Part-of-Speech Tagging for Twitter with Adversarial Neural Networks

Tao Gui, Qi Zhang*, Haoran Huang, Minlong Peng, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{tgui16,qz,huanghr15,mlpeng16,xjhuang}@fudan.edu.cn

Abstract

In this work, we study the problem of part-of-speech tagging for Tweets. In contrast to newswire articles, Tweets are usually informal and contain numerous out-of-vocabulary words. Moreover, there is a lack of large scale labeled datasets for this domain. To tackle these challenges, we propose a novel neural network to make use of out-of-domain labeled data, unlabeled in-domain data, and labeled in-domain data. Inspired by adversarial neural networks, the proposed method tries to learn common features through adversarial discriminator. In addition, we hypothesize that domain-specific features of target domain should be preserved in some degree. Hence, the proposed method adopts a sequence-to-sequence autoencoder to perform this task. Experimental results on three different datasets show that our method achieves better performance than state-of-the-art methods.

1 Introduction

During the last decade, social media have become extremely popular, on which billions of user-generated contents are posted every day. Many users have been writing about their thoughts and lives on the go. The massive unstructured data from social media provides valuable information for a variety of applications such as stock prediction (Bollen et al., 2011), public health analysis (Wilson and Brownstein, 2009; Paul and Dredze, 2011), real-time event detection (Sakaki et al., 2010), and so on. The quality of these applications is highly impacted by the performance of natural language processing tasks.

*Corresponding author.

@DORSEY33	lol	aw	i	thought	u	was	talkin	bout	another	time	.	nd	i	dnt
USR	UH	UH	PRP	VBD	PRP	VBD	VBG	IN	DT	NN	.	CC	PRP	VBP

see	u	either	!	
VB	PRP	RB	.	

Figure 1: An example of tagged Tweet, which contains nonstandard orthography, emoticon, and abbreviation. The tagset is defined similar as that of PTB (Marcus et al., 1993).

Part-of-speech (POS) tagging is one of the most important natural language processing tasks. It has also been widely used in the social media analysis systems (Ritter et al., 2012; Lamb et al., 2013; Kiritchenko et al., 2014). Most state-of-the-art POS tagging approaches are based on supervised methods. Hence, they usually require a large amount of annotated data to train models. Many datasets have been constructed for POS tagging task. Because newswire articles are carefully edited, benchmarks usually use them for annotation (Marcus et al., 1993). However, user-generated contents on social media are usually informal and contain many nonstandard lexical items. Moreover, the difference in domains between training data and evaluation data may heavily impact the performance of approaches based on supervised methods (Caruana and Niculescu-Mizil, 2006). Hence, most POS tagging methods cannot achieve the same performance as reported on newswire domain when applied on Twitter (Owoputi et al., 2013).

To perform the Twitter POS tagging task, some approaches have been proposed to perform the task. Gimpel et al. (2011) manually annotated 1,827 tweets and carefully studied various fea-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2411–2420
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tures. Ritter et al. (2011) also constructed a labeled dataset, which contained 787 tweets, to empirically evaluate the performance of supervised methods on Twitter. Owoputi et al. (2013) incorporated word clusters into the feature sets and further improved the performance. From these works, we can observe that the size of the training data was much smaller than the newswire domain's.

Besides the challenge of lack of training data, the frequent use of out-of-vocabulary words also makes this problem difficult to address. Social media users often use informal ways of expressing their ideas and often spell words phonetically (e.g., "2mor" for "tomorrow"). In addition, they also make extensive use of emoticons and abbreviations (e.g., ":-)" for smiling emotion and "LOL" for laughing out loud). Moreover, new symbols, abbreviations, and words are constantly being created. Figure 1 shows an example of tagged Tweet.

