Tagging The Web: Building A Robust Web Tagger with Neural Network*

Ji Ma[†], **Yue Zhang**[‡] **and Jingbo Zhu**[†]
[†]Northeastern University, China
[‡]Singapore University of Technology and Design
majineu@gmail.com
yue_zhang@sutd.edu.sg
zhujingbo@mail.neu.edu.cn

Abstract

In this paper, we address the problem of web-domain POS tagging using a two-phase approach. The first phase learns representations that capture regularities underlying web text. The representation is integrated as features into a neural network that serves as a scorer for an easy-first POS tagger. Parameters of the neural network are trained using guided learning in the second phase. Experiment on the SANCL 2012 shared task show that our approach achieves 93.27% average tagging accuracy, which is the best accuracy reported so far on this data set, higher than those given by ensembled syntactic parsers.

1 Introduction

Analysing and extracting useful information from the web has become an increasingly important research direction for the NLP community, where many tasks require part-of-speech (POS) tagging as a fundamental preprocessing step. However, state-of-the-art POS taggers in the literature (Collins, 2002; Shen et al., 2007) are mainly optimized on the the Penn Treebank (PTB), and when shifted to web data, tagging accuracies drop significantly (Petrov and McDonald, 2012).

The problem we face here can be considered as a special case of *domain adaptation*, where we have access to labelled data on the source domain (PTB) and unlabelled data on the target domain (web data). Exploiting useful information from the web data can be the key to improving web domain tagging. Towards this end, we adopt the idea of learning representations which has been demonstrated useful in capturing hidden regularities underlying the raw input data (web text, in our case).

Our approach consists of two phrases. In the pre-training phase, we learn an encoder that converts the web text into an intermediate representation, which acts as useful features for prediction tasks. We integrate the learned encoder with a set of well-established features for POS tagging (Ratnaparkhi, 1996; Collins, 2002) in a single neural network, which is applied as a scorer to an easy-first POS tagger. We choose the easy-first tagging approach since it has been demonstrated to give higher accuracies than the standard left-to-right POS tagger (Shen et al., 2007; Ma et al., 2013).

In the fine-tuning phase, the parameters of the network are optimized on a set of labelled training data using guided learning. The learned model preserves the property of preferring to tag *easy* words first. To our knowledge, we are the first to investigate guided learning for neural networks.

The idea of learning representations from unlabelled data and then fine-tuning a model with such representations according to some supervised criterion has been studied before (Turian et al., 2010; Collobert et al., 2011; Glorot et al., 2011). While most previous work focus on in-domain sequential labelling or cross-domain classification tasks, we are the first to learn representations for web-domain structured prediction. Previous work treats the learned representations either as model parameters that are further optimized in supervised fine-tuning (Collobert et al., 2011) or as fixed features that are kept unchanged (Turian et al., 2010; Glorot et al., 2011). In this work, we investigate both strategies and give empirical comparisons in the cross-domain setting. Our results suggest that while both strategies improve in-domain tagging accuracies, keeping the learned representation unchanged consistently results in better cross-domain accuracies.

We conduct experiments on the official data set provided by the SANCL 2012 shared task (Petrov and McDonald, 2012). Our method achieves a

*This work was done while the first author was visiting SUTD.

93.27% average accuracy across the web-domain, which is the best result reported so far on this data set, higher than those given by ensembled syntactic parsers. Our code will be publicly available at https://github.com/majineu/TWeb and http://sourceforge.net/projects/zpar/TWeb.

2 Learning from Web Text

Unsupervised learning is often used for training encoders that convert the input data to abstract representations (i.e. encoding vectors). Such representations capture hidden properties of the input, and can be used as features for supervised tasks (Bengio, 2009; Ranzato et al., 2007). Among the many proposed encoders, we choose the restricted Boltzmann machine (RBM), which has been successfully used in many tasks (Lee et al., 2009b; Hinton et al., 2006). In this section, we give some background on RBMs and then show how they can be used to learn representations of the web text.

2.1 Restricted Boltzmann Machine

The RBM is a type of graphical model that contains two layers of binary stochastic units $\mathbf{v} \in \{0,1\}^V$ and $\mathbf{h} \in \{0,1\}^H$, corresponding to a set of visible and hidden variables, respectively. The RBM defines the joint probability distribution over $\mathbf{v}$ and $\mathbf{h}$ by an energy function

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{c}'\mathbf{h} - \mathbf{b}'\mathbf{v} - \mathbf{h}'\mathbf{W}\mathbf{v}, \tag{1}$$

which is factorized by a visible bias $\mathbf{b} \in \mathbb{R}^V$, a hidden bias $\mathbf{c} \in \mathbb{R}^H$ and a weight matrix $\mathbf{W} \in \mathbb{R}^{H\times V}$. The joint distribution $P(\mathbf{v},\mathbf{h})$ is given by

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(E(\mathbf{v},\mathbf{h})), \tag{2}$$

where Z is the partition function.

