Deep Belief Network based Semantic Taggers for Spoken Language Understanding

Anoop Deoras, Ruhi Sarikaya

Microsoft Corporation, 1065 La Avenida, Mountain View, CA

Anoop.Deoras@microsoft.com, Ruhi.Sarikaya@microsoft.com

Abstract

This paper investigates the use of deep belief networks (DBN) for semantic tagging, a sequence classification task, in spoken language understanding (SLU). We evaluate the performance of the DBN based sequence tagger on the well-studied ATIS task and compare our technique to conditional random fields (CRF), a state-of-the-art classifier for sequence classification. In conjunction with lexical and named entity features, we also use dependency parser based syntactic features and part of speech (POS) tags [1]. Under both noisy conditions (output of automatic speech recognition system) and clean conditions (manual transcriptions), our deep belief network based sequence tagger outperforms the best CRF based system described in [1] by an absolute 2% and 1% F-measure, respectively.Upon carrying out an analysis of cases where CRF and DBN models made different predictions, we observed that when discrete features are projected onto a continuous space during neural network training, the model learns to cluster these features leading to its improved generalization capability, relative to a CRF model, especially in cases where some features are either missing or noisy.

Index Terms: SLU, DBN, CRF, ASR

1. Introduction

Spoken language understanding (SLU) systems aim to automatically identify the intent of the user as expressed in natural language, extract associated arguments or slots, and take actions accordingly to satisfy the user's requests [2]. The SLU task is mostly coined by the DARPA (Defense Advanced Research Program Agency) Airline Travel Information System (ATIS) project during 80s [3]. The ATIS task consisted of spoken queries on flight-related information. An example utterance is *I want to fly to Boston from New York next week*. Understanding was reduced to the problem of extracting task-specific arguments (usually referred to as 'slots' in SLU literature), such as *Destination* (tag for 'Boston') and *Departure Date* (tag for 'next week'). Participating systems employed either a data-driven statistical approach [4, 5] or a knowledge-based approach [6, 7, 8].

The state-of-the-art approaches for slot filling [9, 10, among others] use discriminative statistical models, such as conditional random fields, (CRFs) [11], for modeling. Slot filling is framed as a sequence classification problem to obtain the most probable slot sequence given some word sequence.

Traditional spoken language understanding systems follow a cascade architecture where an Automatic Speech Recognition engine (ASR) is connected to understanding modules such as slot sequence classifiers and intent detectors. In this paper, we focus on the slot sequence modeling task alone. More formally, given an acoustic signal, $\mathbf{A}$, ASR outputs most likely word sequence, $\mathbf{W}^*$ given by: $\mathbf{W}^* = \text{argmax}_{\mathbf{W}\in\mathcal{W}} P(\mathbf{A}|\mathbf{W}) \times P(\mathbf{W})$ [12]. Typically in cascade systems, this ASR 1-best hypothesis, $\mathbf{W}^*$, is then fed into SLU system (say targeting slot sequence classification) to output most likely sequence of slots, $\mathbf{C}^*$ given by: $\mathbf{C}^* = \text{argmax}_{\mathbf{C}\in\mathcal{C}} P(\mathbf{C}|\mathbf{W}^*)$ where $\mathbf{W} = w_1, ..., w_T$ is the input word sequence and $\mathbf{C} = c_1, ..., c_T, c_t \in \mathcal{C}$ is the sequence of associated slot labels in $\mathcal{C}$. CRFs are global models maximizing the likelihood of the entire slot sequence given the sequence of words. They decompose the conditional probability of slot sequence into product of local potential functions: $\psi(c_t, c_{t-1}, \gamma_t(\mathbf{W}))$, each capturing feature for the context at time instant t, here represented by $\gamma_t(\mathbf{W})$. More formally:

$$\begin{aligned} P(\mathbf{C}|\mathbf{W}) &= \frac{1}{Z(\mathbf{W})}\prod_{t=1}^{T}\psi(c_t, c_1^{t-1}, \mathbf{W}) \\ &\approx \frac{1}{Z(\mathbf{W})}\prod_{t=1}^{T}\psi\Big(c_t, c_{t-1}, \gamma_t(\mathbf{W})\Big), \end{aligned} \quad (1)$$

