Gated Task Interaction Framework for Multi-task Sequence Tagging

Isaac K. E. Ampomah
School of Computing, Engineering and Built Environment
Ulster University
Belfast, UK
ampomah-i@ulster.ac.uk

Sally McClean
School of Computing, Engineering and Built Environment
Ulster University
Belfast, UK
si.mcclean@ulster.ac.uk

Zhiwei Lin
School of Computing, Engineering and Built Environment
Ulster University
Belfast, UK
z.lin@ulster.ac.uk

Glenn Hawe
School of Computing, Engineering and Built Environment
Ulster University
Belfast, UK
gi.hawe@ulster.ac.uk

***Abstract*—Recent studies have shown that neural models can achieve high performance on several sequence labelling/tagging problems without the explicit use of linguistic features such as part-of-speech (POS) tags. These models are trained only using the character-level and the word embedding vectors as inputs. Others have shown that linguistic features can improve the performance of neural models on tasks such as chunking and named entity recognition (NER). However, the change in performance depends on the degree of semantic relatedness between the linguistic features and the target task; in some instances, linguistic features can have a negative impact on performance. This paper presents an approach to jointly learn these linguistic features along with the target sequence labelling tasks with a new multi-task learning (MTL) framework called Gated Tasks Interaction (GTI) network for solving multiple sequence tagging tasks. The GTI network exploits the relations between the multiple tasks via neural gate modules. These gate modules control the flow of information between the different tasks. Experiments on benchmark datasets for chunking and NER show that our framework outperforms other competitive baselines trained with and without external training resources.**

I. Introduction

Neural approaches to sequence tagging problems such as *Named Entity Recognition* (NER), *Chunking* and *Part-of-Speech* (POS) tagging have recently achieved remarkable performance [1]. The strength of neural models over traditional machine learning approaches such as Conditional Random Fields (CRF) [2] and Hidden Markov Models (HMMs) [3] lies in their ability to automatically extract features from the input data. Recent studies [4], [5] successfully trained a bidirectional long short-term memory (Bi-LSTM) with a CRF classifier based on the input word-based features (mainly vector representation of words obtained from pre-trained word embedding vectors such as Glove [6]). Character-level representations are also exploited to extend the word-based features as they enhance the performance of models with morphological information [7]–[10].

Aside from the word and the character-level representation, recent works [11]–[13] suggest that higher performance can be obtained by leveraging additional information from linguistic features such as chunking and POS tags. The POS and chunking features along with chemical were used as additional features to train a neural model for chemical NER [11]. They argued that the choice of additional linguistic information affects the performance on the NER task. Other recent works [14]–[16] also suggest directly modeling these linguistic features along with the target task in a multi-task learning fashion can further enhance the performance. Hierarchical neural architectures for MTL were proposed by [14], [15]. The multiple tasks were arranged in terms of their level of complexity to reflect the linguistic hierarchies of the tasks under consideration. Finally, combining neural attention mechanism across dependency parsing, POS and predicate detection tasks, [16] achieved a state-of-the-art performance on Semantic Role Labeling (SRL) task. Though how to efficiently apply MTL to the task of sequence labelling has been extensively studied [17]–[19], the performance of MTL depends on the relatedness of the tasks under consideration. Further research is required to investigate the design constraints and challenges to exploiting the relatedness between the main and auxiliary tasks under MTL efficiently and effectively.

To improve the performance of the multi-task semantic sequence tagging, this paper proposes a novel neural architecture, called Gated Task Interaction (GTI) network. The GTI network takes a sequence of words as input and learns to generate the target sequence labels (e.g. NER) along with the linguistic features (such as the chunking and the POS tags). The proposed GTI neural framework aims to capture the mutual dependencies between multiple sequence labeling problems. That is, given a main(target) task and its associated auxiliary tasks, the GTI model seeks to efficiently exploit the semantic relations between the given tasks so as to improve the model's performance in terms of learning the multiple

TABLE I: Notation Table.

Notation	Description
X	model's input
x_i	*i*-th word/token
$\mathbf{x}$	embedding of the model's input sequence
$\mathbf{x_i}$	embedding of the token/word x_i
$\mathbf{w}(\cdot)$	word embedding look up table
$\mathbf{F}(\cdot)$	'token/word format' look up table
$\mathrm{charCNN}(\cdot)$	character encoding sub-network
$\mathbf{w}(x_i)$	word embedding of token x_i
$\mathbf{F}(x_i)$	'token/word format' embedding for the i-th token
$\mathrm{charCNN}(x_i)$	character-level embedding of the token x_i
$\mathbf{S_m}$	task specific semantic representation for the main task
$\mathbf{S^k_{aux}}$	task specific semantic representation for auxiliary task *k*
Y^k	output label sequence under auxiliary task *k*
Y^m	output label sequence under the main task
$\mathbf{L^k}$	label embedding vector generated from Y^k
$\mathbf{g_k}$	gated auxiliary feature vector from the auxiliary task *k*
$\mathbf{\hat{g}_k}$	gate control vector for auxiliary task *k*
$\mathbf{z_f}$	auxiliary feature vector generated from all the auxiliary task
AuxTask$^{\mathrm{k}}$	sub-network for auxiliary task *k*
MT	the main task's sub-network

objectives. To achieve high performance on the tasks, the GTI framework employs a gating mechanism to control the flow of information between the main task and its associated auxiliary tasks under consideration. The gating mechanism also serves as a regularization technique for improving the model's performance on the associated tasks [20], [21]. In this work, the auxiliary tasks and the main sub-networks are trained jointly end-to-end on the same dataset, eliminating the need for external datasets.

