Dynamic Transfer Learning for Named Entity Recognition

Parminder Bhatia
Amazon, USA
parmib@amazon.com

Kristjan Arumae
Department of Computer Science
University of Central Florida, Orlando, USA
k.arumae@knights.ucf.edu

Busra Celikkaya
Amazon, USA
busrac@amazon.com

Abstract

State-of-the-art named entity recognition (NER) systems have been improving continuously using neural architectures over the past several years. However, many tasks including NER require large sets of annotated data to achieve such performance. In particular, we focus on NER from clinical notes, which is one of the most fundamental and critical problems for medical text analysis. Our work centers on effectively adapting these neural architectures towards low-resource settings using parameter transfer methods. We complement a standard hierarchical NER model with a general transfer learning framework consisting of parameter sharing between the source and target tasks, and showcase scores significantly above the baseline architecture. These sharing schemes require an exponential search over tied parameter sets to generate an optimal configuration. To mitigate the problem of exhaustively searching for model optimization, we propose the Dynamic Transfer Networks (DTN), a gated architecture which learns the appropriate parameter sharing scheme between source and target datasets. DTN achieves the improvements of the optimized transfer learning framework with just a single training setting, effectively removing the need for exponential search.

Introduction

Natural Language Processing (NLP) applications have been significantly enhanced through advances in neural architecture design. Tasks such as machine translation, summarization (See, Liu, and Manning 2017), language modeling (Mikolov et al. 2010), and information extraction have all achieved state of the art systems using deep neural networks, however with a caveat. These applications require large datasets to generalize well, and naturally sparse domains benefit less from such robust systems. One such domain is medical data. Specifically, clinical notes, the free text contents of electronic health records (EHR), have limited availability due to the delicate nature of their content. Privacy concerns prevent the public release of clinical notes, and furthermore de-identification, and annotation is a lengthy and costly process.

We are interested in Named Entity Recognition (NER) within low-resource areas such as medical domains (Jin et al. 2018). NER is a sequence labeling task similar to part of speech (POS) tagging, and text chunking. For medical data, NER is an important application as an information extraction tool for downstream tasks such as entity linking (Francis-Landau, Durrett, and Klein 2016) and relation extraction (Verga, Strubell, and McCallum 2018). Medical text has challenges that are unique to its domain as well. Clinicians will often use shorthand or abbreviations to produce patient release notes with irregular grammar. This gives the text a significantly less formal grammatical structure than standard NER datasets which often focus on newswire data (Ratinov and Roth 2009). There is also a high degree of variance across sub-domains, which can be attributed to the degree of specialty hospital departments have (e.g. cardiology vs. radiology). While certain medical jargon, and hospital procedure may be invariant of specialty; diseases, treatments, and medications will likely be correlated under these specific sub-domains. Building an NER system that can learn to generalize well across these is therefore quite difficult, and building individual systems for sub-domains is equally arduous due to the lack of data. Therefore, we turn towards transfer learning to diminish the effects of data accessibility, and to leverage overlapping representation across sub-domains.

Transfer learning (Yang, Salakhutdinov, and Cohen 2017) is a learning paradigm that seeks to enhance performance of a target task with knowledge from a source task. This can take several forms: as pretraining, where a model is first trained for a source task and then some or all weights are used for initialization of the target task; or in place of feature engineering using word embeddings (Bhatia, Guthrie, and Eisenstein 2016; Bojanowski et al. 2016), a popular approach for most NLP tasks. We look towards parameter sharing methods (Peng and Dredze 2017) to transfer overlapped representation from source to target task, when both are NER.

Parameter sharing schemes utilize tied weights between layers of a neural network across several tasks. Finding useful configurations of parameter sharing has been the focus of several recent papers (Peng and Dredze 2017; Yang, Salakhutdinov, and Cohen 2016; Fan et al. 2017; Guo, Pasunuru, and Bansal 2018b; Wang et al. 2018). As model depth increases the number of possible architectures grows exponentially, and it becomes difficult to exhaustively

search through all configurations to choose the best model. We show that these design choices are a learnable component of the model, and propose a new transfer learning architecture; a generalized neural model which dynamically updates independent and shared components achieving similar scores of models which have been fully tuned.

Our contributions are as follows:

- We propose the **Tunable Transfer Network** (TTN). A framework which unifies existing parameter sharing techniques into a single model. This network compartmentalizes all components of our baseline architecture. Furthermore, we fully explore three degrees of parameter sharing with this system: *hard*, *soft*, and *independent*. This architecture allows searching for the parameter sharing scheme that best suits the transfer learning setting.
- Addressing the large search space problem in TTN, we propose a **Dynamic Transfer Networks** (DTN), a gated architecture that learns the appropriate parameter sharing between source and target tasks across multiple sharing schemes. DTN mitigates the issue of exhaustive architecture exploration, while achieving similar performance of the optimized tunable network.
- We present a thorough empirical analysis of parameter sharing for low resource named entity recognition on medical data. We also demonstrate DTN's effectiveness on a non-medical dataset achieving best results in such settings.