where $Z(\mathbf{W})$ is the partition function [11]. Typical features that are extracted at every time step include current word token, lexical word tokens from left and right n-gram context, named entity features for the corresponding local context, syntactic features such as part of speech tags (POS) etc. Apart from the features that are specific to the observation sequence – $\mathbf{W}$, slot label, c_{t-1}, assigned to the word at the previous time instant is used as a feature too. CRF assigns weight to each feature whose value is dependent upon the frequency with which that particular feature is observed in the training data. Models trained with such large number of discrete feature combinations are often susceptible to data sparsity problem. Neural networks, however, project input features onto a continuous space where inherently these features cluster together leading to model's relatively better generalization capability. This is very motivating especially in cases such as semantic tagging where traditionally quite a large number of features (individually or jointly with other features) have been used. It has been shown that with more features, the semantic tagging model learns to do a better job at predicting semantic tags, however, when some of the features provided during test time are noisy or missing, the CRF model fails to assign reliable probabilities to correct tags [13]. If, however, neural network based architecture is used for this problem, we believe, the model may be able to generalize well and it may improve its prediction power.

To investigate this hypothesis, in this paper, we use deep belief networks, a class of neural networks, to solve semantic tagging task for spoken language understanding using a variety of features. The name 'deep belief network' is given to the class of those neural networks which have many (more than 1) hidden

layers and also involve a pre-training initialization step (much more principled than random initialization) before the standard back propagation learning phase. Thus deep belief networks – DBNs, are those deep neural networks – DNNs, which are initialized using a pair wise unsupervised local learning of restricted Boltzman machines (RBM) [14, 15]. Recent advances in DBNs for acoustic modeling [16], image classification [15] and more recently in utterance classification [17] motivated us to investigate the use of DBNs as the neural network models for semantic tagging, a task, to our best knowledge, has never been tried before using neural networks.

We use DBNs as discriminative models to model posterior distribution of the slot sequence given the sequence of words, similar to CRFs. However, unlike CRFs, in DBN based approach, we decompose the conditional probability of slot sequence into product of local *probability* functions, each modeling the distribution of a particular slot tag given the context at that time instant. More formally, we decompose $P(\mathbf{C}|\mathbf{W})$ as shown below:

$$P(\mathbf{C}|\mathbf{W}) = \prod_{t=1}^{T} P(c_t|c_1^{t-1}, \mathbf{W}) \approx \prod_{t=1}^{T} P\Big(c_t|c_{t-1}, \gamma_t(\mathbf{W})\Big) \tag{2}$$

where $\gamma_t(\mathbf{W}))$ captures local features (lexical, named entities (NE), part of speech (POS) etc.) at time t, similar to CRFs. However, unlike CRFs, where each local model is an unnormalized potential function $\psi\Big(c_t, c_{t-1}, \gamma_t(\mathbf{W})\Big)$ (see 1), we use a probability distribution function, $P\Big(c_t|c_{t-1}, \gamma_t(\mathbf{W})\Big)$, thus removing the necessity to normalize the product over the entire sequence. Once the individual local models are trained, Viterbi decoding is carried out to find the best slot sequence given the sequence of words.

Unlike tasks such as acoustic modeling and digit recognition, where the input feature vector presented to DBNs is dense and real valued, classification tasks in natural language processing have input features which are often sparse and at-least as long as the size of the lexical vocabulary (in thousands). Huge input feature vector is a bottleneck for the pre-training phase of DBN training as each step of it involves reconstructing (through Gibbs sampling from an exponential distribution) all the elements on the input feature vector. To overcome this limitation, in this paper, we propose *discriminative embedding* technique which projects the sparse and large input layer onto a small, dense and real valued feature vector, which is then subsequently used for pre-training the network and then to do discriminative classification. In the past, researchers have used latent Dirichlet analysis (LDA) or neural network language modeling (LM) based technique to obtain word embeddings (see [18] for a comprehensive survey of various word embedding techniques) to overcome similar limitations. In our work, we found that if the embedding procedure is carried out in a discriminative fashion (i.e. by minimizing the tagging errors) rather than in an unsupervised fashion (LDA and LM like methods aim to maximize the likelihood of observations – $\mathbf{W}$, without taking into account the tags associated with them), it results in a much better feature representation as it is more suited to the task at hand. Moreover, in our approach, we project the totality of all features onto the continuous space resulting into an even better embedding. LDA and LM based embedding techniques have a limited scope as they can robustly provide embeddings of lexical features only.