To tackle the challenges posed by the lack of training data and the out-of-vocabulary words, in this paper, we propose a novel recurrent neural network, which we call *Target Preserved Adversarial Neural Network* (TPANN) to perform the task. It can make use of a large quantity of annotated data from other resource-rich domains, unlabeled in-domain data, and a small amount of labeled in-domain data. All of these datasets can be easily obtained. To make use of unlabeled data, motivated by the work of Goodfellow et al. (2014) and Chen et al. (2016), the proposed method extends the bi-directional long short-term memory recurrent neural network (bi-LSTM) with an adversarial predictor. To overcome the defect that adversarial networks can merely learn the common features, we propose to use an autoencoder only acting on target dataset to preserve its own specific features. For tackling the out-of-vocabulary problem, the proposed method also incorporates a character level convolutional neutral network to leverage subword information.

The contributions of this work are as follows:

- We propose to incorporate large scale unlabeled in-domain data, out-of-domain labeled data, and in-domain labeled data for Twitter part-of-speech tagging task.
- We introduce a novel recurrent neural network, which can learn domain-invariant representations through in-domain and out-of-domain data and construct a cross domain POS tagger through the learned representations. The proposed method also tries to preserve the specific features of target domain.
- Experimental results demonstrate that the proposed method can lead to better performance in most of cases on three different datasets.

2 Approach

In this work, we propose a novel recurrent neural network, Target Preserved Adversarial Neural Network (TPANN), to learn common features between resource-rich domain and target domain, simultaneously to preserve target domain-specific features. It extends the bi-directional LSTM with adversarial network and autoencoder. The architecture of TPANN is illustrated in Figure 2. The model consists of four components: *Feature Extractor*, *POS Tagging Classifier*, *Domain Discriminator* and *Target Domain Autoencoder*. In the following sections, we will detail each part of the proposed architecture and training methods.

2.1 Feature Extractor

The feature extractor $\mathcal{F}$ adopts CNN to extract character embedding features, which can tackle the out-of-vocabulary word problem effectively. To incorporate word embedding features, we concatenate word embedding to character embedding as the input of bi-LSTM on the next layer. Utilizing a bi-LSTM to model sentences, $\mathcal{F}$ can extract sequential relations and context information.

We denote the input sentence as $\mathbf{x}$ and the i-th word as x_i. $x_i \in \mathcal{S}(x)$ and $x_i \in \mathcal{T}(x)$ represent input samples are from source domain and target domain, respectively. We denote the parameters of $\mathcal{F}$ as θ_f. Let $\mathcal{V}$ be the vocabulary of words, and $\mathcal{C}$ be the vocabulary of characters. d is the dimensionality of character embedding then $Q \in \mathbb{R}^{d \times |\mathcal{C}|}$ is the representation matrix of vocabulary. We assume that word $x_i \in \mathcal{V}$ is made up of a sequence of characters $\mathbf{C}^i = [c_1, c_2, \ldots, c_l]$, where l is the max length of word and every word will be padded to this length. Then $\mathbf{C}^i \in \mathbb{R}^{d \times l}$ would be the inputs of CNN.

We apply a narrow convolution between $\mathbf{C}^i$ and filter $\mathbf{H} \in \mathbb{R}^{d \times k}$, where k is the width of the filter.