The affine form of E with respect to $\mathbf{v}$ and $\mathbf{h}$ implies that the visible variables are conditionally independent with each other given the hidden layer units, and vice versa. This yields the conditional distribution:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{V} P(v_j|\mathbf{h}) \quad P(\mathbf{h}|\mathbf{v}) = \prod_{i=1}^{H} P(h_i|\mathbf{v})$$

$$P(v_j = 1|\mathbf{h}) = \sigma(\mathbf{b}_j + W_{\cdot j}\mathbf{h}) \tag{3}$$

$$P(h_i = 1|\mathbf{v}) = \sigma(\mathbf{c}_j + W_{i\cdot}\mathbf{v}) \tag{4}$$

Here σ denotes the sigmoid function. Parameters of RBMs $\theta = \{\mathbf{b},\mathbf{c},\mathbf{W}\}$ can be trained efficiently using contrastive divergence learning (CD), see (Hinton, 2002) for detailed descriptions of CD.

2.2 Encoding Web Text with RBM

Most of the indicative features for POS disambiguation can be found from the words and word combinations within a local context (Ratnaparkhi, 1996; Collins, 2002). Inspired by this observation, we apply the RBM to learn feature representations from word n-grams. More specifically, given the i^{th} word w_i of a sentence, we apply RBMs to model the joint distribution of the n-gram $(w_{i-l}, \cdots, w_{i+r})$, where l and r denote the left and right window, respectively. Note that the visible units of RBMs are binary. While in our case, each visible variable corresponds to a word, which may take on tens-of-thousands of different values. Therefore, the RBM need to be re-factorized to make inference tractable.

We utilize the Word Representation RBM (WRRBM) factorization proposed by Dahl et al. (2012). The basic idea is to share word representations across different positions in the input n-gram while using position-dependent weights to distinguish between different word orders.

Let w_k be the k-th entry of lexicon L, and $\mathbf{w}_k$ be its *one-hot* representation (i.e., only the k-th component of $\mathbf{w}_k$ is 1, and all the others are 0). Let $\mathbf{v}^{(j)}$ represents the j-th visible variable of the WRRBM, which is a vector of length $|L|$. Then $\mathbf{v}^{(j)} = \mathbf{w}_k$ means that the j-th word in the n-gram is w_k. Let $\mathbf{D} \in \mathbb{R}^{D\times|L|}$ be a projection matrix, then $\mathbf{D}\mathbf{w}_k$ projects w_k into a D-dimensional real value vector (embedding). For each position j, there is a weight matrix $\mathbf{W}^{(j)} \in \mathbb{R}^{H\times D}$, which is used to model the interaction between the hidden layer and the word projection in position j. The visible biases are also shared across different positions ($b^{(j)} = b \;\forall j$) and the energy function is:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{c}'\mathbf{h} - \sum_{j=1}^{n}(\mathbf{b}'\mathbf{v}^{(j)} + \mathbf{h}'\mathbf{W}^{(j)}\mathbf{D}\mathbf{v}^{(j)}), \tag{5}$$

which yields the conditional distributions:

$$P(\mathbf{v}|\mathbf{h}) = \prod_{j=1}^{n} P(\mathbf{v}^{(j)}|\mathbf{h}) \quad P(\mathbf{h}|\mathbf{v}) = \prod_{i=1} P(h_i|\mathbf{v})$$

$$P(h_i = 1|\mathbf{v}) = \sigma(c_i + \sum_{j=1}^{n}\mathbf{W}_{i\cdot}^{(j)}\mathbf{D}\mathbf{v}^{(j)}) \tag{6}$$

$$P(\mathbf{v}^{(j)} = \mathbf{w}_k|\mathbf{h}) = \frac{1}{Z}\exp(\mathbf{b}'\mathbf{w}_k + \mathbf{h}'\mathbf{W}^{(j)}\mathbf{D}\mathbf{w}_k) \tag{7}$$

Again Z is the partition function.

The parameters $\{\mathbf{b}, \mathbf{c}, \mathbf{D}, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}\}$ can be trained using a Metropolis-Hastings-based CD variant and the learned word representations also capture certain syntactic information; see Dahl et al. (2012) for more details.

Note that one can stack standard RBMs on top of a WRRBM to construct a Deep Belief Network (DBN). By adopting greedy layer-wise training (Hinton et al., 2006; Bengio et al., 2007), DBNs are capable of modelling higher order non-linear relations between the input, and has been demonstrated to improve performance for many computer vision tasks (Hinton et al., 2006; Bengio et al., 2007; Lee et al., 2009a). However, in this work we do not observe further improvement by employing DBNs. This may partly be due to the fact that unlike computer vision tasks, the input structure of POS tagging or other sequential labelling tasks is relatively simple, and a single non-linear layer is enough to model the interactions within the input (Wang and Manning, 2013).