We will first introduce related work as background for NER as well as transfer learning, followed by our proposed architecture, system setup, and dataset information. We conclude with our findings on low resource settings in both medical and non-medical domains.

Related Work

NER models achieved their recent success with neural architectures. In 2016 several works (Lample et al. 2016; Chiu and Nichols 2016; Yang, Salakhutdinov, and Cohen 2016) proposed hierarchical sequence to sequence deep learning frameworks. The models enjoyed RNN, or CNN encoders, but generally utilized conditional random fields (CRF) as decoders. Many subsequent works have focused on fine-tuning for speed or parameter size, while keeping this model design at a high level.

Transfer learning for both NER, and other NLP tasks has also been extensively studied. Here, we will look towards generic models, with more of a focus on those which targeted the medical domain. Sachan, Xie, and Xing (2017) leverage unsupervised pretraining in the form of forward and backward language modeling to initialize most of the parameters of an NER architecture. Their model was also evaluated on medical data and although the performance increased with pre-training, the evaluation showed low recall from unseen entities. Yang, Salakhutdinov, and Cohen (2016) were among the first to explore parameter sharing with the general neural NER architecture. The authors explored training for NER with other sequence tagging tasks, across multiple languages. Continuing their work they also correlated task similarity with the number of shared layers in a model (Yang, Salakhutdinov, and Cohen 2017). For example, tasks in the same language, and with similar labels would share a larger number of layers, whereas sequencing in English and Spanish, regardless of the output space may share only the input embeddings. The approach of sharing lower level layers was also used for semantic parsing (Fan et al. 2017), and co training language models (Liu et al. 2017). In the latter only a character level encoder was shared between tasks, and highway units control feature transfer to downstream components. We employ a similar technique by gating features from multiple inputs at the same layer. Shared label embedding layers have also shown favorable results (Augenstein, Ruder, and Søgaard 2018; Fan et al. 2017). For multiple tasks a single softmax is used with masking for non-task labels. The shared embeddings better promote label synergy.

Directly sharing parameters has been widely used, however transfer learning schemes have utilized a soft sharing paradigm as well, where model parameters or outputs are constrained to a similar space. Most similar to our work, Wang et al. (2018) use two constraints to promote shared representations of overlapping output distributions, as well as latent representations. This work minimizes parameter difference of the CRFs which is derived as the Kullback Leibler divergence upper bound minimization of the target task against the source across overlapping labels from both tasks. Additionally they constrain the model to produce similar latent representations for tokens with the same tag. This work is also applied towards NER across several medical sub-domains. Using soft sharing transfer learning for summarization Guo, Pasunuru, and Bansal (2018b) jointly train three generative models. Their work was also novel to not have the forked design, in that both the input and output layers were independent. The same authors used a similar architecture with more ablation on sharing for sentence simplification (Guo, Pasunuru, and Bansal 2018a).

The parameter sharing architectures discussed here all suffer from the need to exhaustively search for the best architecture. Our approach mitigates this procedure by allowing the model to learn which form of parameter sharing it should employ at various layers, and is able to do this during a single training session.

Our model also draws inspiration from pointer networks (Vinyals, Fortunato, and Jaitly 2015; See, Liu, and Manning 2017). Pointer networks have shown great performance in assisting generative models augment their output distribution with knowledge of the input sequence. Our work, however, uses this technique to transfer the signal across several parameter sharing components.

Models

We first present a standard neural framework for NER. We expand on that architecture by building the Tunable Transfer Network (TTN), to incorporate transfer learning options to each layer. Finally, we introduce the Dynamic Transfer Network (DTN), as a trainable transfer learning framework extending the TTN.