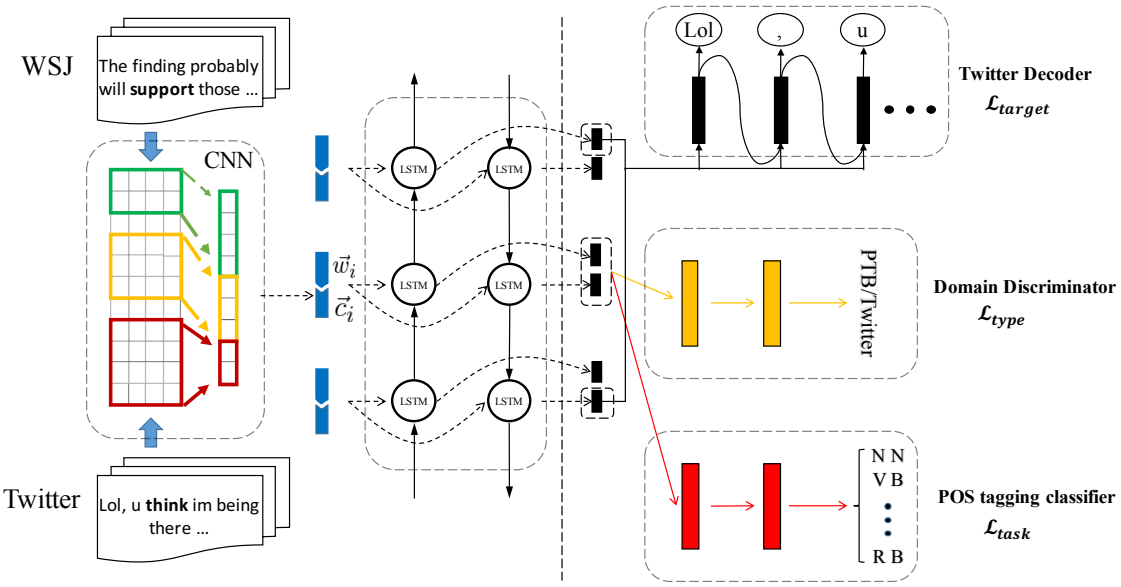

Figure 2: The general architecture of the proposed method.

After that we add a bias and apply nonlinearity to obtain a feature map $\mathbf{m}^i \in \mathbb{R}^{l-k+1}$. Specifically, the j-th element of $\mathbf{m}^i$ is given by:

$$\mathbf{i}^k[j] = \tanh(\langle \mathbf{C}^i[*, j : j+k-1], \mathbf{H} \rangle + b), \quad (1)$$

where $\mathbf{C}^k[*, j : j+k-1]$ is the j-to-$(j+k-1)$-th column of $\mathbf{C}^i$ and $\langle A, B \rangle = \mathrm{Tr}(AB^T)$ is the Frobenius inner product. We then apply a max-over-time pooling operation (Collobert et al., 2011) over the feature map. CNN uses multiple filters with varying widths to obtain the feature vector $\vec{c}_i$ for word x_i. Then, the character-level feature vector $\vec{c}_i$ is concatenated to the word embedding $\vec{w}_i$ to form the input of bi-LSTM on the next layer. The word embedding $\vec{w}$ is pretrained on 30 million tweets. Then, the hidden states $\mathbf{h}$ of bi-LSTM turn into the features that will be transfered to $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$, i.e. $\mathcal{F}(\mathbf{x}) = \mathbf{h}$.

2.2 POS Tagging Classifier and Domain Discriminator

POS tagging classifier $\mathcal{P}$ and domain discriminator $\mathcal{Q}$ take $\mathcal{F}(x)$ as input. They are standard feed-forward networks with a softmax layer for classification. $\mathcal{P}$ predicts POS tagging label to get classification capacity, and $\mathcal{Q}$ discriminates domain label to make $\mathcal{F}(x)$ domain-invariant.

The POS tagging classifier $\mathcal{P}$ maps the feature vector $\mathcal{F}(x_i)$ to its label. We denote the parameters of this mapping as θ_y. The POS tagging classifier is trained on N_s samples from the source domain with the cross entropy loss:

$$\mathcal{L}_{task} = -\sum_{i=1}^{N_s} y_i * \log \hat{y}_i, \quad (2)$$

where y_i is the one-hot vector of POS tagging label corresponding to $x_i \in \mathcal{S}(x)$, $\hat{y}_i$ is the output of top softmax layer: $\hat{y}_i = \mathcal{P}(\mathcal{F}(x_i))$. During the training time, The parameters θ_f and θ_y are optimized to minimize the classification loss $\mathcal{L}_{task}$. This ensures that $\mathcal{P}(\mathcal{F}(x_i))$ can make accurate prediction on the source domain.

Conversely, domain discriminator maps the same hidden states $\mathbf{h}$ to the domain labels with parameters θ_d. The domain discriminator aims to discriminate the domain label with following loss function:

$$\mathcal{L}_{type} = -\sum_{i=1}^{N_s+N_t} \{d_i \log \hat{d}_i + (1-d_i) \log(1-\hat{d}_i)\}, \quad (3)$$

where d_i is the ground truth domain label for sample i, $\hat{d}_i$ is the output of top layer: $\hat{d}_i = \mathcal{Q}(\mathcal{F}(x_i))$. N_t means N_t samples from the target domain. The domain discriminator is trained towards a saddle point of the loss function through minimizing the loss over θ_d while maximizing the loss over θ_f (Ganin et al., 2016). Optimizing θ_f ensures that the domain discriminator can't discriminate

the domain, i.e., the feature extractor finds the common features between the two domains.