The GTI framework is evaluated with two sequence labelling tasks: NER tagging on the CoNLL-2003 shared task [22], and chunking on CoNLL-2000 shared task [23]. The contributions of this work are:

- proposing a novel neural architecture for multi-task linguistic sequence tagging/labelling.
- evaluating the GTI framework empirically on benchmark datasets for sequence tagging problems (NER and chunking).
- determining the impact of the choice of auxiliary tasks and hyperparameters (mainly the LSTM hidden state size) on the performance of our proposed approach.

The remainder of the paper is organized as follows. Section II introduces our MTL sequence labeling framework. The experiments conducted are presented in Section III, and the results are compared and discussed in Section IV. Section V briefly discusses the related works. The conclusion is presented in Section VI.

II. GTI Framework

In this section, we provide a brief introduction to the components of our model and present the GTI framework for the multi-task sequence labelling problem.

A. Gated Task Interaction (GTI) Network

Fig. 1a illustrates the basic neural network architecture of the proposed GTI framework and our architecture with two auxiliary tasks (Y^1 and Y^2) is shown in Fig. 1b. Given an input sequence $X = [x_1, x_2, \cdots, x_n]$ containing *n* tokens, a main/target task $Y^m = [y^m_1, y^m_2, \cdots, y^m_n]$ and a set of K auxiliary tasks $\{Y^k : 1 \leq k \leq K\}$, where $Y^k = [y^k_1, y^k_2, \cdots, y^k_n]$ is an auxiliary task, the GTI framework aims (1) to extract features from predicting Y^k, and (2) to use the extracted features for predicting Y^m. These notations are summarized in Table I.

Our GTI network is composed of the following primary components:

1) **Sentence Encoder:** handles the generation of the task specific semantic representations $\mathbf{S^k_{aux}} \in \mathbb{R}^{n\times d}$ and $\mathbf{S_m} \in \mathbb{R}^{n\times d}$ for the auxiliary task *k* and the main-task respectively from the input sequence X (where *n* is the number of tokens and *d* is the dimension of the hidden representation).
2) **AuxTask$^{\mathbf{k}}$ Network:** handles the generation of the output tags Y^k from the $\mathbf{S^k_{aux}}$.
3) **Gated Interaction Layer (GIL):** extracts an auxiliary feature vector $\mathbf{z_f}$ using the output tags from the auxiliary tasks under consideration via the Lemb$^{\mathrm{k}}$, the Trans$^{\mathrm{k}}$, the Compose$^{\mathrm{k}}$, the Gate$^{\mathrm{k}}$ and sum-blocks sub-modules.
4) **Main-task (MT) Network:** generates main/target task output tags Y^m from $\mathbf{S_m}$ and $\mathbf{z_f}$.

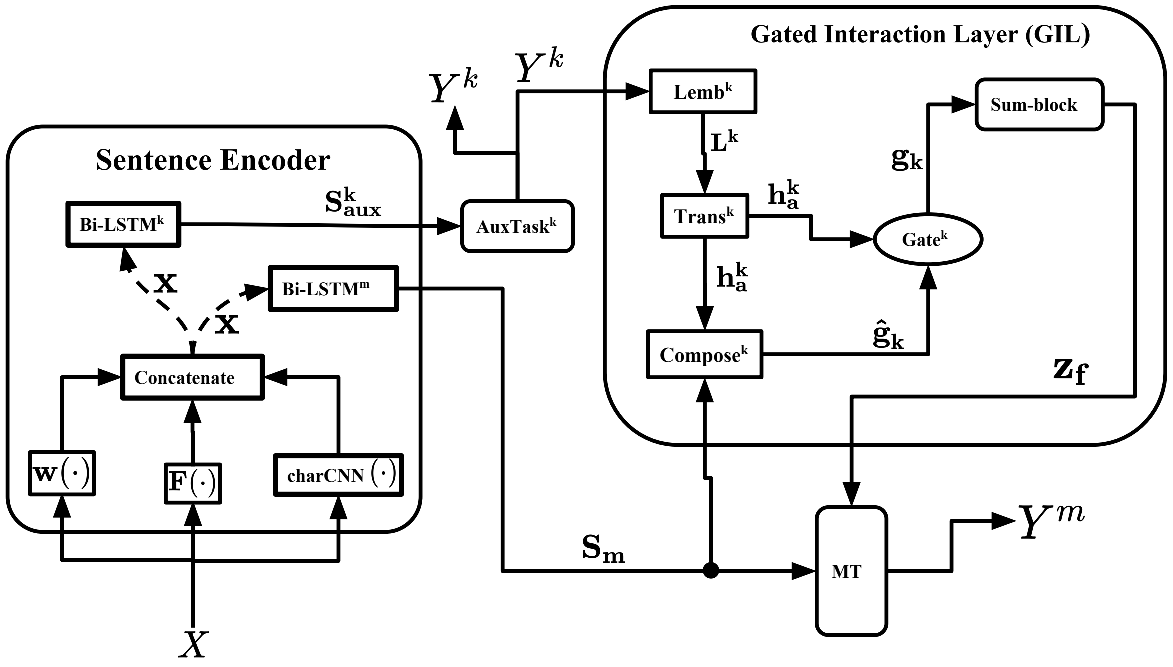
For sequence tagging problems such as NER, the tagging scheme imposes a constraint on the order of the output tags. For example, using the IOBES tagging scheme, output tag sequence starting with I- tag without a preceding B-tag or having I-MISC following a B-LOC is an illegal and meaningless sequence. Softmax classifiers fail to capture the tag-order constraint imposed by the tagging scheme.