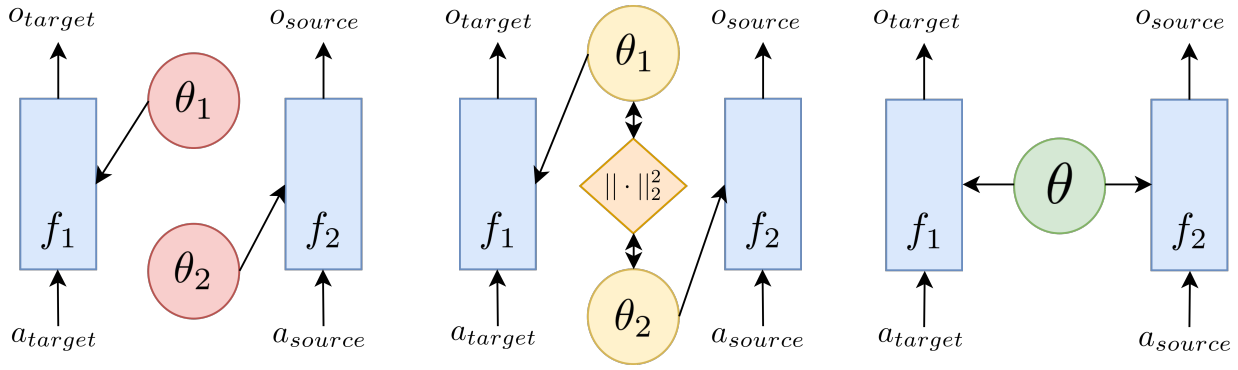

Figure 1: Tunable network architecture: This model is built with the option of independent (left), soft shared (center), or hard shared (right) weights for each of the main components. The components, presented as f_1 and f_2, refer to either one of the encoders or the decoder of the target and source task respectively. The blocks in the figure represent an arbitrary layer in the network, therefore a could refer to input embeddings, or latent representations of tokens, and o will similarly represent any component output. For both the independent and soft shared approaches θ_1, and θ_2 represent weights assigned to their respective functions, with the center configuration employing the soft sharing constraint $\mathcal{L}_{share}$ between them.

Named Entity Recognition Architecture

A sequence tagging problem such as NER can be formulated as maximizing the conditional probability distribution over tags $\mathbf{y}$ given an input sequence $\mathbf{x}$, and model parameters θ.

$$P(\mathbf{y}|\mathbf{x},\theta) = \prod_{t=1}^{T} P(y_t|x_t, y_{1:t-1}, \theta)$$

T is the length of the sequence, and $y_{1:t-1}$ are tags for the previous tokens. The architecture we use as a foundation is that of (Chiu and Nichols 2016; Lample et al. 2016; Yang, Salakhutdinov, and Cohen 2016), and while we provide a brief overview of this model we refer the reader to any of these works for architectural insights. The model consists of three main components: the (i) character and (ii) word encoders, and the (iii) decoder/tagger.

Encoders Given an input sequence $\mathbf{x} \in \mathbb{N}^T$ whose coordinates indicate the words in the input vocabulary, we first encode the character level representation for each word. For each x_t the corresponding sequence $\mathbf{c}^{(t)} \in \mathbb{R}^{L \times e_c}$ of character embeddings is fed into an encoder. Here L is the length of a given word and e_c is the size of the character embedding. The character encoder employs two Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) units which produce $\overrightarrow{h}^{(t)}_{1:l}$, and $\overleftarrow{h}^{(t)}_{1:l}$, the forward and backward hidden representations respectively, where l is the last timestep in both sequences. We concatenate the last timestep of each of these as the final encoded representation, $h_c^{(t)} = [\overrightarrow{h_l^{(t)}}||\overleftarrow{h_l^{(t)}}]$, of x_t at the character level.

The output of the character encoder is concatenated with a pre-trained word embedding (Pennington, Socher, and Manning 2014), $m_t = [h_c^{(t)}||\text{emb}_{word}(x_t)]$, which is used as the input to the word level encoder. Similar to the character encoder we use a bidirectional LSTM (BiLSTM) (Graves, Mohamed, and Hinton 2013) to encode the sequence at the word level. The word encoder does not lose resolution, meaning the output at each timestep is the concatenated output of both word LSTMs, $h_t = [\overrightarrow{h_t}||\overleftarrow{h_t}]$.

Decoder and Tagger Finally the concatenated output of the word encoder is used as input to the decoder, along with the label embedding of the previous timestep. During training we use teacher forcing (Williams and Zipser 1989) to provide the gold standard label as part of the input.

$$o_t = \text{LSTM}(o_{t-1}, [h_t||\hat{y}_{t-1}])$$
$$\hat{y}_t = \text{softmax}(\mathbf{W}o_t + b^s),$$

where $\mathbf{W} \in \mathbb{R}^{d \times n}$, d is the number of hidden units in the decoder LSTM, and n is the number of tags. The model is trained in an end to end fashion using a standard cross-entropy objective.

In most of the recent NER literature the focus has been on optimizing accuracy and speed by investigating different neural mechanisms for the three components (Yang, Salakhutdinov, and Cohen 2016). Both convolutional and recurrent networks have been explored for the encoders, with either conditional random fields (CRF), or single directional RNNs employed as the decoder/tagger. Since extensive work has been performed on this front we fix the design settings and focus only on transfer learning while using this common NER architecture. We also find that using an LSTM over a CRF gives us two benefits. We enjoy a more interpretable model, since we are able to view individual tag scores. This also provides a sense of uniformity to the architecture, having an RNN at every layer.