The rest of the paper is organized as follows: In Sec. 2 we describe, in brief, the model structure of a deep belief network. In subsection Sec. 2.1, we describe the proposed modification to the DBN architecture to obtain discriminative embedding features for the purpose of pre-training and discriminative classification via back propagation training. We present experimental results in Sec. 3, analysis of some results in Sec. 4 and then finally conclude in Sec. 5.

2. Deep Belief Network Model Description

Deep Belief Network (DBN) is built up with a stack of probabilistic model called Restricted Boltzmann Machine (RBM) [14, 15]. RBMs are trained using the contrastive divergence (CD) [14] learning procedure. Each RBM layer is trained by using the previous layer's hidden units ($\mathbf{h}$) as input/visible units ($\mathbf{v}$). Deep networks have higher modeling capacity than shallow networks but are also much harder to train, because the objective function of a deep network is highly non-convex function of the parameters, with many distinct local minimum in parameter space. Contrastive divergence based pre-training of these RBM layers is carried out to initialize the weights of DBN. After the deep network is initialized, back-propagation [19] algorithm is used to fine tune the weights of deep networks in a discriminative fashion. We refer interested readers to [14] for a detailed description about RBM based pre-training technique.

As described above, a DBN is formed by stacking multiple RBMs on top of each other. Thus input to i^{th} RBM is output of $i-1^{th}$ RBM. We will represent i^{th} stacked RBM by RBM_i and denote the weight parameters for this layer by Θ_i. Thus once RBM_1 is constructed and pre-trained, we obtain the posterior distribution over hidden vectors $P(\mathbf{h}|\mathbf{v};\Theta_i)$ and sample $\mathbf{h}$, which then becomes input for second RBM layer: RBM_2. Continuing in this fashion, we form a multi layer deep belief network with weights initialized by the pre-training procedure.

The topmost layer of the neural network uses a soft max function to compute the probability distribution over class labels. A back-propagation algorithm is then used to fine tune weights of the neural network. In our work, we use sigmoid activation function to obtain values at various hidden and output units given the corresponding inputs.

2.1. Discriminative Embedding Techniques

For natural language processing applications, n-gram lexical features are represented as a concatenation of n "1 of N coded" binary vectors, where N is the size of the lexical vocabulary. With a lexical vocabulary running in thousands, this feature representation becomes really huge. This is not so much of a problem for back-propagation because in it, one needs to update only those weights which are connected to non-zero input units (at most n). It is, however, the pre-training phase, for which large input layer causes a bottleneck, as it has to be reconstructed in each epoch.[1] To solve this problem, we propose to divide our training procedure in 3 phases:

1. **Obtain Embeddings:** For a network with sparse input layer, 1 hidden layer and output label layer, randomly initialize the weights and run back-propagation training to minimize the classification error and obtain set of weights between input and hidden layer: Θ_1. For every input feature, we obtain the values at the hidden layer by forward propagating the units through the trained network. The hidden units act as the feature embeddings for the corresponding inputs.

[1] In our experience, sub-sampling input features for the purpose of reconstruction led us to sub-optimal results. However, recent work [20] show promise and as part of our future work, we may explore this idea.

2. **Do Pre-training with embedded features:** Obtain embedding features for each input sample. With this as the input, form a DBN by stacking RBMs on top of each other. We will refer to this as RBM stack. With random initialization of weights, do pre-training of this network and obtain set of weights: $\Theta_2, \Theta_3, \ldots$.

3. **Fine tune the weights with back-propagation:** Attach the original sparse binary input layer, the above RBM stack and also the output label layer. Randomly initialize the weights between top most RBM layer and output label layer. Initialize the weights between input layer and first RBM layer with Θ_1. Initialize the weights of RBM stack with $\Theta_2, \Theta_3, \ldots$. Fine tune all these weights except Θ_1 with back-propagation to minimize the classification error rate (one could re-tune Θ_1 although in our work, we did not find it necessary). This final network is then used during decoding.

2.2. Learning Techniques

We divide our training data into several mini batches and learn neural network weights (parameters of the model) using online version of conjugate gradient (CG) optimization. Unlike stochastic gradient descent (SGD) optimization, conjugate gradient does not require tuning of the learning parameter and is generally believed to converge faster. However, rather than using conjugate gradient optimization in batch mode (which can be impractical for large training corpus), we use it under online (or stochastic) setting. For each mini batch, we update the parameters by running CG for a finite number of steps. Thus rather than learning the local optimum solution for each mini batch, we truncate the search after a small number of steps (typically 3). The weights, however, have to be regularized to avoid over-fitting of the model on the training data. Typically L2 regularization of the entire weight vector is carried out as a way to penalize very large values.

