WEAKLY SUPERVISED NEURAL NETWORKS FOR PART-OF-SPEECH TAGGING

Sumit Chopra, Srinivas Bangalore

AT&T Labs-Research, 180 Park Ave, Florham Park, NJ
{schopra,srini}@research.att.com

ABSTRACT

We introduce a simple and novel method for the weakly supervised problem of Part-Of-Speech tagging with a dictionary. Our method involves training a connectionist network that simultaneously learns a distributed latent representation of the words, while maximizing the tagging accuracy. To compensate for the unavailability of true labels, we resort to training the model using a *Curriculum*: instead of random order, the model is trained using an ordered sequence of training samples, proceeding from "easier" to "harder" samples. On a standard test corpus, we show that without using any grammatical information, our model is able to outperform the standard EM algorithm in tagging accuracy, and its performance is comparable to other state-of-the-art models. We also show that curriculum learning for this setting significantly improves performance, both in terms of speed of convergence and in terms of generalization.

1. INTRODUCTION

In spoken language understanding (SLU) and natural language processing (NLP), machine learning techniques such as Hidden Markov Models (HMM), Maximum Entropy models, Support Vector Machines and Conditional Random Fields have become the norm to solve a range of disambiguation tasks such as part-of-speech (POS) tagging, named-entity tagging, supertagging, and parsing [1]. These techniques crucially rely on large amounts of data annotated with the appropriate task labels for training the models. However, data annotation is usually a tedious and an expensive process. In order to address this limitation, a number of alternate techniques have been proposed recently, both in the unsupervised and weakly supervised settings for many NLP tasks [2].

In particular for POS tagging task, which is considered an important processing step towards semantics for speech and language applications, Merialdo in [3] introduced a weakly supervised setting. It involves developing a tagger for a language using large amounts of un-annotated text, aided by a lexicon of words with all possible POS tags for each word in that language (a *dictionary*). In practice, such a lexicon can easily be extracted from a physical dictionary of a language without the need for text annotation. This problem has received a lot of attention recently and has been approached as a shared task for exploring novel models, with a view of potentially being useful in more realistic settings.

The most popular approach for solving this problem involves variants of HMM-based generative models trained using Expectation-Maximization (EM) and finding the parameters using Maximum Likelihood Estimation (MLE) [3]. Departing from MLE, Goldwater et al. [4] present a Bayesian approach with sparse priors, and significantly improve the tagging accuracy. In contrast, Goldberg et al. [5] show that when EM is provided good initial conditions, the EM-HMM combination can result in significant improvements in tagging accuracy. More recently, Ravi and Knight [6] obtain substantial improvements over previously published results, by using EM to find the parameters of a model while using an integer program to simultaneously constrain the model in order to find the smallest model that explains the data. A number of authors have departed from EM-HMM combination, including Smith and Eisner [7], who use a contrastive estimation algorithm to train a log-linear model, and Toutanova et al. [8] who propose a Bayesian LDA based model.

We propose a simple yet novel method using a neural network for the problem of Part-of-Speech tagging with a dictionary. Each word in the corpus is represented as a low dimensional latent distributed vector. A multi-layer neural network, and the latent representation of the words are simultaneously trained, while minimizing the tagging error. In order to accomodate the fact that it is a weakly supervised training task, we propose a novel method for training the model based on stochastic gradient descent, which exploits a regimented learning strategy called *Curriculum Learning* [9]. Finally, in order to handle multiple output tags associated with each training sample, we propose a modification to the negative log-likelihood loss function which is optimized during training. Once trained, tagging a new sentence is extremely fast using our model.

We evaluate the performance of our model on the same Penn Treebank test set used in the previous works. We show that our model outperforms the standard EM algorithm with only lexical information and without an explicit model of the syntactic constraints (typically encoded using n-grams of POS history). We also demonstrate the utility of structuring the training regimen according to curriculum learning in terms of convergence rates and accuracy on the task.

The outline of the paper is as follows. In Section 2, we present the model. Section 3 explains in detail the idea of curriculum learning. Experiments and results are presented in Section 4. Discussions and conclusions are given in Section 5.

2. THE NEURAL NETWORK ARCHITECTURE

The architecture of our tagging model is similar to the one proposed in [10] for tagging and [11, 12] for language modeling and other NLP tasks. It is composed of two components. See Figure 1.