3 Neural Network for POS Disambiguation

We integrate the learned WRRBM into a neural network, which serves as a scorer for POS disambiguation. The main challenge to designing the neural network structure is: on the one hand, we hope that the model can take the advantage of information provided by the learned WRRBM, which reflects general properties of web texts, so that the model generalizes well in the web domain; on the other hand, we also hope to improve the model's discriminative power by utilizing well-established POS tagging features, such as those of Ratnaparkhi (1996).

Our approach is to leverage the two sources of information in one neural network by combining them though a shared output layer, as shown in Figure 1. Under the output layer, the network consists of two modules: *the web-feature module*, which incorporates knowledge from the pre-trained WRRBM, and *the sparse-feature module*, which makes use of other POS tagging features.

3.1 The Web-Feature Module

The web-feature module, shown in the lower left part of Figure 1, consists of a input layer and two hidden layers. The input for the this module is the word n-gram $(w_{i-l}, \ldots, w_{i+r})$, the form of which

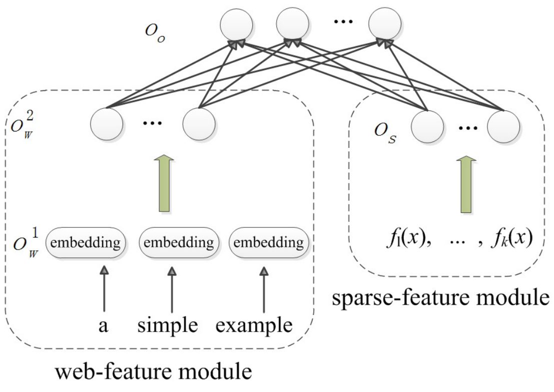

Figure 1: The proposed neural network. The web-feature module (lower left) and sparse-feature module (lower right) are combined by a shared output layer (upper).

is identical to the training data of the pre-trained WRRBM.

The first layer is a linear projection layer, where each word in the input is projected into a D-dimensional real value vector using the projection operation described in Section 2.2. The output of this layer $\mathbf{o}_w^1$ is the concatenation of the projections of $w_{i-l}, \ldots, w_{i+r}$:

$$\mathbf{o}_w^1 = \begin{pmatrix} \mathbf{M}_w^1 \mathbf{w}_{i-l} \\ \vdots \\ \mathbf{M}_w^1 \mathbf{w}_{i+r} \end{pmatrix} \tag{8}$$

Here $\mathbf{M}_w^1$ denotes the parameters of the first layer of the web-feature module, which is a $D \times |L|$ projection matrix.

The second layer is a sigmoid layer to model non-linear relations between the word projections:

$$\mathbf{o}_w^2 = \sigma(\mathbf{M}_w^2 \mathbf{o}_w^1 + \mathbf{b}_w^2) \tag{9}$$

Parameters of this layer include: a bias vector $\mathbf{b}_w^2 \in \mathbb{R}^H$ and a weight matrix $\mathbf{M}_w^2 \in \mathbb{R}^{H \times nD}$.

The web-feature module enables us to explore the learned WRRBM in various ways. First, it allows us to investigate knowledge from the WRRBM incrementally. We can choose to use only the *word* representations of the learned WRRBM. This can be achieved by initializing only the first layer of the web module with the projection matrix $\mathbf{D}$ of the learned WRRBM:

$$\mathbf{M}_w^1 \leftarrow \mathbf{D}. \tag{10}$$

Alternatively, we can choose to use the hidden states of the WRRBM, which can be treated as the representations of the input *n-gram*. This can be achieved by *also* initializing the parameters of the second layer of the web-feature module using the position-dependent weight matrix and hidden bias of the learned WRRBM:

$$\mathbf{b}_w^2 \leftarrow \mathbf{c} \tag{11}$$

$$\mathbf{M}_w^2 \leftarrow (\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(n)}) \tag{12}$$

Second, the web-feature module also allows us to make a comparison between whether or not to further adjust the pre-trained representation in the supervised fine-tuning phase, which corresponds to the supervised learning strategies of Turian et al. (2010) and Collobert et al. (2011), respectively. To our knowledge, no investigations have been presented in the literature on this issue.

3.2 The Sparse-Feature Module

The sparse-feature module, as shown in the lower right part of Figure 1, is designed to incorporate commonly-used tagging features. The input for this module is a vector of boolean values $\Phi(x) = (f_1(x), \ldots, f_k(x))$, where x denotes the partially tagged input sentence and $f_i(x)$ denotes a feature function, which returns 1 if the corresponding feature fires and 0 otherwise. The first layer of this module is a linear transformation layer, which converts the high dimensional sparse vector into a fixed-dimensional real value vector:

$$\mathbf{o}_s = \mathbf{M}_s\Phi(x) + \mathbf{b}_s \tag{13}$$

Depending on the specific task being considered, the output of this layer can be further fed to other non-linear layers, such as a sigmoid or hyperbolic tangent layer, to model more complex relations. For POS tagging, we found that a simple linear layer yields satisfactory accuracies.