Tunable Transfer Network

The tunable transfer network extends to the three components from the previous sections. Here we focus on how best to benefit from transfer learning with respect to each layer. To reformulate the architecture from this perspective the model will always train on two tasks, henceforth labeled as *source* and *target*. Model parameters will be decomposed as:

$$\theta = \theta_{source} \cup \theta_{target} \cup \theta_{shared}$$

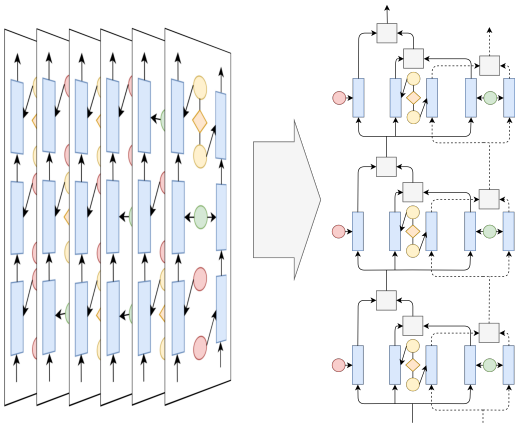

Figure 2: Tunable to Dynamic Transfer Network. The NER architecture, using all combinations of components from Figure 1, gives us 27 possible architectures (left). We show, through gating multiple sharing paradigms, that the DTN is able to learn how to produce a similar architecture (right).

Source and target parameters are updated by training examples from their respective datasets, while shared parameters receive updates from both tasks. Updates for parameters will depend on the *batch focus*, meaning for a given forward pass of the model a batch will contain data from either the source or target task. During training we shuffle the batches among tasks to allow the model to alternate randomly between them.

We now describe the parameter sharing architectures:

- Independent parameters, Figure 1 (left). Relative to the component, the network performs no transfer learning across the two parameter sets. For some layers the model performs best when no shared knowledge exists.
- Hard parameter sharing, Figure 1 (right). The parameters of both components reference the same set of weights, and each task in turn updates them.
- Soft parameter sharing, Figure 1 (center). Individual weights are given to both source and target components, however if this sharing paradigm is present in the model we add an additional segment to the objective:

$$\mathcal{L}_{share} = ||\theta_{source} - \theta_{target}||_2^2$$

Here, we minimize the l_2 distance between parameters as a form of regularization. Soft sharing loosely couples corresponding parameters to one another while allowing for more freedom than hard sharing, hence allowing different tasks to choose what sections of their parameters space to share.

The sharing paradigms from TTN intuitively represent the relatedness of the latent representation of the two tasks for a given component. Since these are tunable hyperparameters of the architecture, we optimize the model by finding the best configuration of sharing. Optimizing this involves training $O(M^N)$ unique models, where M is the number of sharing schemes, and N the number of tunable layers. Another problem with the current setup is that for some output distributions the target task may already exhibit high confidence in labels, and introducing a sharing scheme may in fact induce a bias towards the source task.

Dynamic Transfer Network

Searching across different model architectures motivates us to build a model similar to Figure 2 which is robust enough to overcome an exponential search of model architecture and achieve similar results compared to the tuned TTN model. As mentioned above, being able to tune model architecture is costly, and it is preferable to allow the system to learn how much of a representation to exploit from the source task vs. feedback from its own labels.

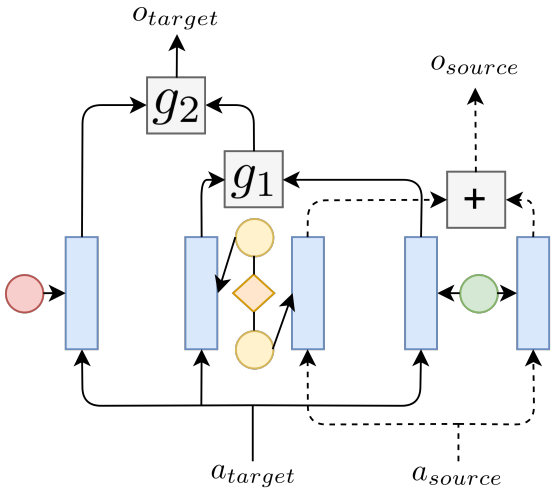

Figure 3: Dynamic Transfer Network: For each encoder and decoder layer of the baseline architecture, we use the DTN architecture. After passing through their respective RNNs (blue), the target (solid line) uses g_1 (Eq. 1) to gate the best representation of the sharing mechanisms. Similarly, g_2 (Eq. 3) gates the output of an independent RNN and g_1. The source task (dashed line) has no gating, and is added elmentwise to produce the its respective output.