Recently Hinton et.al.[21] proposed a weight constraining process for regularizing the neural network model training. Instead of penalizing the L2 norm of the whole weight vector, an upper bound is set on the L2 norm of the incoming weight vector for each hidden unit. Whenever a weight update violates this constraint, the incoming weights are scaled down until the constraint is satisfied.

In our work, we used a variation of the above proposed weight constraining technique for regularization. Instead of continually scaling down the L2 norm of the incoming weight vector until the constraint is satisfied, we constrain each individual weight only once at every update. Thus if the weight update increases the absolute value of a weight above a threshold we scale it down by dividing it with some constant. Value of this constant and the threshold has to be chosen by doing a cross validation experiment on some held out set.

In order to find the effect of pre-training on final trained weights, we ran an experiment in which we compared the L2 norms of incoming weights at each hidden unit with and without pre-training initialization. From Figure 1, it can be seen that when weights are randomly initialized followed by back-propagation training ('$\times$' in Fig. 1), the final weights tend to have a lot of variance ($\mu = 292, \sigma = 283$). Pre-training based initialization followed by back-propagation fine tuning (without weight constraining at hidden unit level) ('$\blacklozenge$') reduces this variance to a great extent ($\mu = 178, \sigma = 117$). When models are randomly initialized followed by back-propagation training (without any pre-training) and weight constraining is done during training ('$\bullet$'), the individual weights get regularized extremely well ($\mu = 93, \sigma = 27$). For all the 3 settings above, the starting initial weights were exactly the same, suggesting that the differences in the final weights were only due to the effects of weight constraining and pre-training. In our experience, we find pre-training as an implicit way to regularize network's parameters. Explicit regularization by way of weight constraining regularizes the model further. Looking at the L2 norm distribution of weights trained with pre-training based initialization, gives us a range of values within which search for an explicit threshold to cap the individual weights can be carried out.

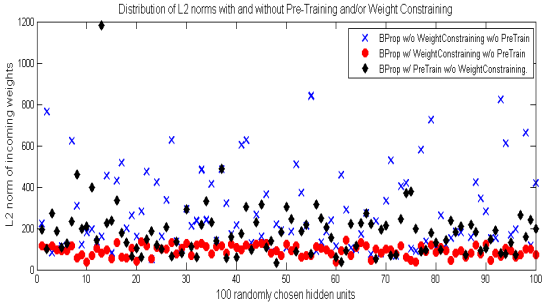

Figure 1: Distribution of L2 norms of incoming weights to individual hidden units obtained after back-propagation training with and without pre-training and/or weight constraining.

3. Experiments and Results

We evaluate the deep belief network based sequence taggers on the most commonly used data set for SLU research – ATIS corpus [3]. In this paper, we used the ATIS corpus as used in [13, 1, 22, 10]. The training set contains 4978 utterances (67k word tokens and 21k tags), while the test set contains 893 utterances (11k word tokens and 3.7k tags). Named entities are further marked via table lookup, including domain specific entities such as city, airline, airport names and dates. The ATIS utterances are represented using semantic frames, where each sentence has a goal or goals and slots filled with phrases. The values of the slots are not normalized or interpreted. An example utterance & annotation in In-Out-Begin (IOB) representation is shown in Fig. 2.

Tur et.al.[1] trained an automatic speech recognizer using generic dictation models using the Microsoft's commercial speech recognition system. We used the ASR output from their setup. The WER for the transcribed ATIS test data was 13.76%.

We trained the deep neural network (using the steps 1 through 3 in Sec.2.1) with 2 hidden layers of sizes 100 and 200 units respectively.[2] The threshold for weight constraining was 2. CRF models were trained using CRF++ [23] toolkit with L2 regularization. Both classifiers were fed with exact same feature set. For DBN, we trained local probabilistic models at every time instant t: $P(c_t|c_{t-1}, \gamma_t(\mathbf{W}))$ (see Eqn. 2), where c_t is the current tag, c_{t-1} is one of the hypothesized tags from immediate past and $\gamma_t(\mathbf{W})$ are the features extracted at time instant t. In all, we used 4 classes of features:

1. **Lexical (Lex):** At every time instant, we used 2 words from left, 2 words from right and the current word as lex-

[2] We thank Xin Wu for implementing initial version of the DBN software program.