The web-feature and sparse-feature modules are combined by a linear output layer, as shown in the upper part of Figure 1. The value of each unit in this layer denotes the score of the corresponding POS tag.

$$\mathbf{o}_o = \mathbf{M}_o \begin{pmatrix} \mathbf{o}_w \\ \mathbf{o}_s \end{pmatrix} + \mathbf{b}_o \tag{14}$$

In some circumstances, probability distribution over POS tags might be a more preferable form of output. Such distribution can be easily obtained by adding a soft-max layer on top of the output layer to perform a local normalization, as done by Collobert et al. (2011).

Algorithm 1 Easy-first POS tagging

```
Input:  $x$ a sentence of $m$ words $w_1, \ldots, w_m$
Output:  tag sequence of $x$
1: $\mathbf{U} \leftarrow [w_1, \ldots, w_m]$    // untagged words
2: while $\mathbf{U} \neq []$ do
3:     $(\hat{w}, \hat{t}) \leftarrow \arg\max_{(w,t)\in\mathbf{U}\times\mathbf{T}} S(w,t)$
4:     $\hat{w}.t \leftarrow \hat{t}$
5:     $\mathbf{U} \leftarrow \mathbf{U}/[\hat{w}]$    // remove $\hat{w}$ from $\mathbf{U}$
6: end while
7: return $[w_1.t, \ldots, w_m.t]$
```

4 Easy-first POS tagging with Neural Network

The neural network proposed in Section 3 is used for POS disambiguation by the easy-first POS tagger. Parameters of the network are trained using guided learning, where learning and search interact with each other.

4.1 Easy-first POS tagging

Pseudo-code of easy-first tagging is shown in Algorithm 1. Rather than tagging a sentence from left to right, easy-first tagging is based on a deterministic process, repeatedly selecting the *easiest* word to tag. Here "easiness" is evaluated based on a statistical model. At each step, the algorithm adopts a scorer, the neural network in our case, to assign a score to each possible word-tag pair (w, t), and then selects the highest score one $(\hat{w}, \hat{t})$ to tag (i.e., tag $\hat{w}$ with $\hat{t}$). The algorithm repeats until all words are tagged.

4.2 Training

The training algorithm repeats for several iterations over the training data, which is a set of sentences labelled with gold standard POS tags. In each iteration, the procedure shown in Algorithm 2 is applied to each sentence in the training set.

At each step during the processing of a training example, the algorithm calculates a *margin loss* based on two word-tag pairs $(\overline{w}, \overline{t})$ and $(\hat{w}, \hat{t})$ (line 4 $\sim$ line 6). $(\overline{w}, \overline{t})$ denotes the word-tag pair that has the highest model score among those that are *inconsistent* with the gold standard, while $(\hat{w}, \hat{t})$ denotes the one that has the highest model score among those that are *consistent* with the gold standard. If the loss is zero, the algorithm continues to process the next untagged word. Otherwise, parameters are updated using back-propagation.

The standard back-propagation algorithm

(Rumelhart et al., 1988) cannot be applied directly. This is because the standard loss is calculated based on a *unique* input vector. This condition does not hold in our case, because $\hat{w}$ and $\overline{w}$ may refer to different words, which means that the margin loss in line 6 of Algorithm 2 is calculated based on two different input vectors, denoted by $\langle\hat{w}\rangle$ and $\langle\overline{w}\rangle$, respectively.

We solve this problem by decomposing the margin loss in line 6 into two parts:

- $1 + nn(\overline{w}, \overline{t})$, which is associated with $\langle\overline{w}\rangle$;
- $-nn(\hat{w}, \hat{t})$, which is associated with $\langle\hat{w}\rangle$.

In this way, two separate back-propagation updates can be used to update the model's parameters (line 8 $\sim$ line 11). For the special case where $\hat{w}$ and $\overline{w}$ do refer to the same word w, it can be easily verified that the two separate back-propagation updates equal to the standard back-propagation with a loss $1 + nn(w, \overline{t}) - nn(w, \hat{t})$ on the input $\langle w\rangle$.

The algorithm proposed here belongs to a general framework named *guided learning*, where search and learning interact with each other. The algorithm learns not only a local classifier, but also the inference order. While previous work (Shen et al., 2007; Zhang and Clark, 2011; Goldberg and Elhadad, 2010) apply guided learning to train a linear classifier by using variants of the perceptron algorithm, we are the first to combine guided learning with a neural network, by using a margin loss and a modified back-propagation algorithm.

5 Experiments

5.1 Setup

Our experiments are conducted on the data set provided by the SANCL 2012 shared task, which aims at building a single robust syntactic analysis system across the web-domain. The data set consists of labelled data for both the source (Wall Street Journal portion of the Penn Treebank) and target (web) domains. The web domain data can be further classified into five sub-domains, including emails, weblogs, business reviews, news groups and Yahoo!Answers. While emails and weblogs are used as the development sets, reviews, news groups and Yahoo!Answers are used as the final test sets. Participants are not allowed to use web-domain labelled data for training. In addition to labelled data, a large amount of unlabelled data on the web domain is also provided. Statistics about labelled and unlabelled data are summarized in Table 1 and Table 2, respectively.