2.3 Target Domain Autoencoder

Through training adversarial networks, we can obtain domain-invariant features $\mathbf{h}_{common}$, but it will weaken some domain-specific features which are useful for POS tagging classification. Merely obtaining domain invariant features would therefore limit the classification ability.

Our model tries to tackle this defect by introducing domain-specific autoencoder $\mathcal{R}$, which attempts to reconstruct target domain data. Inspired by (Sutskever et al., 2014) but different from (Dai and Le, 2015), we treat the feature extractor $\mathcal{F}$ as encoder. In addition, we combine the last hidden states of the forward LSTM and backward LSTM in $\mathcal{F}$ as the initial state $h_0(dec)$ of the decoder LSTM. Hence, we don't need to reverse the order of words of the input sentences and the model avoids the difficulty of "establish communication" between the input and the output (Sutskever et al., 2014).

Similar to (Zhang et al., 2016), we use $h_0(dec)$ and embedding vector of the previous word as the inputs of the decoder, but in a computationally more efficient manner by computing previous word representation. We assume that $(\hat{x}_1, \cdots, \hat{x}_T)$ is the output sequence. z_t is the t-th word representation: $z_t = MLP(h_t)$, and MLP is the multiple perceptron function. Hidden state $h_t = LSTM([h_0(dec) : z_{t-1}], h_{t-1})$, where $[\cdot : \cdot]$ is the concatenation operation. We estimate the conditional probability $p(\hat{x_1}, \cdots, \hat{x_T}|h_0(dec))$ as follows:

$$p(\hat{x_1}, \cdots, \hat{x_T}|h_0(dec)) = \prod_{t=1}^{T} p(\hat{x}_t|h_0(dec), z_1, \cdots, z_{t-1}), \quad (4)$$

where each $p(\hat{x}_t|h_0(dec), z_1, \cdots, z_{t-1})$ distribution is computed with softmax over all the words in the vacabulary.

Our aim is to minimize the following loss function with respect to parameters θ_r:

$$\mathcal{L}_{target} = -\sum_{i=1}^{N_t} x_i * \log \hat{x}_i, \quad (5)$$

where x_i is the one-hot vector of i-th word. This makes $h_0(dec)$ learn an undercomplete and most salient sentence representation of target domain data. When the adversarial networks try to optimize the hidden representation to common representation $\mathbf{h}_{common}$, The target domain autoencoder counteracts a tendency of the adversarial network to erase target domain features by optimizing the common representation to be informative on the target-domain data.

2.4 Training

Our model can be trained end-to-end with standard back-propagation, which we will detail in this section.

Our ultimate training goal is to minimize the total loss function with parameters $\{\theta_f, \theta_y, \theta_r, \theta_d\}$ as follows:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{task} + \beta\mathcal{L}_{target} + \gamma\mathcal{L}_{type}, \quad (6)$$

where α, β, γ are the weights to balance the effects of $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{Q}$.

For obtaining domain-invariant representation $\mathbf{h}_{common}$, inspired by (Ganin and Lempitsky, 2015), we introduce a special gradient reversal layer (GRL), which does nothing during forward propagation, but negates the gradients if it receives backward propagation, i.e. $g(\mathcal{F}(x)) = \mathcal{F}(x)$ but $\nabla g(\mathcal{F}(x)) = -\lambda\nabla\mathcal{F}(x)$. We insert the GRL between $\mathcal{F}$ and $\mathcal{Q}$, which can run standard Stochastic Gradient Descent with respect to θ_f and θ_d. The parameter $-\lambda$ drives the parameters θ_f not to amplify the dissimilarity of features when minimize $\mathcal{L}_{tpye}$. So by introducing a GRL, $\mathcal{F}$ can drive its parameters θ_f to extract hidden representations that help the POS tagging classification and hamper the domain discrimination.