The first component, called the *embedding layer*, is a learnable linear mapping which maps each word onto a low dimensional latent space. Let the set of N unique words in the vocabulary of the corpus be denoted by $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$. Let us assume that each word w_i is coded using a 1-of-n coding (i.e., the i-th word of the vocabulary is coded by setting the i-th element of the N dimensional vector to 1 and all the other elements to 0). The *embedding layer*, maps each word w_i to a continuous vector z_i which lies in a D dimensional space: $z_i = C \cdot w_i, \quad i \in \{1, \ldots, N\}$, where $C \in \Re^{D \times N}$ is a projection matrix. Continuous representation of any k-gram $(w_i w_{i+1} \ldots w_{i+k})$ is $z = (z_i z_{i+1} \ldots z_{i+k})$ – obtained by concatenating the representation of each of its words. The encoding of the dictionary is as follows. Let K be the total number of tags in the dictionary. Then the dictionary entry d_i, is a binary vector of size K with the j^{th} element set to 1 if the tag j is associated with word w_i.

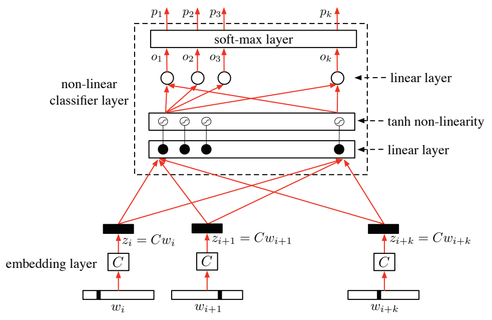

Fig. 1. *The neural network architecture, CNET, used for POS tagging and simultaneously learning word embedding.*

The second component, called the *classification layer* takes as input the representation of the k-gram, and produces as output a vector of tag probabilities which are used to make the decision. In our experiments this layer was composed of a single standard perceptron layer, followed by a fully connected linear layer with size equal to the number of classes. The output of this linear layer is passed through a soft-max non-linearity to generate conditional probabilities for each class (tag). In particular, let o_j denote the output of the j-th unit of the last linear layer. Then the output of the i-th unit of the soft-max non-linearity is given by $p_i = \frac{e^{o_i}}{\sum_j e^{o_j}}$. This classifier can be viewed as a parametric function $G_{\mathcal{M}}$ with parameters $\mathcal{M}$.

In summary, each word in a given sequence of k words is first mapped onto a D dimensional continuous space to create a feature vector of size kD. This vector is then fed as input to the classifier which generates a vector of tag probabilities conditioned on the input. The two sets of trainable parameters for the model are the mapping matrix C, and the set of parameters $\mathcal{M}$ associated with the non-linear classifier $G_{\mathcal{M}}$.

2.1. The Loss Function

Using the POS tags of each word in the dictionary, we construct a training set $\mathcal{S} = \{(x^i, y^i) : i = 1, \ldots, N\}$, consisting of the input-output pairs. Each input x^i is a sequence of k words obtained by sliding a window of size k over the entire corpus. In particular, for a sentence ($W = w_1 \ldots w_r$), each training example x^i consists of a target word w_j, with six words from its left ($c_l = w_{j-6} \ldots w_{j-1}$) and right context ($c_r = w_{j+1} \ldots w_{j+6}$), in addition to orthographic features (o_{w_j}) such as three character suffix and prefix, digit and upper case features. Thus the input x^i is the 4-tuple (c_l, w_j, c_r, o_{w_j}). The corresponding output y^i is set equal to d_j: the binary dictionary vector associated with the target word w_j. Note that depending on the word ambiguity one or more elements of the vector d_j could be set to 1. The parametric classifier $G_{\mathcal{M}}$ generates a vector of tag probabilities $p^i = G_{\mathcal{M}}(z^i)$. Training involves adjusting the parameters of the model (C and $\mathcal{M}$) so that these probabilities associated with the true tags is maximized for the target word. This is achieved by maximizing the likelihood of the training data.

In particular, let the symbol $\otimes$ denote the operation of component-wise multiplication of two vectors. For the sample i we define q^i as the index corresponding to the largest element of the vector $p^i \otimes y^i$. Then the likelihood associated with the i-th training sample under the model and the dictionary is given by

$$p^i = [G_{\mathcal{M}}(z^i)]_{q^i} = \frac{e^{o_{q^i}}}{\sum_j e^{o_j}}. \tag{1}$$