Algorithm 2 Training over one sentence

```
Input: $(x, t)$ a tagged sentence, neural net $nn$
Output: updated neural net $nn'$
 1: $\mathbf{U} \leftarrow [w_1, \ldots, w_m]$        // untagged words
 2: $\mathbf{R} \leftarrow [(w_1, t_1), \ldots, (w_m, t_m)]$ // reference
 3: while $\mathbf{U} \neq []$ do
 4:     $(\overline{w}, \overline{t}) \leftarrow \arg\max_{(w,t)\in(\mathbf{U}\times\mathbf{T}/\mathbf{R})} nn(w, t)$
 5:     $(\hat{w}, \hat{t}) \leftarrow \arg\max_{(w,t)\in\mathbf{R}} nn(w, t)$
 6:     $loss \leftarrow \max(0, 1 + nn(\overline{w}, \overline{t}) - nn(\hat{w}, \hat{t}))$
 7:     if $loss > 0$ then
 8:         $\hat{e} \leftarrow nn$.BackPropErr($\langle\hat{w}\rangle, -nn(\hat{w}, \hat{t})$)
 9:         $\overline{e} \leftarrow nn$.BackPropErr($\langle\overline{w}\rangle, 1+nn(\overline{w}, \overline{t})$)
10:         $nn$.Update($\langle\hat{w}\rangle, \hat{e}$)
11:         $nn$.Update($\langle\overline{w}\rangle, \overline{e}$)
12:     else
13:         $\mathbf{U} \leftarrow \mathbf{U}/\{\hat{w}\}$,    $\mathbf{R} \leftarrow \mathbf{R}/(\hat{w}, \hat{t})$
14:     end if
15: end while
16: return $nn$
```

The raw web domain data contains much noise, including spelling error, emotions and inconsistent capitalization. Following some participants (Le Roux et al., 2012), we conduct simple preprocessing steps to the input of the development and the test sets[1]

- Neutral quotes are transformed to opening or closing quotes.
- Tokens starting with "www.", "http." or ending with ".org", ".com" are converted to a "#URL" symbol
- Repeated punctuations such as "!!!!" are collapsed into one.
- Left brackets such as "<","{" and "[" are converted to "-LRB-". Similarly, right brackets are converted to "-RRB-"
- Upper cased words that contain more than 4 letters are lowercased.
- Consecutive occurrences of one or more digits within a word are replaced with "#DIG"

We apply the same preprocessing steps to all the unlabelled data. In addition, following Dahl et

[1]The preprocessing steps make use of no POS knowledge, and does not bring any unfair advantages to the participants.

	Training set	Dev set			Test set			
	WSJ-Train	Emails	Weblogs	WSJ-dev	Answers	Newsgroups	Reviews	WSJ-test
#Sen	30060	2,450	1,016	1,336	1,744	1,195	1,906	1,640
#Words	731,678	29,131	24,025	32,092	28,823	20,651	28,086	35,590
#Types	35,933	5,478	4,747	5,889	4,370	4,924	4,797	6,685

Table 1: Statistics of the labelled data. #Sen denotes number of sentences. #Words and #Types denote number of words and unique word types, respectively.

	Emails	Weblogs	Answers	Newsgroups	Reviews
#Sen	1,194,173	524,834	27,274	1,000,000	1,965,350
#Words	17,047,731	10,365,284	424,299	18,424,657	29,289,169
#Types	221,576	166,515	33,325	357,090	287,575

Table 2: Statistics of the raw unlabelled data.

features	templates
unigram	$H(w_i)$, $C(w_i)$, $L(w_i)$, $L(w_{i-1})$, $L(w_{i+1})$, t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}
bigram	$L(w_i) \odot L(w_{i-1})$, $L(w_i) \odot L(w_{i+1})$, $t_{i-2} \odot t_{i-1}$, $t_{i-1} \odot t_{i+1}$, $t_{i+1} \odot t_{i+2}$, $L(w_i) \odot t_{i-2}$, $L(w_i) \odot t_{i-1}$, $L(w_i) \odot t_{i+1}$, $L(w_i) \odot t_{i+2}$
trigram	$L(w_i) \odot t_{i-2} \odot t_{i-1}$, $L(w_i) \odot t_{i-1} \odot t_{i+1}$, $L(w_i) \odot t_{i+1} \odot t_{i+2}$

Table 3: Feature templates, where w_i denotes the current word. $H(w)$ and $C(w)$ indicates whether w contains hyphen and upper case letters, respectively. $L(w)$ denotes a lowercased w.

al. (2012) and Turian et al. (2010), we also lowercased all the unlabelled data and removed those sentences that contain less than 90% a-z letters.

The tagging performance is evaluated according to the official evaluation metrics of SANCL 2012. The tagging accuracy is defined as the percentage of words (punctuations included) that are correctly tagged. The averaged accuracies are calculated across the web domain data.