In order to preserve target domain-specific features, we only optimize the autoencoder on target domain data for reconstruction tasks.

Through above procedures, the model can learn the common features between domains, simultaneously preserve target domain-specific features. Finally, we can update the parameters as follows:

$$\begin{aligned}
\theta_f &= \theta_f - \mu(\alpha\frac{\partial\mathcal{L}^i_{task}}{\partial\theta_f} + \beta\frac{\partial\mathcal{L}^i_{target}}{\partial\theta_f} - \gamma\cdot\lambda\frac{\partial\mathcal{L}^i_{type}}{\partial\theta_f}) \\
\theta_y &= \theta_y - \mu\cdot\alpha\frac{\partial\mathcal{L}^i_{task}}{\partial\theta_y} \\
\theta_r &= \theta_r - \mu\cdot\beta\frac{\partial\mathcal{L}^i_{target}}{\partial\theta_r} \\
\theta_d &= \theta_d - \mu\cdot\gamma\frac{\partial\mathcal{L}^i_{type}}{\partial\theta_d},
\end{aligned} \quad (7)$$

Dataset		# Tokens
WSJ		1,173,766
UNL		1,177,746
RIT-Twitter	RIT-Train	10,652
	RIT-Dev	2,242
	RIT-Test	2,291
NPSCHAT		44,997
ARK-Twitter	OCT27	26,594
	DAILY547	7,707

Table 1: The statistics of the datasets used in our experiments.

where μ is the learning rate. Because the size of the WSJ is more than 100 times that of the labeled Twitter dataset, if we directly train the model with the combined dataset, the final results are much worse than those using two training steps. So, we adopt adversarial training on WSJ and unlabeled Twitter dataset at the first step, then use a small number of in-domain labeled data to fine-tune the parameters with a low learning rate.

3 Experiments

In this section, we will detail the datasets used for experiments and experimental setup.

3.1 Datasets

The methods proposed in this work incorporate out-of-domain labeled data from resource-rich domains, large scale unlabeled in-domain data, and a small number of labeled in-domain data. The datasets used in this work are as follows:

Labeled out-of-domain data. We use a standard benchmark dataset for adversarial POS tagging, namely the Wall Street Journal (WSJ) data from the Penn TreeBank v3 (Marcus et al., 1993), sections 0-24 for the out-of-domain data.

Labeled in-domain data. For training and evaluating POS tagging approaches, we compare the proposed method with other approaches on three benchmarks: RIT-Twitter (Ritter et al., 2011), NPSCHAT (Forsyth, 2007), and ARK-Twitter (Gimpel et al., 2011).

Unlabeled in-domain data. For training the adversarial network, we need to use a dataset that has large scale unlabeled tweets. Hence, in this work, we construct large scale unlabeled data (UNL), from Twitter using its API.

The detailed data statistics of the datasets used in this work are listed in Table 1.

3.2 Experimental Setup

We select both state-of-the-art and classic methods for comparison, as follows:

- **Stanford POS Tagger**: Stanford POS Tagger is a widely used tool for newswire domains (Toutanova et al., 2003). In this work, we train it using two different sets, the WSJ (sections 0-18) and a WSJ, IRC, and Twitter mixed corpus. We use **Stanford-WSJ** and **Stanford-MIX** to represent them, respectively.

- **T-POS**: T-Pos (Ritter et al., 2011) adopts the Conditional Random Fields and clustering algorithm to perform the task. It was trained from a mixture of hand-annotated tweets and existing POS-labeled data.

- **GATE Tagger**: GATE tagger (Derczynski et al., 2013) is based on vote-constrained bootstrapping with unlabeled data. It combines cases where available taggers use different tagsets.

- **ARK Tagger**: ARK tagger (Owoputi et al., 2013) is a system that reports the best accuracy on the RIT dataset. It uses unsupervised word clustering and a variety of lexical features.

