amount of unlabeled data can be substantially larger than the labeled data. It is one of our future research work directions to investigate using substantially larger amounts of unlabeled data to train RBMs in a separate application.

In the table we also show feature combination results where DBN learned features are combined with the original features (DBN-2) as input to an SVM classifier. The results indicate that we get consistent gains when DBN based features are combined with the original features across all labeled data sizes. Finally, we combine DBN based features where RBMs are trained with large (relative to the labeled data) collection of unlabeled data with the original features using an SVM classifier. This set-up is called DBN-3 and the results are given in the last column of Table I. The results show that DBN-3 improves the call routing performance consistently across all data sizes with the exception of the 1 K data size where Boosting performs better. For smaller amounts of labeled data the performance improvements over SVM are significant. For example, 0.8%, 1.9%, 1.2%, 1.3% and 1.2% absolute improvements are obtained for 1 K through 5 K labeled data amounts. The improvements were smaller but consistent all the way to 27 K labeled data. The performance gains are coming largely from using unlabeled data, which is used to train RBMs, when the labeled data size is small. The results indicate that gains for DBN-1 and DBN-2 are approximately additive.

We also investigate whether binarization of the features for DBNs give them an advantage by also testing the SVM classifier with binarized word count features. The n–gram features are formed based on the existence of these features regardless of the actual counts that they are observed in the sentence. There are about 15% of the sentences that had n–gram features of count two or more. However, classification results across all data sizes show that the feature binarization did not change the SVM performance (the changes were in the second decimal).

VI. Conclusion and Future Work

This work presented a successful application of Deep Belief Nets (DBNs) to a natural language call–routing task. DBNs use unsupervised learning to discover multiple layers of features that are then used in a feed–forward neural network and fine–tuned to optimize discrimination. When the amount of training data is limited, unsupervised feature discovery makes DBNs less prone to overfitting than feedforward neural networks initialized with random weights, and it also makes it easier to train neural networks with many hidden layers.

DBNs produce better classification results than several other widely used learning techniques, outperforming Maximum Entropy and Boosting based classifiers. Their performance is almost identical to SVMs, which are the best of the other techniques that we investigated.

We further extended our initial work by treating DBNs as feature generators to capture and model the underlying structure in the input data. The learned features are used in conjunction with the original inputs to do classification using an SVM. We also leveraged additional unlabeled data to improve the modeling performance. Both of these extensions resulted in additional improvement in call–routing classification performance. In the future, we plan to consider DBNs for sequence tagging for slot detection and entity tagging in spoken language understanding.

References

[1] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proc. DARPA Workshop Speech Nat. Lang.*, Hidden Valley, PA, USA, Jun. 1990.

[2] Y.-Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *Proc. ICSLP*, Pittsburgh, PA, USA, Sep. 2006.

[3] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proc. Interspeech*, Antwerp, Belgium, 2007.

[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *Proc. Int. Conf. Mach. Learn.*, 2001.

[5] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may I help you?," *Speech Commun.*, vol. 23, pp. 113–127, 1997.

[6] R. E. Schapire and Y. Singer, "Boostexter: A boosting based system for text categorization," *Mach. Learn.*, vol. 39, no. 2/3, pp. 135–168, 2000.

[7] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer–Verlag, 1995.

[8] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proc. ICASSP*, Hong Kong, Apr. 2003, pp. 632–635.

[9] *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, G. Tur and R. D. Mori, Eds. New York, NY, USA: Wiley, 2011.

[10] G. E. Hinton, "Learning multiple layers of representation," *TRENDS Cognitive Sci.*, vol. 11, no. 10, pp. 428–434, 2007.

[11] G. E. Dahl, M. Ranzato, A. Momamed, and G. E. Hinton, "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machines," in *Advances in Neural Information Processing Systems NIPS*. Cambridge, MA, USA: MIT Press, 2010.

[12] M. Welling, M. Rosen-Zvi, and G. E. Hinton, "Exponential family of harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2005, pp. 1481–1488.

[13] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, and P. Vincent, "Why Does Unsupervised Pre-training Help Deep Learning?," *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, 2010.

[14] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2002, vol. 11.

[15] R. Sarikaya, G. Hinton, and B. Ramabhadran, "Deep belief networks for natural language call-routing," in *Proc. ICASSP*, 2011, pp. 5680–5683.

[16] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, pp. 1771–1800, 2002.