The likelihood of the entire training set is given by

$$L = \prod_{i=1}^{N} p^i = \prod_{i=1}^{N} [G_{\mathcal{M}}(z^i)]_{q^i} = \prod_{i=1}^{N} \frac{e^{o_{q^i}}}{\sum_j e^{o_j}}, \tag{2}$$

which is maximized by minimizing the negative log likelihood loss

$$\mathcal{L} = -\log L = \frac{1}{N} \sum_{i=1} \left(-o_{q^i} + log \sum_j e^{o_j} \right), \tag{3}$$

with respect to the parameters $\mathcal{M}$ and C, using stochastic gradient descent algorithm. In particular each epoch of training (a single pass through the entire training set) is composed of two phases. **In Phase 1**: the matrix C is kept fixed and the loss is minimized with respect to the parameters $\mathcal{M}$ of the classifier using stochastic gradient descent algorithm. **In Phase 2**: the parameters $\mathcal{M}$ are fixed and the loss is minimized with respect to the matrix C, again using stochastic gradient descent. However instead of randomly picking a training samples during each step of gradient descent, we follow the curriculum learning strategy described in the next section to select the examples based on the ambiguity of the target word.

3. CURRICULUM LEARNING

Concept learning in humans is based on a training regime that is not random but relies on a highly structured education system. Children are first taught simple concepts and as they achieve a level of mastery, move to learning more complex concepts. This learning strategy has been validated in the "less is more" hypothesis by psycholinguists investigating language acquisition [13] as well as by educational psychologists following the *scaffolding* theory [14].

The question of whether machine learning algorithms can benefit from such an organized training strategy, has been explored by a number of researchers at the intersection of machine learning and cognitive science. This question was recently revisited in [9], where they formally coined the term *Curriculum Learning*. The basic idea is to start the training process of a model using "easy" samples to learn simpler aspects of the task, and gradually move to "harder" samples to learn more complex aspects. They formalize the definition of "easy" and "hard" from the viewpoint of Continuation Methods [15]. According to their definition, a training regime can be classified as a *Curriculum* if over a period of time, the entropy of the example set used for training increases monotonically.

We structure the curriculum for our task of weakly supervised part-of-speech tagging as follows. We start the model training using stochastic gradient descent with samples that consist of words which have only one POS tag in the dictionary (i.e. no ambiguity). After training the model for some number of epochs on these training samples, we grow the learner's training set (and hence increase its entropy) by adding in examples of words which have two POS tags in the dictionary (i.e. an ambiguity of one) and train the model until convergence. We follow this training regime with ever increasing size of the training set until all the available samples are included in the learner's training set. Table 1 lists a subset of words grouped according to the stage of the curriculum in which they are considered during training based on the ambiguity in their tags.

4. EXPERIMENTS AND RESULTS

We use the Penn Treebank part-of-speech (POS) corpus [16] for the experiments in this paper. For evaluating the model, we used the test set commonly used on this task [6] which consists of 24,115 words. We used the same dictionary of word-to-tags association as in [6].

Table 1. Subset of target words according to the stage of the curriculum (C-stage) (up till stage 4) to which they belong.

C-stage	Word - (Tags)
1	bullishness - (NN), hard-hit - (JJ), rectified - (VBN), crunched - (VBD), Prince - (NNP), relish - (VBZ)
2	interbank - (JJ, NN), narrowed - (VBN, VBD), Indonesia - (NNP, NN), bites - (NNS, VBZ), processes - (NNS, VBZ), embarked - (VBN, VBD)
3	triple - (RB, VB, JJ), processed - (VBN, VBD, JJ) Silver - (NNP, JJ, NN), following - (VBG, JJ, NN) stop - (VBP, VB, NN), man - (UH, VB, NN)
4	protected - (VBN, VBD, VB JJ) British - (NNPS, NNS, NNP, JJ) right - (ADV, RB, JJ, NN)

We replaced the tag associated with each word in the labeled training set of around 1.1M words with the tags associated with that word in the dictionary. This resulted in an ambiguously tagged sentence corpus. The average POS ambiguity per word with this transformation is 2.3 tags/word. We note that the objective of the model training is to accurately resolve this ambiguity on both the training and test set.

The architecture of the neural network we used is shown in Figure 1. The dimension of the latent space onto which the words were embedded was set to 25. The *classifier layer* consisted of a fully connected hidden perceptron layer followed by a linear layer, followed by a soft-max non-linearity. The number of units in the perceptron layer was 400, and the linear layer had 47 units (equal to the number of classes). Since the performance of the system on the tuning set was fairly robust to the values of these hyper-parameters (the size of the embedding, the size of each hidden layer, and the learning rate of *embedding* and *classifier*), they were fixed for all the experiments. The learning rates of the *classifier* and *embedding* layers were set to 5×10^{-5} and 1×10^{-3} respectively, and were decreased by half after every 10 epochs. In the curriculum, the number of epochs after which samples with higher ambiguity were added was set to 5.