- **bi-LSTM**: Bidirectional Long Short-Term Memory (LSTM) networks have been widely used in a variety of sequence labeling tasks (Graves and Schmidhuber, 2005). In this work, we evaluate it at character level, word level, and combining them together. **bi-LSTM (word level)** uses one layer of bi-LSTM to extract word-level features and adopts a random initialization method to transform words to vectors. **bi-LSTM (character level)** represents a method that combines bi-LSTM and CNN-based character embedding, a similar approach with character-aware neural network described in (Kim et al., 2015) to handle the out-of-vocabulary words. **bi-LSTM (word level pretrain)** architecture is the same as that of bi-LSTM(word level) but adopts word2vec tool (Mikolov et al., 2013) to vectorize. **bi-LSTM (combine)** concatenates word to character features.

Methods	RIT-Test	RIT-Dev
Stanford-WSJ (Toutanova et al., 2003)	73.37%	83.29%
Stanford-MIX	83.14%	84.19%
T-POS (Ritter et al., 2011)	84.55%	84.83%
GATE Tagger (Derczynski et al., 2013)	88.69%	89.37%
ARK Tagger (Owoputi et al., 2013)	90.40%	-
bi-LSTM (word level)	75.91%	76.94%
bi-LSTM (word level pretrain)	85.99%	86.93%
bi-LSTM (character level)	82.85%	84.30%
bi-LSTM (combine)	89.48%	89.30%
bi-LSTM (combine + WSJ)	83.54%	83.64%
bi-LSTM (combine + WSJ + adversarial)	83.76%	84.45%
bi-LSTM (combine + WSJ + fine-tune)	89.87%	90.23%
bi-LSTM (combine + WSJ + adversarial + fine-tune)	90.60%	90.73%
TPANN (combine + WSJ + adversarial + fine-tune + autoencoder)	**90.92%**	**91.08%**

Table 2: Token level accuracies of different methods on RIT-Test and RIT-Dev. bi-LSTM(combine) refers to combining word level with character level. bi-LSTM(combine + WSJ) refers to the model trained on WSJ and tested on RIT. bi-LSTM(combine + WSJ + adversarial) refers to adversarial model trained on 1.1 million tokens of labeled WSJ data and the same scale of unlabeled Twitter data, then tested on RIT. Fine-tune means adding RIT-train data to fine-tune.

The hyper-parameters used for our model are as follows. AdaGrad optimizer trained with cross-entropy loss is used with 0.1 as the default learning rate. The dimensionality of word embedding is set to 200. The dimensionality for random initialized character embedding is set to 25. We adopt a bi-LSTM for encoding with each layer consisting of 250 hidden neurons. We set three layers of standard LSTM for decoding. Each LSTM layer consists of 500 hidden neurons. Adam optimizer trained with cross-entropy loss is used to fine-tune with 0.0001 as the default learning rate. Fine-tuning is run for 100 epochs using early stop.

4 Results and Discussion

In this section, we will report experimental results and a detailed analysis of the results for the three different datasets.

4.1 Evaluation on RIT-Twitter

The RIT-Twitter is split into training, development and evaluation sets (RIT-Train, RIT-Dev, RIT-Test). The splitting method is shown in (Derczynski et al., 2013), and the dataset statistics are listed in Table 1. Table 2 shows the results of our method and other approaches on the RIT-Twitter dataset. RIT-Twitter uses the PTB tagset with several Twitter-specific tags: retweets, @usernames, *hashtags*, and urls. Since words in these categories can be tagged almost perfectly using simple regular expressions, similar to (Owoputi et al., 2013), we use regular expressions to tags these words appropriately for all systems.

From the results of the Stanford-WSJ, we can observe that the newswire domain is different from Twitter. Although the token-level accuracy of the Stanford POS Tagger is higher than 97.0% on the PTB dataset, its performance on Twitter drops sharply to 73.37%. By incorporating some in-domain labeled data for training, the accuracy of Stanford POS Tagger can reach up to 83.14%. Taking a variety of linguistic features and many other resources into consideration, the T-POS, GATE tagger, and ARK tagger can achieve better performance.