Unlike the softmax classifier, CRF models the tagging decisions jointly hence capturing the dependencies between them [5], [7]. Therefore we use CRF to predict the output tags for both the AuxTask$^{\mathrm{k}}$ and Main-task (MT) networks. For each task under consideration, we compute the sentence CRF loss [2] using the forward-backward algorithm at training time. Following [1], we use the Viterbi algorithm to find the most likely tag sequence during testing.

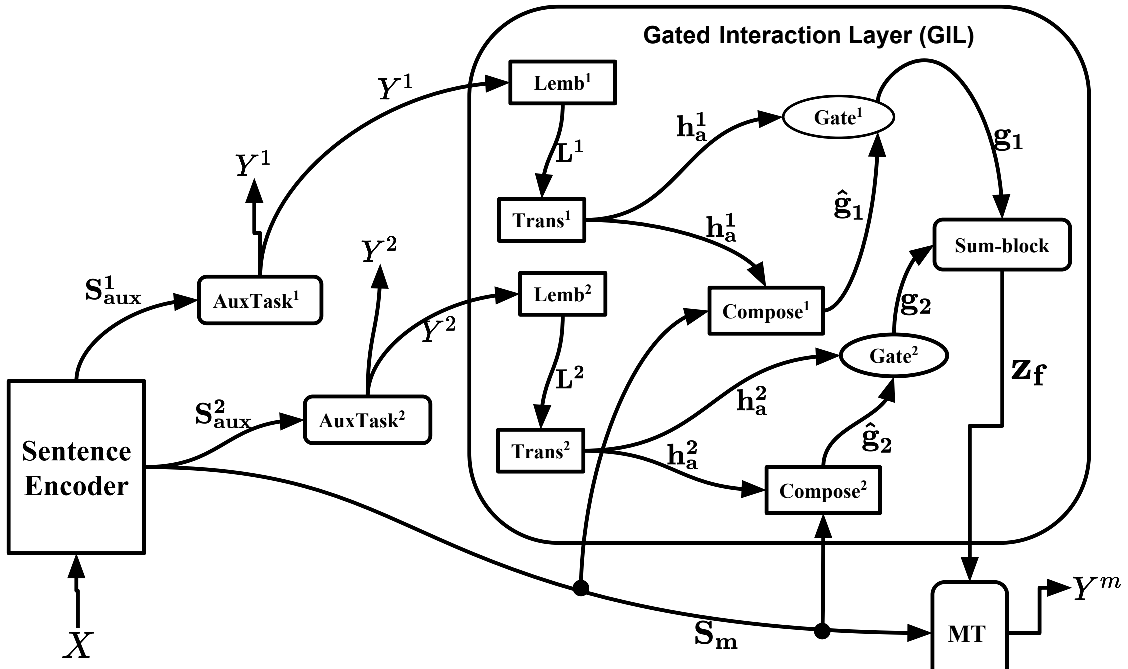
Sentence Encoder: The first stage of this layer generates the embedding $\mathbf{x_i} \in \mathbb{R}^d$ for each token x_i as the concatenation of the word embedding $\mathbf{w}(x_i)$, the character-level embedding from $\mathrm{charCNN}(x_i)$ and the 'token/word format' embedding $\mathbf{F}(x_i)$ as shown below:

$$\mathbf{x_i} = [\mathbf{w}(x_i); \mathrm{charCNN}(x_i); \mathbf{F}(x_i)] \tag{1}$$

$\mathbf{w}(\cdot)$ is a token look-up table which maps the token x_i to a high dimensional vector $\mathbf{w}(x_i)$. The word format embedding $\mathbf{F}(\cdot)$ (initialised as a 1-hot feature vector) maps the 'token-format' (i.e. whether it is numeric, punctuation, lower, upper-cased or alphanumeric) of x_i into a representation vector. $\mathrm{charCNN}(\cdot)$ is a character encoding sub-network computing the embedding of an input word based on its characters. As shown by [5], [7], [10], learning a word representation from its character captures important morphological information useful for solving several tagging tasks such as NER, POS etc. Within $\mathrm{charCNN}(\cdot)$, a look-up table first maps the sequence



(a) The basic form of the GTI architecture with a main-task Y^m and one auxiliary task Y^k.



(b) The GTI architecture with two auxiliary tasks sub-networks (*AuxTask*^{1} and *AuxTask*^{2} generating the output labels Y^1 and Y^2 respectively) and a single main/target task sub-network (*MT* generating the labels for the main Task Y^m).