Words						
..	from	tacoma	to	san	jose	..
↓	↓	↓	↓	↓	↓	↓
..	O	B-from.city	O	B-to.city	I-to.city	..
Slot-Tags						

Figure 2: An example utterance semantically annotated in In-Out-Begin (IOB) format.

Setup	Manual		ASR	
	CRF	DBN	CRF	DBN
Lex	91.0	93.2	86.7	86.4
+ NE	94.4	95.3	91.9	92.8
+ Sntc	94.6	96.0	91.6	93.5

Table 1: Performance comparison (using F-measure) of CRF and DBN classifiers on ATIS test set under clean (manual transcriptions) and noisy (ASR output) conditions.

ical features. The lexical vocabulary for ATIS corpus is 895 words, so the lexical features resulted in a vector of size 4475 bits ($= 5 \times 895$), with only 5 bits (corresponding to 5 words) switched on, while all the other off. Out of vocabulary (OOV) words were represented with a '0 of N' coded vector.

2. **Named Entities (NE):** Each word in the ATIS corpus is marked with a named entity. So for every word in the 5 word window used in lexical features, we form a similar binary vector of size equal to the size of the named entity gazetteer. The total number of named entity tags that came with ATIS corpus were 134, so we formed a vector of size 670 bits ($= 5 \times 134$) again with a maximum of 5 bits on.

3. **Syntactic Features (Sntc):** In a study carried out by Tur et.al. [13], authors showed that in-spite of using 5 word window and named entities, some of the errors in the ATIS domain were caused due to model's inability to capture cues occurring far beyond the n-gram window. They proposed to solve this problem by using part of speech tags and some long span features obtained via dependency parsers [1]. We used head words for the current word only while part of speech tags were extracted for the 5 word window.[3] The total number of POS tags were 41 resulting in the feature vector of size 205 bits ($= 5 \times 41$). Feature vector for head words (immediate head word and predicate head word) were of size equal to lexical vocabulary, hence resulting in a vector of size 1790 ($= 2 \times 895$).

4. **Slot-Tag:** We used hypothesized slot-tag from the immediate past as an additional feature. The total number of slot-tags for the ATIS corpus is 128. Thus for the final model, the size of the input layer was 7268 bits with a maximum of 18 bits on (corresponding to 5 words, 2 head words, 5 POS, 5 NE and 1 hypothesized slot-tag). For both CRF and DBN, this feature was always used irrespective of whether syntactic features and/or named entities were used or not.

There are 128 slot-tags in the ATIS domain and thus the output layer of the deep belief network has these many units. Each unit corresponds to a particular slot-tag. Value obtained at each output unit (after applying soft-max) corresponds to the likelihood of seeing the associated slot-tag given the context represented by a 7268 bit long input vector.

Following the literature [10], F-measure for evaluating the model performance was used. Slot sequence was represented in the conventional IOB representation (see Fig. 2) and CoNLL evaluation script[4] was used to compute F-measure. From Table 1, we can clearly see that DBN model outperforms CRF based sequence taggers significantly.[5] With a full gamut of features, we achieve $\mathbf{96.0}\%$ F-measure on manual transcriptions, which is 1.4% better than that by CRF. Compared to the previous best results (95%) reported in [1], we achieve a 1.0% absolute improvement.[6] The results under noisy conditions are even more encouraging. On ASR output, DBN based model outperforms CRF by as much as 1.9% absolute. Such improvements are both quantitatively as well as qualitatively significant. Output of an ASR is usually far from perfect and hence if we are able to do significantly better sequence tagging on such noisy text, it is a very favorable setting for any real life spoken language understanding system.

4. Analysis

Upon further analysis, we observed that DBN was able to produce correct tags even in those cases for which 'observation-tag' did not occur in training data. An example sentence is: '*Does tacoma airport offer transportation*'. In the training data, the word *tacoma* never occurs together with *airport* and is always labeled as *B-city-name* rather than as *B-airport-name*. However, the word *airport* is tagged as *I-airport-name* many times in the training data. DBN model tags *tacoma airport* with *airport-name* slots, while CRF fails to tag it at all (due to inconsistency of *B-city-name* tag and *I-airport-name* tag going side by side). Since *tacoma airport* bi-gram feature was unknown to CRF, it defaulted to the unigram based features resulting into no tags due to inconsistent predicted tag combination for the word *tacoma* and *airport*. DBN models, however, were able to generalize much better. By projecting the context onto a continuous space, the models learned the fact that the word preceding *airport* is likely to be an airport name rather than anything else.