Therefore we propose to use the Dynamic Transfer Network (DTN), where gating mechanisms similar to highway units(Srivastava, Greff, and Schmidhuber 2015), or pointer generators (See, Liu, and Manning 2017), control the signal strength from a shared and non-shared component of the network. We use these gates to choose the best representation between hard and soft sharing, and then between sharing and independent parameters. This multi-staged gating is similar to the layered pointers used by (McCann et al. 2018).

The architecture of DTN is illustrated in Figure 3. To begin, our source and target inputs both pass through their respective RNNs which employ soft (center), and hard (right) sharing, in parallel. The target and source RNNs take as input a_{target}, and a_{source} respectively. This produces two latent representations for both: $h_{\text{t-soft}}$, $h_{\text{s-soft}}$, $h_{\text{t-hard}}$, and $h_{\text{s-hard}}$, where t, and s denote target and source. We then determine which sharing mechanism was more useful for the target task using a gating function:

(A) Medication (i2b2) to TTP (i2b2) (10%)

Model	Precision	Recall	F_1
Baseline	55.20	48.25	51.47
Highest Performance TTN			
IIS	75.79	74.43	75.10
HIH	75.65	74.29	74.96
III	75.42	74.34	74.87
Lowest Performance TTN			
HSS	74.92	73.71	74.31
SSI	75.65	72.83	74.21
SSH	74.65	73.29	73.96
Avg.	75.47	73.69	74.57 ± 0.24
DTN	75.65	73.61	74.46
DTN (HS)	**75.83**	**74.09**	**74.95**

(B) Medication (i2b2) to Medication (Affiliate) (5%)

Model	Precision	Recall	F_1
Baseline	64.37	57.49	60.73
Highest Performance TTN			
HHI	77.06	64.38	70.03
SII	74.72	65.31	69.70
IIH	75.70	63.76	69.22
Lowest Performance TTN			
SSS	72.96	61.48	66.73
ISI	73.30	62.32	67.36
HSH	72.46	61.74	66.67
Avg.	73.27	62.61	67.76 ± 1.06
DTN	74.62	65.01	69.51
DTN (HS)	**72.83**	**66.93**	**69.95**

Table 1: Test set performance during low resource training. Table 1A displays results from i2b2, transferring from medication to TTP. Table 1B uses i2b2 medication as source and our affiliate medication data as a target. The baseline is the current state-of-the art optimized architecture for NER. For the tunable network (TTN) we indicate the sharing setting alongside each model (**S for soft shared, H for hard, and I for independent**). The ordering of the letters follows the that of the components (char enc., word enc., and decoder). For the sake of space we show only the three best, and three worst TTN results, along with the average across all 27 models. DTN, and DTN Hard-Soft (HS) are represented in the bottom two rows respectively.

$$g_1 = \sigma(\mathbf{Q}^\intercal h_{\text{t-soft}} + \mathbf{R}^\intercal h_{\text{t-hard}} + \mathbf{S}^\intercal a_{\text{target}} + b_{g_1}) \quad (1)$$

$$o_{\text{shared}} = (1 - g_1)h_{\text{t-hard}} + g_1 \cdot h_{\text{t-soft}} \quad (2)$$

We also used an independent (left) RNN, to produce a third latent representation for the target, h_{ind}. Our second gating function takes this, as well as the output of the first gated function as input.

$$g_2 = \sigma(\mathbf{T}^\intercal h_{\text{ind}} + \mathbf{U}^\intercal o_{\text{shared}} + \mathbf{V}^\intercal a_{\text{target}} + b_{g_2}) \quad (3)$$

$$o_{\text{target}} = (1 - g_2)h_{\text{ind}} + g_2 \cdot o_{\text{shared}} \quad (4)$$

The final result is a combined representation of the target task as input to subsequent layers. For both gates, σ is the sigmoid function, and $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$, $\mathbf{T}$, $\mathbf{U}$, $\mathbf{V}$, b_{g_1}, and b_{g_2} are trainable parameters. Since our task focuses on how best to adapt the layer towards the target task, the source hidden representations are simply added element-wise to produce:

$$o_{\text{source}} = h_{\text{s-hard}} + h_{\text{s-soft}}$$

The final loss for a network using DTN (Figure 2) has the weighted soft sharing regularization objective, along with the cross entropy loss of both tasks.

$$\mathcal{L}_{CE} = \mathcal{L}_{\text{target}} + \mathcal{L}_{\text{source}}$$

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{share}$$

TTN has a similar objective, however not all configurations will contain $\mathcal{L}_{share}$.

Inference Both the TTN, and DTN use only parameters for the target task during evaluation and inference. Meaning that we discard any portions of the model that only concern the source task during evaluation. E.g. in Figure 1 the system would discard f_2, and Θ_2.