Fig. 1: Gated Task Interaction (GTI) Architectures: *AuxTask*^{k} ($\{k : 1 \leq k \leq K\}$) and *MT* are the auxiliary and the main task sub-networks respectively, $\mathbf{S_m}$ and $\mathbf{S^k_{aux}}$ are the task-specific representation vectors generated from the input token sequence X for learning the main and auxiliary task *k* respectively.

of characters of a given token x_i into a vector sequence $\mathbf{C^i} = \left[\mathbf{c^i_1}, \mathbf{c^i_2}, \cdots, \mathbf{c^i_l}\right]$, where $\mathbf{c^i_j}$ is the vector *j*-th character in the *i*-th token of length *l*. The generated vector sequence is then passed to a character-CNN [7], [8], which computes the character-level embedding feature vector of the input token x_i after the convolution operations. The character look-up table is updated during training.

The last stage of the sentence encoder is the generation of the contextual-semantic representation vectors ($\mathbf{S^k_{aux}}$ and $\mathbf{S_m}$) from the input sequence embedding $\mathbf{x}$ as shown in Fig. 1a. $\mathbf{S^k_{aux}}$ and $\mathbf{S_m}$ are the task-specific representation vectors computed to solve the auxiliary task *k* and the main/target-task *m* respectively. BiLSTM network is adopted to compute these vectors. $\mathbf{S^k_{aux}}$ and $\mathbf{S_m}$ are obtained by concatenating the backward and the forward context representation vectors from their respective BiLSTM encoder. Dropouts of rate 0.25 is applied to the input of the all the BiLSTMs networks.

AuxTask^k Network: This sub-network takes as input the auxiliary task-specific semantic representation $\mathbf{S^k_{aux}}$ and generate the label sequence $Y^k = [y^k_1, y^k_2, \cdots, y^k_n]$ for auxiliary task *k*. The AuxTask^k network consists of a linear transformation layer to process the $\mathbf{S^k_{aux}}$ into a hidden representation $\mathbf{A^k} \in \mathbb{R}^{n\times t}$ (where *n* is the number of tokens and *t* is the number possible output tags under the auxiliary task *k*) as shown in Eq. (2).

$$\mathbf{A^k} = \mathbf{W^k_A}\mathbf{S^k_{aux}} + \mathbf{b^k_A} \tag{2}$$

where $\mathbf{W^k_A}$ and $\mathbf{b^k_A}$ are trainable parameters. The $\mathbf{A^k}$ is then passed to the CRF classifier which generates the Y^k tags.

Gated Interaction Layer (GIL): This layer controls the flow of information between the AuxTask^k and the MT sub-networks. The GIL computes the auxiliary feature vector $\mathbf{z_f}$ from the output of the AuxTask^k network and $\mathbf{S_m}$ (computed by the sentence encoder). As shown in Fig. 1a, it consists of the Lemb^k, the Trans^k, the Compose^k, the Gate^k and the sum-block layers. Within the Lemb^k, the Y^k (the one-best tagging sequence predicted by the AuxTask^k network) is mapped into a label embedding vector $\mathbf{L^k}$ using a randomly initialized lookup table. The Trans^k consists of a single layer BiLSTM which processes the $\mathbf{L^k}$ vector into $\mathbf{h^k_a}$. Based on the $\mathbf{h^k_a}$ vector and $\mathbf{S_m}$, the gate control vector $\{\hat{\mathbf{g}}_\mathbf{k} : 1 \leq k \leq K\}$ is generated by the Compose^k sub-layer as shown in Eq. (3).

$$\hat{\mathbf{g}}_\mathbf{k} = \mathbf{W_k}\mathbf{h^k_a} + \mathbf{U_k}\mathbf{S_m} \tag{3}$$

where $\mathbf{U_k}$ and $\mathbf{W_k}$ are weight matrices for generating the $\hat{\mathbf{g}}_\mathbf{k}$. Given the $\hat{\mathbf{g}}_\mathbf{k}$ and $\mathbf{h^k_a}$ vectors, the Gate^k generates the gated auxiliary feature vector $\mathbf{g_k}$ for the auxiliary task *k* as shown below:

$$\mathbf{g_k} = \sigma\left(\hat{\mathbf{g}}_\mathbf{k}\right) \odot \mathbf{h^k_a} \tag{4}$$

where $\odot$ represents an element-wise product operation and $\sigma(\cdot)$ is the sigmoid activation function. The final stage is the computation of $\mathbf{z_f}$, which is a weighted summation of all the $\mathbf{g_k}$ vectors. This is done by the sum-block sub-module as shown in Eq. (5).

$$\mathbf{z_f} = \mathbf{W_f}\sum_{k=1}^{K}\left(\mathbf{g_k}\right) \tag{5}$$

Here $\mathbf{W_f}$ is a weight matrix for the generation of the $\mathbf{z_f}$ and K is the total number of auxiliary tasks under consideration.