We trained the WRRBM on web-domain data of different sizes (number of sentences). The data sets are generated by first concatenating all the cleaned unlabelled data, then selecting sentences evenly across the concatenated file.

For each data set, we investigate an extensive set of combinations of hyper-parameters: the n-gram window (l, r) in $\{(1,1),(2,1),(1,2),(2,2)\}$; the hidden layer size in $\{200, 300, 400\}$; the learning rate in $\{0.1, 0.01, 0.001\}$. All these parameters are selected according to the averaged accuracy on the development set.

5.2 Baseline

We reimplemented the greedy easy-first POS tagger of Ma et al. (2013), which is used for all the experiments. While the tagger of Ma et al. (2013) utilizes a linear scorer, our tagger adopts the neural network as its scorer. The neural network of our baseline tagger only contains the sparse-feature module. We use this baseline to examine the performance of a tagger trained purely on the source domain. Feature templates are shown in Table 3, which are based on those of Ratnaparkhi (1996) and Shen et al. (2007).

Accuracies of the baseline tagger are shown in the upper part of Table 6. Compared with the performance of the official baseline (row 4 of Table 6), which is evaluated based on the output of BerkeleyParser (Petrov et al., 2006; Petrov and Klein, 2007), our baseline tagger achieves comparable accuracies on both the source and target domain data. With data preprocessing, the average accuracy boosts to about 92.02 on the test set of the target domain. This is consistent with previous work (Le Roux et al., 2011), which found that for noisy data such as web domain text, data cleaning is a effective and necessary step.

5.3 Exploring the Learned Knowledge

As mentioned in Section 3.1, the knowledge learned from the WRRBM can be investigated incrementally, using *word representation*, which corresponds to initializing only the projection layer of web-feature module with the projection matrix of the learned WRRBM, or *ngram-level representation*, which corresponds to initializing both the projection and sigmoid layers of the web-feature module by the learned WRRBM. In each case, there can be two different **training strategies** depending on whether the learned representations are further adjusted or kept unchanged during the fine-turning phrase. Experimental results under the 4 combined settings on the development sets are illustrated in Figure 2, 3 and 4, where the

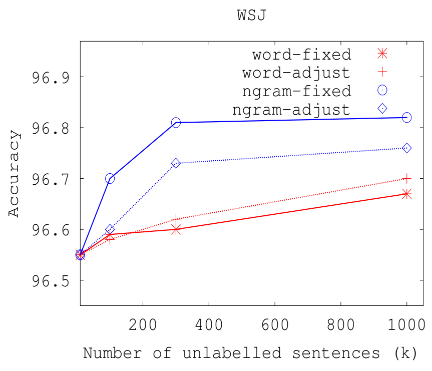

Figure 2: Tagging accuracies on the source-domain data. "word" and "ngram" denote using word representations and n-gram representations, respectively. "fixed" and "adjust" denote that the learned representation are kept unchanged or further adjusted in supervised learning, respectively.

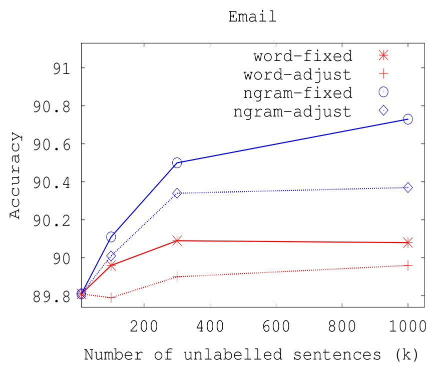

Figure 3: Accuracies on the email domain.

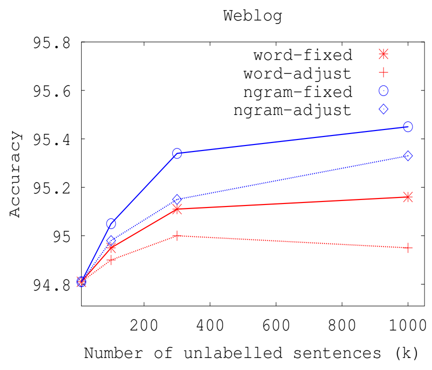

Figure 4: Accuracies on the weblog domain.

x-axis denotes the size of the training data and y-axis denotes tagging accuracy.

5.3.1 Effect of the Training Strategy

From Figure 2 we can see that when knowledge from the pre-trained WRRBM is incorporated, both the **training strategies** ("word-fixed" vs "word-adjusted", "ngram-fixed" vs "ngram-adjusted") improve accuracies on the source domain, which is consistent with previous findings (Turian et al., 2010; Collobert et al., 2011). In addition, adjusting the learned representation or keeping them fixed does not result in too much difference in tagging accuracies.