Experimental Setup

Datasets

Our work utilizes four main corpora where we employ a tagging scheme that follows an inside, outside, begin, end and singleton (IOBES) format. We use the public datasets from the 2009 and 2010 i2b2 challenges for medication (Med) (Uzuner, Solti, and Cadag 2010), and "test, treatment, problem" (TTP) entity extraction.

	Med	TTP	Affiliate	CoNLL	Onto
Tags	25	13	37	17	73
Notes	252	426	1000	1393	3,637
Tokens	336K	416K	1.5M	301K	2M

Table 2: Overview of i2b2, affiliate, and newswire datasets.

The second dataset is obtained through an affiliate, and it is annotated similar to the i2b2 medication challenge. Both of the above datasets contain free-text release notes, which have been de-identified.

Additionally, we explore non-medical, newswire data: CoNLL 2003 English (Tjong Kim Sang and De Meulder 2003) and OntoNotes 5.0 English (Pradhan et al. 2013).

Model Settings

Word, character and tag embeddings are 100, 25, and 50 dimensions respectively. Word embeddings are initialized using GloVe (Pennington, Socher, and Manning 2014), while character and tag embeddings are learned from scratch. Character, and word encoders have 50, and 100 hidden

units respectively. Decoder LSTM has a hidden size of 50. Dropout is used after every LSTM, as well as for word embedding input. We use Adam (Kingma and Ba 2014) as an optimizer. Our model is built using MXNet (Chen et al. 2015). Hyperparameters are tuned using Bayesian Optimization (Snoek, Larochelle, and Adams 2012).

OntoNotes to CoNLL (10%)			
Highest Performance TTN			
Model	Precision	Recall	F_1
HIH	82.74	81.23	81.98
SSI	78.66	78.27	78.36
Avg.	80.66	79.47	80.06 $\pm$ 1.04
DTN (HS)	**82.45**	**81.78**	**82.12**

Table 3: CoNLL test set results using 10 % training data. These results follow the naming format described for Table 1. Here we display only the best and worst TTN model, along with the average of all 27 configurations.

DTN Hard-Soft We also evaluate a simplified version of the DTN presented in the previous section. This model, denoted as DTN (HS), learns the best transfer learning setting between soft coupling and hard sharing. This model retains the first gate (Eq. 1 and 2) from the architecture and uses o_{shared} as the final target signal for each component.

Experiments

Our models are trained until convergence, and we use the development set of the target task to evaluate performance for early stopping. We focus on transfer learning in three settings. The first setting uses only the i2b2 dataset, where the target task is TTP, and the source task is medication. The second set of experiments uses our affiliate medication data as a target, with i2b2 medication data as the source. The third task is non-medical, and uses CoNLL 2003 as the target, with OntoNotes 5.0 as the source. The first and third setting also allows for reproducible performance since the data is publicly available. We evaluate the performance of our models on 10% of the total target dataset for the first TL setting, and 5% for the second setting. For the non-medical setting, we used 10% of the total target dataset. The source dataset is not reduced in any of the experiments. Development and test set are also kept the original size. The baseline follows the construction of the architecture described in the first section of modeling.

Results

We analyze our results from multiple perspectives. We first demonstrate the effectiveness of parameter sharing for low resource settings by conducting experiments in the medical domain followed by results reflecting newswire corpora. We also examine model performance across various data percentages to showcase the uniform performance of DTN models. Furthermore, we explore the gating values across layers to investigate model behavior for the dynamic architecture which suggests why gating can imbibe the characteristics of the best model which varies depending upon the relatedness of the source and target tasks. We report precision, recall, and macro F_1 on the target data test set.

Transfer Learning Performance

The test set results on all medical data are reported in Table 1. For the tunable network, we show results for six models (three best, and three worst), as well as the average result across all 27 configurations (three components, and three sharing schemes following our). This encompasses the $O(M^N)$ models needed to exhaustively search through architectures for this system.

For the first setting (Table 1A), there is on average a 36.66% F_1 gain over the baseline model which indicates that the system greatly benefited from transfer learning. Similarly there was an 11.56% increase for TTN across the medication only tasks (Table 1B). Notably all settings of the tunable model yielded a large margin in performance over both baselines. More consequential, however, is the range of performance among the tunable models. We observed variance in the first task with the lowest F_1 score (soft-soft-hard) of 73.96 versus the highest 75.10 (indep-indep-soft). The second task had a gap of 3.27 F_1 points between high (70.03) and low (66.67) performers. These results validate the need to search for the best architecture for parameter sharing.