Main-task (MT) Network: This is the sub-network for generating the sequence labels Y^m under the main task. It takes as input the $\mathbf{z_f}$ and the $\mathbf{S_m}$ representation vectors from the GIL and the sentence encoder respectively. These input vectors are then combined to compute the representation vector $\mathbf{h^m_f}$ as shown in Eq. (6).

$$\mathbf{h^m_f} = \tanh\left(\mathbf{W_m}\mathbf{S_m} + \mathbf{z_f}\right) \tag{6}$$

where $\mathbf{W_m}$ is a weight matrix. A single BiLSTM layer processes the $\mathbf{h^m_f}$ into the hidden representation $\mathbf{h^m}$ by concatenating the right context representation ($\overrightarrow{h}^{\mathbf{m}}$) and left context representation ($\overleftarrow{h}^{\mathbf{m}}$) vectors, i.e. $\mathbf{h^m} = [\overrightarrow{h}^{\mathbf{m}}; \overleftarrow{h}^{\mathbf{m}}]$. A linear transformation layer then processes the $\mathbf{h^m}$ into an intermediate representation vector $A^m_f \in \mathbb{R}^{n\times m}$ (where *n* is the number of tokens and *m* is the number possible output tags under the main task) as shown below:

$$\mathbf{A^m} = \mathbf{W_A}\mathbf{h^m} + \mathbf{b_A} \tag{7}$$

where $\mathbf{W_A}$ and $\mathbf{b_A}$ are trainable parameters. A CRF classifier then generates the output tags $Y^m = [y^m_1, y^m_2, \cdots y^m_n]$ from the $\mathbf{A^m}$.

B. Joint Model Training

A common approach to learning multiple objectives via MTL involves randomly selecting and alternating between the different tasks and their associated dataset [14], [15], [17]. During the training instance *t*, the loss function is computed only based on the current task's objective. But in this work, the main-task and auxiliary tasks sub-networks are jointly trained on only the same in-domain dataset.

Therefore, given the CRF loss L_m (computed by the main-task's network) and the losses from the AuxTask^k networks L_{aux} (see Eq. (8)), we define the joint training loss function J_{loss} as:

$$L_{aux} = \sum_{k=1}^{K} L(\hat{y}^{(k)}, y^{(k)}) \tag{8}$$

$$J_{loss} = L_m + L_{aux} \tag{9}$$

where $L(\hat{y}^{(k)}, y^{(k)})$ is the CRF loss computed from the auxiliary task *k*.

III. Experiment

We evaluate the performance of the proposed GTI framework on the CoNLL-2000 Chunking dataset [23] and CoNLL-2003 NER dataset [22]. We use the IOBES tagging scheme which as shown by [18] yields a better performance compared to the IOB and the BIO schemes.

- **CoNLL-2000 Chunking** was generated from the Wall Street Journal (WSJ) corpus with sections 15-18 as the training, section 20 for testing. We randomly sampled 1000 sentences from the training set as the validation dataset. Each sentence is pre-labelled with the POS tags and the syntactic chunk tags. Therefore the experiment conducted on this dataset predicts the POS tags as the

TABLE II: Models for CoNLL-2003 NER task.

Index	Model	Main/Target task	AuxTasks
1	GTI-NER-POS	NER	POS
2	GTI-NER-CHUNK	NER	Chunking
3	GTI-NER-CHUNK-POS	NER	Chunking and POS

auxiliary task and chunk labels prediction as the main-task. We refer to the GTI network trained on this dataset as GTI-CHUNK-POS.

- **CoNLL-2003 NER** contains annotations for the following entity types: LOC, PER, ORG and MISC. Each sentence is also pre-labelled with the chunking and POS tags. The dataset comes with training, development and test set splits. Unlike the works by [8], [9], [24], we only trained our models on the training set and use the development set for parameter tuning and report the results in terms of the models' performance on the test set. We performed experiments with 3 different model configurations/instances on this dataset as shown in the Table II. These models were trained with the prediction of the appropriate POS and Chunk tags as the auxiliary tasks and the named entity labels as the main-task.

A. *Model Setup and Hyperparameters*

- **Initialization:** The word embedding vectors were initialized with pre-trained embedding vectors of dimension equal to 100 from Glove[1] [6]. The out-of-vocabulary (OOV) words were initialized by uniform sampling from the distribution [-0.25,0.25]. During training, we do not update the embedding vectors. The character embedding was initialized by sampling from the uniform distribution [-0.5,0.5] with the dimension of the resulting vector equal to that of the pre-trained word embedding vectors.
- **Training:** We set the dropout rates for all the LSTMs, character-CNN and the output of the Trans^k layers to 0.25. The models are trained with the batch-size set as 10. We use the Nadam optimizer with the initial learning rate α_0. During training, the learning rate is updated via a shifted cosine annealing function proposed by [25]. This approach anneals the learning rate from its initial value α_0 to 0 during a single cosine cycle and introduces extra parameters such as the total number of iterations (total number of epochs) T and the number of annealing cycle M. After several initial experiments to pick the best values of T, M and α_0, we choose $\alpha_0 = 0.001$, M=9 and T=270. We did not train the models for the entire T but rather stopped the training 10 epochs after the first 2 cycles (i.e. each model is trained for a total of 70 epochs). This is because we observed a marginal improvement in each model's performance after the first 2 cycles. The neural network architecture was implemented using Keras (version 2.0.5 with tensorflow backend). The models were trained on GeForce GTX 1080 Ti 11 GB RAM GPU. The code will be available at online[2].