On the web-domain data, shown in Figure 3 and 4, we found that leaving the learned representation unchanged ("word-fixed", "ngram-fixed") yields consistently higher performance gains. This result is to some degree expected. Intuitively, unsupervised pre-training moves the parameters of the WRRBM towards the region where properties of the *web domain* data are properly modelled. However, since fine-tuning is conducted with respect to the *source domain*, adjusting the parameters of the pre-trained representation towards optimizing source domain tagging accuracies would disrupt its ability in modelling the web domain data. Therefore, a better idea is to keep the representation unchanged so that we can learn a function that maps the general web-text properties to its syntactic categories.

method	all	non-oov	oov
baseline	89.81	92.42	65.64
word-adjust	+0.09	−0.05	+1.38
word-fix	+0.11	+0.13	+1.73
ngram-adjust	+0.53	+0.52	+0.53
ngram-fix	+0.69	+0.60	+2.30

Table 4: Performance on the email domain.

5.3.2 Word and N-gram Representation

From Figures 2, 3 and 4, we can see that adopting the ngram-level representation consistently achieves better performance compared with using word representations only ("word-fixed" vs "ngram-fixed", "word-adjusted" vs "ngram-adjusted"). This result illustrates that the ngram-level knowledge captures more complex interactions of the web text, which cannot be recovered by using only word embeddings. Similar result was reported by Dahl et al. (2012), who found that using both the word embeddings and the hidden units of a tri-gram WRRBM as additional features for a CRF chunker yields larger improvements than using word embeddings only.

Finally, more detailed accuracies under the 4 settings on the email domain are shown in Table 4. We can see that the improvement of using word

		RBM-E	RBM-W	RBM-M
+acc%	Emails	+0.73	+0.37	+0.69
	Weblog	+0.31	+0.52	+0.54
cov%	Emails	95.24	92.79	93.88
	Weblog	90.21	97.74	94.77

Table 5: Effect of unlabelled data. "+acc" denotes improvement in tagging accuracy and "cov" denotes the lexicon coverages.

representations mainly comes from better accuracy of out-of-vocabulary (oov) words. By contrast, using n-gram representations improves the performance on both oov and non-oov.

5.4 Effect of Unlabelled Domain Data

In some circumstances, we may know beforehand that the target domain data belongs to a certain sub-domain, such as the email domain. In such cases, it might be desirable to train WRRBM using data only on that domain. We conduct experiments to test whether using the target domain data to train the WRRBM yields better performance compared with using mixed data from all sub-domains.

We trained 3 WRRBMs using the email domain data (RBM-E), weblog domain data (RBM-W) and mixed domain data (RBM-M), respectively, with each data set consisting of 300k sentences. Tagging performance and lexicon coverages of each data set on the development sets are shown in Table 5. We can see that using the target domain data achieves similar improvements compared with using the mixed data. However, for the email domain, RBM-W yields much smaller improvement compared with RBM-E, and vice versa. From the lexicon coverages, we can see that the sub-domains varies significantly. The results suggest that using mixed data can achieve almost as good performance as using the target sub-domain data, while using mixed data yields a much more robust tagger across all sub-domains.

5.5 Final Results

The best result achieved by using a 4-gram WRRBM, $(w_{i-2}, \ldots, w_{i+1})$, with 300 hidden units learned on 1,000k web domain sentences are shown in row 3 of Table 6. Performance of the top 2 systems of the SANCL 2012 task are also shown in Table 6. Our greedy tagger achieves 93.27% tagging accuracy, which is significantly better than the baseline's 92.02% accuracy ($p < 0.05$ by McNemar's test). Moreover, we achieve the highest tagging accuracy reported so far on this data set, surpassing those achieved using parser combinations based on self-training (Tang et al., 2012; Le Roux et al., 2012). In addition, different from Le Roux et al. (2012), we do not use any external resources in data cleaning.

6 Related Work

Learning representations has been intensively studied in computer vision tasks (Bengio et al., 2007; Lee et al., 2009a). In NLP, there is also much work along this line. In particular, Collobert et al. (2011) and Turian et al. (2010) learn word embeddings to improve the performance of in-domain POS tagging, named entity recognition, chunking and semantic role labelling. Yang et al. (2013) induce bi-lingual word embeddings for word alignment. Zheng et al. (2013) investigate Chinese character embeddings for joint word segmentation and POS tagging. While those approaches mainly explore token-level representations (word or character embeddings), using WRRBM is able to utilize both word and n-gram representations.

Titov (2011) and Glorot et al. (2011) propose to learn representations from the mixture of both source and target domain unlabelled data to improve cross-domain sentiment classification. Titov (2011) also propose a regularizer to constrain the inter-domain variability. In particular, their regularizer aims to minimize the Kullback-Leibler (KL) distance between the marginal distributions of the learned representations on the source and target domains.

Their work differs from ours in that their approaches learn representations from the feature vectors for sentiment classification, which might be of thousands of dimensions. Such high dimensional input gives rise to high computational cost and it is not clear whether those approaches can be applied to large scale unlabelled data, with hundreds of millions of training examples. Our method learns representations from only word n-grams with n ranging from 3 to 5, which can be easily applied to large scale-data. In addition, while Titov (2011) and Glorot et al. (2011) use the learned representation to improve cross-domain classification tasks, we are the first to apply it to cross-domain structured prediction.