[17] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," Univ. of Toronto Mach. Learn. Tech. Rep., UTML TR 2010-003.

[18] G. E. Hinton, "Training product of experts by minimizing constrastive divergence," *Neural Comput.*, vol. 14, no. 18, pp. 1527–1554, 2002.

[19] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Adv. Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[20] S. Chen and R. Rosenfeld, "A survey of smoothing techniques for ME models," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 37–50, Jan. 2001.

[21] S. D. Pietra, V. D. Pietra, and J. Lafferty, "Inducing features of random fields," *IEEE Trans. Pattern. Anal. Mach. Intell.*, vol. 19, no. 4, pp. 380–93, Apr. 1997.

[22] R. Sarikaya, H-K. J. Kuo, V. Goel, and Y. Gao, "Exploiting Unlabeled Data Using Multiple Classifiers for Improved Natural Language Call-Routing," in *Proc. Interspeech*, Lisbon, Portugal, Sep. 2005.

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.

Ruhi Sarikaya is a principal scientist and the manager of language understanding and dialog systems group at Microsoft. He was a research staff member and team lead in the Human Language Technologies Group at IBM T.J. Watson Research Center for ten years. Prior to joining IBM in 2001 he was a researcher at the Center for Spoken Language Research (CSLR) at the University of Colorado at Boulder for two years. He also spent the summer of 1999 at the Panasonic Speech Technology Laboratory, Santa Barbara, CA. He received the B.S. degree from Bilkent University, Turkey in 1995, M.S. degree from Clemson University, SC in 1997 and the Ph.D. degree from Duke University, NC in 2001 all in electrical and computer engineering. He has published over 70 technical papers in refereed journal and conference proceedings and, is inventor of 25 patents in the area of speech and natural language processing. At IBM he has received several prestigious awards for his work including two Outstanding Technical Achievement Awards (2005 and 2008) and two Research Division Awards (2005 and 2007). Dr. Sarikaya has served as the general co-chair of IEEE SLT 2012, publicity chair of IEEE ASRU 2005 and as associate editors of IEEE TRANSACTIONS ON AUDIO SPEECH AND LANGUAGE PROCESSING and IEEE SIGNAL PROCESSING LETTERS. He also served as the lead guest editor of the special issue on Processing Morphologically-Rich Languages for IEEE TRANSACTIONS ON AUDIO SPEECH AND LANGUAGE PROCESSING and gave a tutorial on Processing Morphologically Rich Languages at Interspeech 2007.

His past and present research interests span all aspects of speech and language processing including natural language processing, spoken dialog systems, speech recognition, machine translation, machine learning, speech-to-speech translation, speaker identification/verification, digital signal processing and statistical modeling. Dr. Sarikaya is a member of IEEE (senior member), ACL and ISCA.

Geoffrey Hinton received his Ph.D. degree in Artificial Intelligence from the University of Edinburgh in 1978. He spent five years as a faculty member at Carnegie Mellon University, Pittsburgh, Pennsylvania, and he is currently a Distinguished Professor at the University of Toronto and a Distinguished Researcher at Google. He is a fellow of the Royal Society and an honorary foreign member of the American Academy of Arts and Sciences. His awards include the David E. Rumelhart Prize, the International Joint Conference on Artificial Intelligence Research Excellence Award, the Killam Prize for Engineering and the Gerhard Herzberg Canada Gold Medal for Science and Engineering. He was one of the researchers who introduced the back-propagation algorithm. His other contributions include Boltzmann machines, distributed representations, time-delay neural nets, mixtures of experts, variational learning, contrastive divergence learning, and Deep Belief Nets.

Anoop Deoras is a research scientist at Microsoft. He received the B.E. degree in Electronics and Telecommunication Engineering from College of Engineering, Pune India in 2003, M.S. degree in Applied Math. & Statistics in 2010 and a M.S. and Ph.D. in Electrical & Computer Engineering from Johns Hopkins University in 2011. He is interested in applying machine learning techniques to speech recognition and spoken language understanding. In his PhD thesis, he investigated several decoding techniques for incorporating complex and long span language models, such as recurrent neural network language models, in automatic speech recognition setup. He is a member of IEEE, ISCA and ACL.

Authorized licensed use limited to: University of Regina IEL. Downloaded on April 10,2020 at 15:44:58 UTC from IEEE Xplore. Restrictions apply.