[1]http://nlp.stanford.edu/projects/glove/
[2]https://github.com/kaeflint/GTI

TABLE III: Effect of hidden state size of LSTM on the performance of the GTI models on the CoNLL-2003 NER dataset.

Model	State size	F_1 score		
		min	mean ($\pm$ **std**)	max
GTI-NER-POS	100	90.94	91.09±0.09	91.17
	150	90.84	91.04±0.13	91.22
	200	91.20	91.33±0.14	91.56
GTI-NER-CHUNK	100	90.88	91.08±0.13	91.22
	150	90.80	90.93±0.08	91.04
	200	91.11	91.22±0.09	91.33
GTI-NER-CHUNK-POS	100	90.83	90.86±0.02	90.89
	150	91.05	91.18±0.10	91.30
	200	91.27	91.38±0.07	91.49

TABLE IV: Effect of hidden state size of LSTM on the performance of the GTI models on the CoNLL-2000 Chunking dataset.

Model	State size	F_1 score		
		min	mean ($\pm$ **std**)	max
GTI-CHUNK-POS	100	94.91	94.96±0.04	95.03
	150	95.01	95.05±0.03	95.11
	200	95.08	95.15±0.05	95.22

There are several parameters whose value affects the performance of the proposed approach. These parameters include but not limited to the dropout rates, number of BiLSTM layers, initial learning rate, choice of RNN units (either LSTM or GRU). In this work, we only explore the impact of the LSTM hidden state size. Therefore, keeping parameters such as dropout rate, dimension of the character and word-format embedding and learning rate the same across the different configurations of the GTI network, we train the models with different the LSTM hidden state size selected from the set $\{100, 150, 200\}$. We evaluate the performance on the NER and chunking prediction based on the official evaluation metric (micro-averaged F_1 score).

IV. Results and Discussion

In this section, we discuss the performance of our GTI models. Under all the training scenarios, we observed that the GTI network's performance on the main-task depends on the auxiliary tasks handled by the AuxTask^k sub-networks. The results (see Table III and Table IV) obtained suggest that aside the relatedness between the tasks, the dimension of the LSTM hidden state also affect the performance especially in the case of NER tagging. Increasing the hidden state size improves the performance at a small cost of computational overhead due to increasing size of the model.

A. *Performance on the NER Tagging*

The results of our GTI models and existing approaches for the NER task are summarised in Table V. To make a fair comparison, we report the F_1 scores of existing models with and without the use of external labelled corpus such as the gazetteers, AIDA and PTB-POS datasets. Our GTI models for the NER task are trained only on the CoNLL-2003 NER dataset. Among our models, the GTI-NER-CHUNK-POS achieved the overall best performance with a score of 91.49 (91.38 $\pm$ 0.07) approximately 0.44 less than the current state-of-the-art model [9]. Their MTL model for NER tagging and

TABLE V: CoNLL-2003 test set F_1 score. † indicates models trained with pre-trained word embedding.

External Resources	Model	$F_1 \pm$ **std**	
gazetteers	Collobert et al. 2011 [1] †	89.59	
	Chiu et al. 2016 [8] †	91.62±0.33	
AIDA dataset	Luo et al. 2015 [26]	91.20	
CoNLL2000/ PTB-POS	Yang et al. 2017 [24]†	91.26	
1B Word dataset & `4096-8192-1024`	Peters et al. 2017 [9]†	91.93±0.19	
1B Word dataset		91.62±0.23	
None	Peters et al. 2017 [9]†	90.87±0.13	
	Collobert et al. 2011 [1]†	88.67	
	Chiu et al. 2016 [8] †	90.91±0.20	
	Luo et al. 2015 [26]	89.90	
	Yang et al. 2017 [24]†	91.20	
	Rei 2017 [27] †	86.26	
	Lample et al. 2016 [5] †	90.94	
	Ma et al. 2016 [7]†	91.21	
	GTI-NER-POS†	mean	91.33±0.14
		max	91.56
	GTI-NER-CHUNK†	mean	91.22±0.09
		max	91.33
	GTI-NER-CHUNK-POS†	mean	91.38±0.07
		max	91.49

TABLE VI: CoNLL-2000 Chunking test set F_1 score.

External Resources	Model	$F_1 \pm$ **std**	
PTB-POS	Søgaard et al. 2016 [14]	95.56	
	Hashimoto et al. 2017	95.77	
CoNLL2000/ PTB-POS	Yang et al. 2017 [24]	95.41	
1B Word dataset	Peters et al. 2017 [9]	96.37 ±0.05	
None	Søgaard et al. 2016 [14]	95.28	
	Yang et al. 2017 [24]	94.66	
	Hashimoto et al. 2017	95.02	
	Rei 2017 [27]	93.88	
	Peters et al. 2017 [9]	95.00±0.08	
	GTI-CHUNK-POS	mean	95.15±0.05
		max	95.22

neural language modelling tasks was trained along with a large amount of extra training corpus which resulted in a score of 91.93. Without any external resources, their model's F_1 score dropped to 90.87 representing a significant loss in performance. Aside from the model proposed by [8], our GTI-NER-CHUNK-POS outperforms all other systems/models, including those trained using external corpus like gazetteers.