5. Conclusion and Future Work

In this paper, we demonstrated a deep belief network based slot sequence classifier. We applied it on the well studied spoken language understanding task of ATIS and obtained new state-of-the-art performances, outperforming the best CRF based system [1]. As part of the future work, we plan to use these DBN models for sequence tagging on word graphs extending our previous research work [24, 25]. In our companion paper [26], we used word confusions to improve various spoken language understanding tasks in a CRF framework. Use of DBNs would only be a natural extension of this work.

We also plan to investigate some model adaptation techniques to benefit from the huge amounts of *unsupervised* data available in the form of search queries.

[3] We thank Tur et.al.[1] for sharing with us these features.

[4] http://www.cnts.ua.ac.be/conll2000/chunking/output.html

[5] The results are statistically significant with a $p<0.01$.

[6] The difference between our CRF performance versus that reported in [1] can be attributed to different CRF toolkits and/or regularization techniques.

6. References

[1] G. Tur, D. Hakkani-Tur, L. Heck, and S. Parthasarathy, "Sentence Simplification for Spoken Language Understanding," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5628 –5631.

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

[3] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the DARPA Workshop on Speech and Natural Language*, Hidden Valley, PA, June 1990.

[4] R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon, "A speech understanding system based on statistical representation of semantics," in *Proceedings of the ICASSP*, San Francisco, CA, March 1992.

[5] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden understanding models of natural language," in *Proceedings of the ACL*, Las Cruces, NM, June 1994.

[6] W. Ward and S.Issar, "Recent improvements in the CMU spoken language understanding system," in *Proceedings of the ARPA HLT Workshop*, March 1994, pp. 213–216.

[7] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1, pp. 61–86, 1992.

[8] J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran, "Gemini: A natural language system for spoken language understanding," in *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, NJ, March 1993.

[9] Y.-Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *Proceedings of the ICSLP*, Pittsburgh, PA, September 2006.

[10] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proceedings of the Interspeech*, Antwerp, Belgium, 2007.

[11] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the ICML*, Williamstown, MA, 2001.

[12] F. Jelinek, *Statistical methods for speech recognition*. Cambridge, MA, USA: MIT Press, 1997.

[13] G. Tur, D. Hakkani-Tur, and L. Heck, "What is left to be understood in ATIS?" in *Proc. of IEEE Spoken Language Technology Workshop (SLT)*, Dec 2010, pp. 19 –24.

[14] G.E.Hinton, "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

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

[16] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained dnns for large vocabulary speech recognition," *IEEE Trans. Audio, Speech, and Lang. Proc.*, Jan. 2012.

[17] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proceedings of the ICASSP*, Prague, Czech Republic, 2011.

[18] J. Turian, L. Ratinov, and Y. Bengio, "Word representation: A simple and general method for semi-supervised learning," in *Proceedings of the ACL*, Uppsala, Sweden, July 2010.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *MIT Press Computational Models of Cognition And Perception Series*, pp. 318–362, 1986.

[20] Y. Dauphin, X. Glorot, and Y. Bengio, "Large-scale learning of embeddings with reconstruction sampling," in *Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Kyoto, Japan, 2012.

[21] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *ArXiv e-prints*, July 2012.

[22] Y. He and S. Young, "A data-driven spoken language understanding system," in *Proceedings of the IEEE ASRU Workshop*, U.S. Virgin Islands, December 2003, pp. 583–588.

[23] T. Kudo, "CRF++: Yet Another CRF toolkit," http://crfpp.googlecode.com/svn/trunk/doc/index.html, 2009.

[24] A. Deoras, R. Sarikaya, G. Tur, and D. Hakkani-Tür, "Joint Decoding for Speech Recognition and Semantic Tagging," in *Proc. of ISCA INTERSPEECH*, Portland, Oregon, US, 2012.

[25] A. Deoras, G. Tur, R. Sarikaya, and D. Hakkani-Tur, "Joint Discriminative Decoding of Words and Semantic Tags for Spoken Language Understanding," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 8, pp. 1612–1621, 2013.

[26] G. Tur, A. Deoras, and D. Hakkani-Tur, "Semantic Parsing Using Word Confusion Networks With Conditional Random Fields," in *Proc. of the INTERSPEECH*, 2013.