Due to the high non-linearity of the model different initializations could potentially lead to different local minima, resulting in vastly different task accuracy. To alleviate this problem, we trained several models independently, each initialized with a different set of random word embedding and classifier parameters, and the model which performed the best on the tuning set was selected as the final model for evaluation.

4.1. Performance on the Test Set

The tagging accuracy of our model, along with the accuracies of a number of other state-of-the-art models is summarized in Table 2. The POS tagging accuracy of our model with curriculum learning (CNet+CL) is superior to the bigram EM-HMM, trigram EM-HMM, and the Bayesian HMM models. The models CE+spl and IP+EM that outperform our model have incorporated certain global characteristics. In particular the CE+spl train a log-linear model with access to the full sequence of POS tags of a sentence, while the IP+EM model exploits the idea of minimizing the POS bigrams across the entire corpus. The results in the table are particularly noteworthy when one considers the fact that unlike other models, our proposed model is quite simple and does not exploit any grammar constrains. The simplicity of the model is that it is entirely lexically driven with no dependencies on the POS tag sequence. We attribute the performance of the model to the lexical representations that are learnt jointly with the discriminative non-linear architecture trained to optimize the POS tagging accuracy. We expect exploiting such global information within our model would likely improve the tagging accuracy even further. Lastly, it is interesting to note that the model without curriculum learning (CNet), where training samples are presented in no particular order is far inferior to all other models. This point is discussed further in Section 4.3.

Table 2. POS tagging accuracy of various models on the standard test set. EM - bigram: EM with a bi-gram tag model; EM - trigram: EM with 3-gram tag model; BHMM: Bayesian HMM with Sparse Priors [4]; CE+spl: Contrastive Estimation with Spelling Model [7]; InitEM-HMM: EM with good initialization [5]; IP+EM: EM with Integer Programming [6]; CNet: Connectionist Network; CL: Curriculum Learning; CD: Clean Dictionary; GC: Grammar Constraints.

Model	Percentage Accuracy using 47 tags
EM - bigram	81.7
EM - trigram	74.5
BHMM [4]	86.8
CE+spl [7]	88.6
InitEM-HMM [5]	91.4
IP+EM [6]	91.6
CNet	**67.72**
CNet + CL	**88.34**
CNet + CL + CD	**90.55**
CNet + CL + CD + GC	**90.80**

4.2. Performance on Pruned Dictionary

Upon analyzing the errors resulting from CNet+CL model, we found that the model was incorrectly tagging **all** the instances of the function words *is* and *an*. The word *is* was tagged as NN, and the word *an* was assigned the tag $\_COMMA\_$. Indeed, the words *is* and *an* are mistagged in their respective dictionary entries: (*is*: NN, VBZ), and (*an*: _COMMA_, DT). The reason behind the model wrongly tagging all the instances of these words can be attributed to its greedy nature. When the model sees the word *is* for the first time, the latent embedding associated with the word is random, and the choice between the tags NN and VBZ is made based on the conditional probabilities assigned to them, which at initialization are close to random. Thus with a 50% chance, the model might pick the wrong tag and back-propagate the gradients. When this happens the model will never be able to correct its mistake and pick the right tag ever again.

After removing such erroneous entries from the dictionary as suggested in [5], the test accuracy of our model improved to $\mathbf{90.55}\%$. Note that in Table 2 the performance of $\mathbf{91.4}\%$ reported in [5] uses such a pruned dictionary along with other language specific and linguistic constraints.

4.3. Impact of Curriculum Learning

In Figure 2, we plot the error rate on the test set for the two models trained with and without curriculum learning. For the model trained with curriculum learning there is a sharp drop in error rate at the end of every fifth epoch. This is when new samples with increasing ambiguity were added to the training set. However, note that the model trained with random selection of training examples, saturates at a very high error rate. The reason for this behavior is that once the model choses an incorrect POS tag for the target word and adjusts its parameters accordingly, this wrong choice is reinforced in subsequent iterations with no possibility of recovery. For a model trained using curriculum learning strategy, this possibility of error is

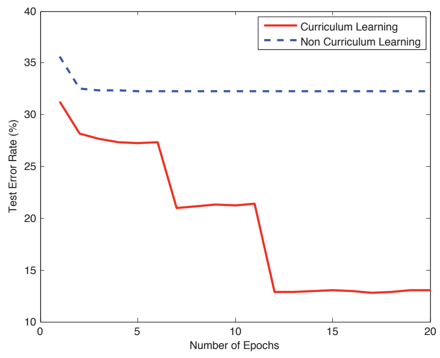

Fig. 2. *Plot showing the test error rates as a function of the number of training epochs when the machine is trained using curriculum learning (solid red line) and non-curriculum learning (dashed blue line).*

greatly reduced due to first training on lower ambiguity words before training on higher ambiguity words.