Table III presents the impact of the hidden state size. One can clearly observe that in most cases increasing the LSTM's hidden state size produces a better performance on the NER tagging. Increasing the hidden state size enables the GTI models to effectively extract/encode the necessary information needed to enhance the performance on the NER task. In all the experiments on the CoNLL-2003 NER dataset, we observed that the choice of auxiliary task consistently affects the overall performance of the GTI models. Compared to chunking task, learning NER along with the POS tagging as auxiliary task (GTI-NER-POS) produced the better model but at a cost of higher variance. On the other hand, among our models, jointly learning the NER tagging along with the linguistic features (chunking and POS tagging) produced the overall best performance in terms of the mean F_1 score and it achieved lower variance.

TABLE VII: Ablation Results. The experiments are conducted on the CoNLL-2003 NER corpus with the LSTM state size is set to 200. The MTL models are trained with NER as the main-task and chunking and POS tagging as the auxiliary tasks. The STL models are trained to generate NER tags.

Model	F_1		
	Min	Mean± **std**	Max
Single Task Learning (STL) models			
BiLSTM-CRF(1)	90.75	90.85±0.084	90.91
BiLSTM-CRF(2)	90.69	90.83±0.08	90.91
Multi-task Learning (MTL) models			
Vanilla MTL	90.99	91.10±0.1	91.23
Pipeline MTL	90.73	90.81±0.08	90.94
TI-NER-CHUNK-POS	90.93	91.13±0.11	91.26
GTI-NER-CHUNK-POS	91.27	91.38±0.07	91.49

B. *Performance on Chunking*

As shown in Table VI, the performance of our GTI-CHUNK-POS model on the CoNLL-2000 Chunking dataset is comparatively lower than the state-of-the-art model [9]. The models by [9], [14], [15] achieved high performance by using external resources/datasets. For example, the baseline model introduced by [9] achieved a score of 95.00, approximately 1.37 lower than when they expanded the training data with external corpus. The lower performance of the models trained on only the CoNLL-2000 chunking dataset can be attributed to the limited amount of available training data. Our GTI-CHUNK-POS model outperforms majority of the baselines trained without extra resources. Similar to the GTI models trained on the CoNLL-2003 NER corpus, we observed that increasing the hidden state size (see Table IV) improves further the performance of the GTI-CHUNK-POS model but with a higher variance.

C. *Ablation*

To investigate the impact of the *Gated Interaction Layer (GIL)* in our proposed GTI approach for the generation of the NER tags, we performed 5 ablation studies. The first 2 (BiLSTM-CRF(1) and BiLSTM-CRF(2)) are regular single-task learning baseline (Non-MTL) models for generating the NER tags. BiLSTM-CRF(1) is similar to the approach proposed by [5], [7] for the task of NER. They trained a BiLSTM-CRF architecture using the words and character-level representation as inputs. BiLSTM-CRF(2) extends the words and character-level representations with linguistic features (chunking and the POS tags) as additional inputs to the model for the NER prediction.

Three re-implementations of the GTI multi-task learning approach are also presented. Vanilla MTL is the re-implementation without the *Gated Interaction Layer (GIL)*. This model simultaneously learns the generation of the NER tasks along with the linguistic features (chunking and the POS tags). Pipeline MTL is a variant of MTL model analogous to the BiLSTM-CRF(2) model and it is a pipeline system which predicts the labels for auxiliary tasks (chunking and POS tag generation) first and then uses them as features for the main task (NER tagging). Without the GIL components

(Trans^k, Composite^k and Gate^k), our proposed architecture is more similar Pipeline MTL. Finally, Task Interaction-NER-CHUNK-POS (TI-NER-CHUNK-POS) is also modeled after the GTI-NER-CHUNK-POS model without the Gate^k within the *GIL*. TI-NER-CHUNK-POS's *Gated Interaction Layer* consists of only the Lemb^k, Trans^k, Composite^k and the Sum-block sub-layers. We compare the performance of the TI-NER-CHUNK-POS to GTI-NER-CHUNK-POS to verify that the performance gain over the Vanilla MTL and Pipeline MTL models is as a result of the gating mechanism controling the flow of information between the tasks.

As shown in Table VII, we observed no significant change in performance when we augmented the input features with the linguistic features. Learning the linguistic features (chunking and the POS tags) simultaneously along with NER tagging under the Vanilla MTL approach yielded a marginal performance gain of about 0.25 over the single models, BiLSTM-CRF(1) and BiLSTM-CRF(2). The performance gain over the single models can be attributed to the inductive bias information provided by the auxiliary tasks [28]. This enables the model to learn a shared representation beneficial to the main task.