The second part of Table 2 shows the results of the bi-LSTM based methods, which are trained on the RIT-Train dataset. According to the results of word level, we can see that word2vec can provide valuable information. The pre-trained word vectors in bi-LSTM(word level pretrain) give almost 10% higher accuracy than bi-LSTM(word level).

Comparing the character-level bi-LSTM with word-level bi-LSTM with random initialization, we can observe that the character-level method can achieve better performance than the word-level method. bi-LSTM(combine) combines word with character features, as described in Section 2.1,

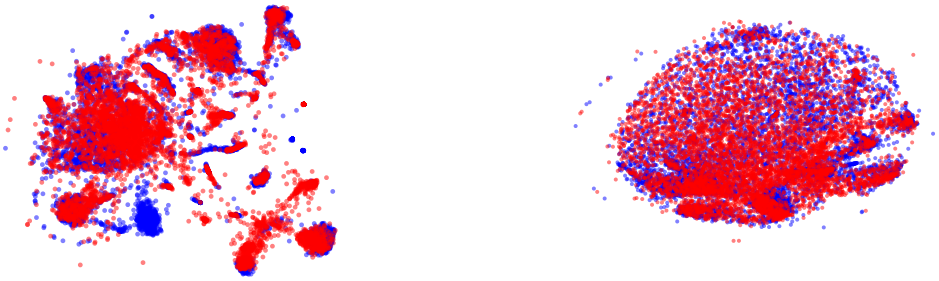

Figure 3: The visualization of bi-LSTM's outputs of the extracted features. The left figure shows the results when no adversary is performed. The right figure shows the results when the adversary procedure is incorporated into training. Blue points correspond to the source PTB domain examples, and red points correspond to the target Twitter domain.

which achieves the best results at 89.48% in the bi-LSTM based baseline systems and shows that the morphological features and pre-trained word vectors are both useful for POS tagging.

The third part of Table 2 shows the results of our methods incorporating out-of-domain labeled data, in-domain unlabeled data, and in-domain labeled data. Putting everything together, our model can achieve 90.92% on this dataset. Compared with the architecture without an adversarial model, our method is almost 1% better. It demonstrates that adversarial networks can significantly help with tasks of this nature. Through introducing the autoencoder in target domain, we can preserve domain-specific features for better performance. Compared with the ARK tagger, which achieves the previous best result on this dataset, our model is also 0.52% better, the error reduction rate is more than 5.5%.

To better understand why adversarial networks can help transfer domains from newswire to Twitter, in this work we also followed the method Ganin and Lempitsky (2015) used to visualize the outputs of LSTM with t-SNE (Van Der Maaten, 2013). Figure 3 shows the visualization results. From the figure, we can see that the adversary in our method makes the two distributions of features much more similar, which means that the outputs of bi-LSTM are domain-invariant. Hence, the PTB training data can provide much more help than directly combining PTB and RIT-Train together.

Methods	**Accuracy**
Forsyth (2007)	90.8%
ARK Tagger	93.4% ± 0.3%
TPANN	**94.1%**

Table 3: Tagging accuracies on NPSChat Corpus.

4.2 Evaluation on NPSChat

IRC, which contains Internet relay room messages from 2006, is a medium of online conversational text. Its content is very similar to tweets. We evaluate the proposed method on the NPSChat corpus (Forsyth, 2007), a PTB-part-of-speech annotated dataset of IRC.

We compared our method with a tagger in the same setup as experiments with (Forsyth, 2007). The training part contains 90% of the data. The testing part contains the other 10%. Table 3 shows the results of the ARK Tagger and our method. We used PTB, unlabeled Twitter, and the training part of NPSChat to train our model. From the results, we can see that our model achieved 94.1% accuracy. This is significantly better than the result Forsyth (2007) reported, which was 90.8%. They trained their tagger with a mix of several POS-annotated corpora (12K from Twitter, 40K from IRC, and 50K from PTB). Our method also outperforms state-of-the-art results 93.4%±0.3%, which was achieved by the ARK Tagger with various external corpus and features, e.g., Brown clustering, PTB, Freebase lists of celebrities, and video games.

Methods	Accuracy
Gimpel et al. (2011) version 0.2	90.8%
ARK Tagger	93.2%
TPANN	92.8%

Table 4: Tagging accuracies on DAILY547.