Lastly, once the representations of the words and the parameters of the classifier layer are learnt, the process of decoding a new sample involves a single forward propagation step. This step is composed of a few matrix vector multiplications and addition operations. At no point does our model make use of any tag sequence information and hence avoids any form of viterbi decoding. This makes the architecture very efficient during decoding. In particular, our model is capable of tagging 7945 words per second on a Xeon7550 machine.

5. DISCUSSIONS AND CONCLUSIONS

We presented a novel method for weakly supervised training of neural networks for part-of-speech tagging using a dictionary. Our method makes use of an organized training regime, called Curriculum Learning, to train the neural network while at the same time training a distributed embedding of the words. We have demonstrated that the performance of our model is similar to the state-of-the-art models which use global constraints of the task, while the model outperforms previously explored expectation-maximization techniques, using entirely lexical information.

As noted in the introduction, many speech and language tasks are solved using machine learning techniques which have benefitted from availability of annotated data. However, data annotation for supervised machine learning is tedious and expensive. As the complexity of tagging tasks increases, supervised annotation becomes harder to assure correct accuracy. The technique presented in this paper addresses this issue directly by not requiring a supervised data set.

The technique used in this paper might also be used for another disambiguation task called Supertagging. The idea of supertagging is to annotate words with syntactic structures that can easily lead to a syntactic parse and subsequently an interpretation of a sentence. The supertagging task naturally lends itself to curriculum learning. Learning can begin with supertags that have limited structure and subcategorization frames such as intransitive verbs. As the network learns, the more complicated supertags can be presented as training samples to the network.

While the performance of the model is based entirely on lexical information, it stands to question as to whether the performance can be further improved using global or long-distance constraints. As a preliminary experiment, we used the tag-tag constraints as a post-process and applying it to the output of the learnt model gave marginal improvement (last row Table 2). We believe that we could improve the results of the model by exploiting these constraints within the training loop in a more principled manner. Another extension would involve initializing the embedding more intelligently (for instance, the embedding obtained from training a neural network language model), as opposed to randomly, in order to avoid the optimization settling for a poor local minima.

6. REFERENCES

[1] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2008.

[2] "NAACL 2009 Workshop on Semi-Supervised Learning for NLP," 2009.

[3] B. Merialdo, "Tagging English Text with a Probabilistic Model," *Computational Linguistics*, 1994.

[4] S. Goldwater and T. Griffiths, "A Fully Bayesian Approach to Unsupervised Part-Of-Speech Tagging," *In Proc. of ACL*, pp. 744 – 751, June 2007.

[5] Y. Goldberg, M. Adler, and M. Elhadad, "EM Can Find Pretty Good HMM POS-Taggers (When Given a Good Start)," *In Proc. of ACL*, pp. 746 – 754, June 2008.

[6] S. Ravi and K. Knight, "Minimized Models for Unsupervised Part-Of-Speech Tagging," *In Proc. of ACL-IJCNLP*, pp. 504 – 512, August 2009.

[7] N. Smith and J. Eisner, "Contrastive Estimation: Training Log-Linear Models on Unlabeled Data," *In Proc. ACL*, pp. 354 – 362, 2005.

[8] K. Toutanova and M. Johnson, "A Bayesian LDA Based Model for Semi-Supervised Part-Of-Speech Tagging," *NIPS*, no. 20, 2008.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 41–48.

[10] S. Chopra and S. Bangalore, "Non-Linear Tagging Models with Localist and Distributed Word Representation," *In Proc. of 36th ICASSP*, May 2011.

[11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A Neural Probabilistic Language Model," *JMLR*, vol. 3, pp. 1137–1155, 2003.

[12] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," in *ICML08*. New York, NY, USA: ACM, 2008, pp. 160–167.

[13] B. Goldowsky and E. Newport, "Modeling the Effects of Processing Limitations on the Acquisition of Morphology: The Less is More Hypothesis," *The Proc. of the 11th West Coast Conference on Formal Linguistics*, 1993.

[14] L. Vygotsky, *The Collected Works of L.S. Vygotsky: The Fundamentals of Defectology*, 1988, vol. 2.

[15] E. L. Allgower and K. Georg, "Numerical Continuation Methods. An Introduction," *Springer-Verlag*, 1980.

[16] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19.2, pp. 313–330, 1993.