Blitzer et al. (2006) propose to induce shared representations for domain adaptation, which is based on the alternating structure optimization

System	Answer	Newsgroup	Review	WSJ-t	Avg
baseline-raw	89.79	91.36	89.96	97.09	90.31
baseline-clean	91.35	92.06	92.92	97.09	92.02
best-clean	**92.50**	**93.83**	**93.64**	97.44	**93.27**
baseline-offical	90.20	91.24	89.33	97.08	90.26
Le Roux et al.(2011)	91.79	93.81	93.11	97.29	92.90
Tang et al. (2012)	91.76	92.91	91.94	**97.49**	92.20

Table 6: Main results. "baseline-raw" and "baseline-clean" denote performance of our baseline tagger on the raw and cleaned data, respectively. "best-clean" is best performance achieved using a 4-gram WRRBM. The lower part shows accuracies of the official baseline and that of the top 2 participants.

(ASO) method of Ando and Zhang (2005). The idea is to project the original feature representations into low dimensional representations, which yields a high-accuracy classifier on the target domain. The new representations are induced based on the auxiliary tasks defined on unlabelled data together with a dimensionality reduction technique. Such auxiliary tasks can be specific to the supervised task. As pointed out by Plank (2009), for many NLP tasks, defining the auxiliary tasks is a non-trivial engineering problem. Compared with Blitzer et al. (2006), the advantage of using RBMs is that it learns representations in a pure unsupervised manner, which is much simpler.

Regarding using neural networks for sequential labelling, our approach shares similarity with that of Collobert et al. (2011). In particular, we both use a non-linear layer to model complex relations underling word embeddings. However, our network differs from theirs in the following aspects. Collobert et al. (2011) model the dependency between neighbouring tags in a generative manner, by employing a transition score A_{ij}. Training the score involves a forward process of complexity $O(nT^2)$, where T denotes the number of tags. Our model captures such a dependency in a discriminative manner, by just adding tag-related features to the sparse-feature module. In addition, Collobert et al. (2011) train their network by maximizing the training set likelihood, while our approach is to minimize the margin loss using guided learning.

7 Conclusion

We built a web-domain POS tagger using a two-phase approach. We used a WRRBM to learn the representation of the web text and incorporate the representation in a neural network, which is trained using guided learning for easy-first POS tagging. Experiment showed that our approach achieved significant improvement in tagging the web domain text. In addition, we found that keeping the learned representations unchanged yields better performance compared with further optimizing them on the source domain data. We release our tools at https://github.com/majineu/TWeb.

For future work, we would like to investigate the two-phase approach to more challenging tasks, such as web domain syntactic parsing. We believe that high-accuracy web domain taggers and parsers would benefit a wide range of downstream tasks such as machine translation.

Acknowledgements

We would like to thank Hugo Larochelle for his advices on re-implementing WRRBM. We also thank Nan Yang, Shujie Liu and Tong Xiao for the fruitful discussions, and three anonymous reviewers for their insightful suggestions. This research was supported by the National Science Foundation of China (61272376; 61300097), the research grant T2MOE1301 from Singapore Ministry of Education (MOE) and the start-up grant SRG ISTD2012038 from SUTD.

References

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 1–9, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, Cambridge, MA.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. Also published as a book. Now Publishers, 2009.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

George E. Dahl, Ryan P. Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 679–686, New York, NY, USA, July. Omnipress.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc of ICML 2011*, pages 513–520.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August.

Joseph Le Roux, Jennifer Foster, Joachim Wagner, Rasul Samad Zadeh Kaljahi, and Anton Bryl. 2012. DCU-Paris13 Systems for the SANCL 2012 Shared Task. In *Proceedings of the NAACL 2012 First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, pages 1–4, Montréal, Canada, June.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009a. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc of ICML 2009*, pages 609–616.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Ng. 2009b. Unsupervised feature learning for audio classification using convolutional deep belief networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104.

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first pos tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria, August. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Barbara Plank. 2009. Structural correspondence learning for parse disambiguation. In Alex Lascarides, Claire Gardent, and Joakim Nivre, editors, *EACL (Student Research Workshop)*, pages 37–45. The Association for Computer Linguistics.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. 2007. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, Cambridge, MA.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 760–767, Prague, Czech Republic, June. Association for Computational Linguistics.

Buzhou Tang, Min Jiang, and Hua Xu. 2012. Varderlibt's systems for sancl2012 shared task. In *Proceedings of the NAACL 2012 First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, Montréal, Canada, June.

Ivan Titov. 2011. Domain adaptation by constraining inter-domain variability of latent feature representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 62–71, Portland, Oregon, USA, June. Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July. Association for Computational Linguistics.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–175, Sofia, Bulgaria, August. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, Seattle, Washington, USA, October. Association for Computational Linguistics.