4.3 Evaluation on ARK-Twitter

ARK-Twitter data contains an entire dataset consisting of a number of tweets sampled from one particular day (October 27, 2010) described in (Gimpel et al., 2011). This part is used for training. They also created another dataset, which consists of 547 tweets, for evaluation (DAILY547). This dataset consists of one random English tweet from every day between January 1, 2011 and June 30, 2012. The distribution of training data may be slightly different from the testing data, for example a substantial fraction of the messages in the training data are about a basketball game. Since ARK-Twitter uses a different tagset with PTB, we manually construct a table to link tags for the two datasets.

Table 4 shows the results of different methods on this dataset. From the results, we can see that our method can achieve a better result than (Gimpel et al., 2011). However, the performance of our method is worse than the ARK Tagger. Through analyzing the errors, we find that 16.7% errors occurr between nouns and proper nouns. Since our method do not include any ontology or knowledge, proper nouns can not be easily detected. However, the ATK Tagge add a token-level name list feature. The name list is useful for proper nouns recognition, which fires on names from many sources, such as Freebase lists of celebrities, the Moby Words list of US Locations, proper names from Mark Kantrowitz's name corpus and so on. So, our model is also competitive when lacking of manual feature knowledge.

5 Related Work

Part-of-Speech tagging is an important pre-processing step and can provide valuable information for various natural language processing tasks. In recent years, deep learning algorithms have been successfully used for POS tagging. A number of approaches have been proposed and have achieved some progress. Santos and Guimaraes (2015) proposed using a character-based convolutional neural network to perform the POS tagging problem. Bi-LSTMs with word, character or unicode byte embedding were also introduced to achieve the POS tagging and named entity recognition tasks (Plank et al., 2016; Chiu and Nichols, 2015; Ma and Hovy, 2016). In this work, we study the problem from a domain adaption perspective. Inspired by these works, we also propose to use character-level methods to handle out-of-vocabulary words and bi-LSTMs to model the sequence relations.

Adversarial networks were successfully used for image generation (Goodfellow et al., 2014; Dosovitskiy et al., 2015; Denton et al., 2015), domain adaption (Tzeng et al., 2014; Ganin et al., 2016), and semi-supervised learning (Denton et al., 2016). The key idea of adversarial networks for domain adaption is to construct invariant features by optimizing the feature extractor as an adversary against the domain classifier (Zhang et al., 2017).

Sequence autoencoder reads the input sequence into a vector and then tries to reconstruct it. Dai and Le (2015) used the model on a number of different tasks and verified its validity. Li et al. (2015) introduced the model to hierarchically build an embedding for a paragraph, showing that the model was able to encode texts to preserve syntactic, semantic, and discourse coherence.

In this work, we incorporate adversarial networks with autoencoder to obtain domain-invariant features and keep domain-specific features. Our model is more suitable for target domain tasks.

6 Conclusion

In this work, we propose a novel adversarial neural network to address the POS tagging problem. Besides learning common representations between source domain and target domain, it can simultaneously preserve specific features of target domain. The proposed method leverages newswire resources and large scale in-domain unlabeled data to help POS tagging classification on Twitter, which has a few of labeled data. We evaluate the proposed method and several state-of-the-art methods on three different corpora. In most of the cases, the proposed method can achieve better performance than previous methods. Experimental results demonstrate that the proposed

method can make full use of these resources, which can be easily obtained.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088) and STCSM (No.16JC1420401).

References

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Emily Denton, Sam Gross, and Rob Fergus. 2016. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. 2015. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

Eric N Forsyth. 2007. Improving automated lexical and discourse analysis of online chat dialog.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, page 11801189.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Alex Lamb, Michael J Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *HLT-NAACL*, pages 789–795.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their

compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Michael J Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Laurens Van Der Maaten. 2013. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342*.

Kumanan Wilson and John S Brownstein. 2009. Early detection of disease outbreaks using the internet. *Canadian Medical Association Journal*, 180(8):829–831.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. *NIPS 2016 Workshop on Adversarial Training*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188*.