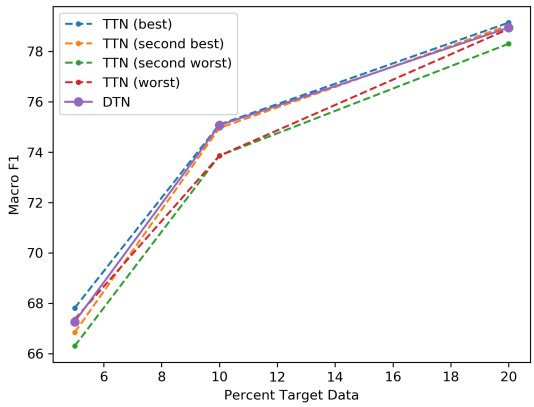

Figure 5: We select the two best, and worst performing models from the earlier medication to TTP experiments and compare their results against DTN across multiple low resource settings. We observe variance between the best and the worst models and effectiveness of DTN to generalize.

DTN In general, DTN performed very well, and more intriguing was the capability of DTN (HS), as it surpassed its more complex counterpart. For the first task, the dynamic model achieved a score of 74.46, and DTN (HS) outperformed all but the best two TTN, and scoring more than one standard deviation higher from the mean of the 27 TTN models. The second set of experiments is more indicative of

Input	Physical	examination	of	the	RLE	showed	mild	pain	in	right	hip
True Tag	B-test	I-test	I-test	I-test	E-test	O	B-problem	I-problem	I-problem	I-problem	E-problem
TTN (IIS)	B-test	E-test	O	B-test	E-test	O	B-problem	I-problem	I-problem	I-problem	E-problem
TTN (SIS)	B-test	E-test	O	O	O	O	B-problem	E-problem	O	B-problem	E-problem
TTN (IHI)	B-test	E-test	O	O	O	O	B-problem	E-problem	O	O	O
TTN (SHS)	B-test	I-test	I-test	I-test	E-test	O	B-problem	I-problem	I-problem	I-problem	E-problem
DTN	B-test	I-test	I-test	I-test	E-test	O	B-problem	I-problem	I-problem	I-problem	E-problem
Char enc											
Word enc											
Dec											

Figure 4: Through visualization we show that the DTN is able to adaptively learn the optimal transfer learning schema across a sequence. To demonstrate this we feed "Physical examination of the RLE showed mild pain in the right hip" to the DTN and the four models randomly selected from the top 20 percentile of the best performing TTN models. We show the ground truth and detected tags for each token, where O denotes any token which is not a named entity. The bottom three rows indicate the value of the gating signals for the three major components of the DTN. Since each component has two gates (Eq. 1 and 3), we use blue to illustrate g_1, and red for g_2. A darker color for g_1 indicates the model preferred soft sharing over hard. Appropriately a darker shade of red indicates that the model favored independent over any value of g_1. We show the model not only learns how to accurately predict the output tags, but also that it does not follow any specific sharing scheme.

the power of DTN. Here, we see a higher variance among TTN architectures, while DTN continues to stay competitive. DTN (HS) reaches more than two standard deviations above the average tunable model, and outperforms all but the single best. We hypothesize that the DTN (HS) performance can be at least partially attributed to fewer parameters, and that it was less likely to overfit on the small target datasets. Model performance is more commendable for the newswire data. In Table 3, we see a boost over the best performing TTN, with DTN (HS) placing two standard deviations above the TTN average.

Once again, our model is designed not to outperform all TTN but to reach a competitive performance with significantly reduced training time. We showed that the TTN configurations exhibit variability across training conditions, while DTN is able to match the top results from it. Figure 5 further illustrates this phenomenon.

We chose the best and worst TTN settings for a particular low resource (10%) setting (from i2b2 medication to TTP) and we see that the rankings are not as tightly coupled when we re-execute the experiments with more (20%) or fewer (5%) training samples. This illustrates that a particular sharing scheme varies with data, and cannot be relied upon across experiments, whereas using DTN a model remains competitive.

Gating

We take a closer look at output across a sequence in Figure 4. We compare the output of the DTN against better performing TTN models to show how the model adapts when others fail. The illustration is indicative that the model does not rely on a particular gating scheme consistently. Instead we observe the changes in gating across a sequence, where the model relies on multiple learning schemes for a given token.

We further analyzed the contributions of DTN between the different sharing schemes. Upon a closer inspection of the output layer gates as shown in Table 4, we observe significant variance among parameter sharing across different tag types. The parameter sharing for tags depends on the relatedness of the target and source tags. For example, *Form* is not present in the i2b2 (source) dataset. We discern that the decoder sharing scheme for the *Form* tag prefers hard sharing thus smaller value, as it can not leverage much information from the soft sharing scheme. Overall we observe interesting insights, where a parameter sharing scheme depends on the tag type as well as temporality thereby making RNN more robust to the sensitivity of the data.

Component	Char Enc	Word Enc	Decoder
Medication Name	0.64	0.91	0.77
Form	0.88	0.99	0.18
Dosage	0.69	0.99	0.26
Frequency	0.81	0.98	0.22
Overall	0.65	0.32	0.82

Table 4: Gate activations are averaged across all tokens from input, for experiment two. These results look at a gate choosing between hard and soft sharing (Eq. 1). A low value indicates the gate favored hard sharing, whereas a value closer to 1.0 favors soft sharing.

Conclusion

In this paper we have shown that tuning a transfer learning architecture in low resource settings will allow for a more efficient architecture. We further mitigated this exponential search process by introducing the dynamic transfer network to learn the best transfer learning settings for a given hierarchical architecture. We showed the generalization of this model across different named entity recognition datasets. For future work, we plan to explore our model on other sequential problems such as translation, summarization, chat bots as well as explore more advanced gating schemes.

References

[Augenstein, Ruder, and Søgaard 2018] Augenstein, I.; Ruder, S.; and Søgaard, A. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *arXiv preprint arXiv:1802.09913*.

[Bhatia, Guthrie, and Eisenstein 2016] Bhatia, P.; Guthrie, R.; and Eisenstein, J. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 490–500.

[Bojanowski et al. 2016] Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

[Chen et al. 2015] Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; and Zhang, Z. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

[Chiu and Nichols 2016] Chiu, J., and Nichols, E. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics* 4(1):357–370.

[Fan et al. 2017] Fan, X.; Monti, E.; Mathias, L.; and Dreyer, M. 2017. Transfer learning for neural semantic parsing. *arXiv preprint arXiv:1706.04326*.

[Francis-Landau, Durrett, and Klein 2016] Francis-Landau, M.; Durrett, G.; and Klein, D. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734*.

[Graves, Mohamed, and Hinton 2013] Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

[Guo, Pasunuru, and Bansal 2018a] Guo, H.; Pasunuru, R.; and Bansal, M. 2018a. Dynamic multi-level multi-task learning for sentence simplification. *arXiv preprint arXiv:1806.07304*.

[Guo, Pasunuru, and Bansal 2018b] Guo, H.; Pasunuru, R.; and Bansal, M. 2018b. Soft layer-specific multi-task summarization with entailment and question generation. *arXiv preprint arXiv:1805.11004*.

[Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

[Jin et al. 2018] Jin, M.; Bahadori, M. T.; Colak, A.; Bhatia, P.; Celikkaya, B.; Bhakta, R.; Senthivel, S.; Khalilia, M.; Navarro, D.; Zhang, B.; et al. 2018. Improving hospital mortality prediction with medical named entities and multimodal learning. *arXiv preprint arXiv:1811.12276*.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Lample et al. 2016] Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, 260–270.

[Liu et al. 2017] Liu, L.; Shang, J.; Xu, F.; Ren, X.; Gui, H.; Peng, J.; and Han, J. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

[McCann et al. 2018] McCann, B.; Keskar, N. S.; Xiong, C.; and Socher, R. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

[Mikolov et al. 2010] Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Peng and Dredze 2017] Peng, N., and Dredze, M. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 91–100.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

[Pradhan et al. 2013] Pradhan, S.; Moschitti, A.; Xue, N.; Ng, H. T.; Björkelund, A.; Uryupina, O.; Zhang, Y.; and Zhong, Z. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 143–152.

[Ratinov and Roth 2009] Ratinov, L., and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

[Sachan, Xie, and Xing 2017] Sachan, D. S.; Xie, P.; and Xing, E. P. 2017. Effective use of bidirectional language modeling for medical named entity recognition. *arXiv preprint arXiv:1711.07908*.

[See, Liu, and Manning 2017] See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1073–1083.

[Snoek, Larochelle, and Adams 2012] Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959.

[Srivastava, Greff, and Schmidhuber 2015] Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2377–2385. MIT Press.

[Tjong Kim Sang and De Meulder 2003] Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 142–147. Association for Computational Linguistics.

[Uzuner, Solti, and Cadag 2010] Uzuner, Ö.; Solti, I.; and Cadag, E. 2010. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association* 17(5):514–518.

[Verga, Strubell, and McCallum 2018] Verga, P.; Strubell, E.; and McCallum, A. 2018. Simultaneously self-attending to all mentions for full-abstract biological relation extraction. *arXiv preprint arXiv:1802.10569*.

[Vinyals, Fortunato, and Jaitly 2015] Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2692–2700.

[Wang et al. 2018] Wang, Z.; Qu, Y.; Chen, L.; Shen, J.; Zhang, W.; Zhang, S.; Gao, Y.; Gu, G.; Chen, K.; and Yu, Y. 2018. Label-aware double transfer learning for cross-specialty medical named entity recognition. *arXiv preprint arXiv:1804.09021*.

[Williams and Zipser 1989] Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

[Yang, Salakhutdinov, and Cohen 2016] Yang, Z.; Salakhutdinov, R.; and Cohen, W. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

[Yang, Salakhutdinov, and Cohen 2017] Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.