The Pipeline MTL was expected to further improve the performance of the Vanilla MTL but surprisingly, the F_1 score dropped by about 0.29 almost similar to the performance of the non-MTL models (BiLSTM-CRF(1) and BiLSTM-CRF(2)). With the addition of the Trans^k and Composite^k components within the GIL, the Task Interaction model (TI-NER-CHUNK-POS) obtained a higher performance compared to the Pipeline MTL model. But without the Gate^k component within the *Gated Interaction Layer*, the TI-NER-CHUNK-POS model achieved no significant performance gain (F_1 score of 91.13 ± 0.11) over the Vanilla MTL model (F_1 score of 91.10 ± 0.1). Adding the Gate^k component further improves the performance achieving a gain of about 0.57, 0.28 and 0.25 higher than the Pipeline MTL, Vanilla MTL and TI-NER-CHUNK-POS models respectively. The improvement in the F_1 score over the TI-NER-CHUNK-POS model comes at no increase in the number of hyperparameters of the model. As mentioned above, the gating mechanism enhances the interaction between the main task and its associated auxiliary tasks by controlling the information flow between tasks under consideration. Overall, the MTL models consistently outperform the single models (BiLSTM-CRF(1) and BiLSTM-CRF(2)) and the use of our gating mechanism via the *Gated Interaction Layer* can further improve the performance on the sequence generation task.

V. Related Works

Linguistic sequence labelling tasks such as NER, Chunking, SRL, POS etc, have been well studied over the years. But several of these approaches such as SVM [29], CRF [2] and Hidden Markov Models (HMMs) [3] depend on hand-crafted features. Models adopting hand-crafted features are difficult to adopt for a new domain or task. For example, to adapt a NER model for POS tagging, new hand-crafted features have to be generated. This is a very expensive approach to learning.

The automatic feature engineering via neural models has recently received a lot of attention. [1] used CNN to extract input features vectors from the sequence of words. These features are then passed to CRF classifier. Contrast to this, other works by [5], [30] replaced CNN with LSTM/BiLSTMs for extracting features from the input words. Also, [7] combined both LSTM and CNN. [4] extended the input word-level feature vectors with hand-crafted features such as spelling features (e.g it whether starts with a capital letter, whether it has all lower case letters etc.) for a given word in the input sequence. Aside from the word-level features, recent works [5], [7], [8] rely on the character-level features. All these approaches achieved remarkable performance over the traditional approaches to sequence labelling. To further improve the performance of neural models, other works [11]–[13] leverage additional information from linguistic features such as chunking and POS tags. However, the change in performance depends on the degree of semantic relatedness between the linguistic features and the target task.

Multi-task Learning (MTL): The downside to the application of deep learning is that it is data intensive requiring a large amount of labelled examples. To tackle the data scarcity problem, works by [1], [8] augmented the dataset with language specific resources such as gazetters. Furthermore, recent approaches suggest guiding the learning process with extra knowledge via multi-task learning (MTL). Following the work by [31], multi-task learning has been applied to many NLP problems as well as other neural network architectures. The MTL approach to learning involves optimising multiple tasks simultaneously. This mostly involves sharing the model's weights between the tasks under consideration. Under the Sequence tagging, a number of MTL models have been proposed. [32] explored a shared representation learning strategy that supports domain adaptation for multiple tasks. [19] explored identifying the beneficial auxiliary tasks that can be modelled along with a given main/target task. Also, [33] proposed a unified network where the weights of the word embedding layer is shared between multiple sequence labelling tasks such as POS, SRL, Chunking and NER. [14] presented an approach to MTL where the different objectives are supervised at different levels. A similar approach was adopted by [15] where they successively grew their network depth to tackle increasingly complex NLP tasks. In contrast to these works, the outputs of our auxiliary tasks are fed-back into the network via a *Gated Interaction Layer (GIL)* which transforms them into features usable by the other tasks under consideration. This approach controls the flow of information between the multiple tasks.

A distinction between the different MTL approaches also lies in the training strategy employed. A common approach involves training the MTL model on different task specific corpus by randomly switching between the different tasks and updating both the task-specific and shared parameters based on its corpus. [15], [17], [19] employed this training strategy. A joint end-to-end model training strategy is mostly suitable for cases where the alternative tasks are treated as auxiliary objectives on the same dataset. Works by [9], [10], [27] trained a sequence labelling task such as NER along with unsupervised learning tasks such as language modelling. The joint training approach eliminates the need for external corpus for the auxiliary tasks as the same dataset is used to learn all the multiple tasks. We employed this strategy to train all the GTI models.

VI. Conclusion

In this work, we proposed an MTL framework for sequence labelling, which exploits the relatedness between a given main/target task and its associated auxiliary tasks. The experimental results show that by jointly learning the linguistic features along with the target sequence labelling task, the GTI model achieves high performance on the baseline datasets for chunking and NER tagging tasks. The main and auxiliary tasks are trained on the same dataset eliminating the need for extra corpus for each task. This makes the GTI framework ideal for low resource tasks such as NER and Chunking.

In the future, we aim to improve the performance of our GTI framework by exploring further the impact of other hyperparameters such as the dropout rates and number of BiLSTM layers. Finally, we intend to test the performance of our proposed approach on learning other sequence labelling tasks such as SRL and Error Detection.

Acknowledgment

This work is partially funded by the VCRS scholarship, by the EU Horizon 2020 Research and Innovation Programme under Grant 690238 for DESIREE Project, by the UK EPSRC under Grant EP/P031668/1, by the BT Ireland Innovation Centre (BTIIC).

References

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[2] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[3] G. Zhou and J. Su, "Named entity recognition using an hmm-based chunk tagger," in *Proceedings of ACL-2002*, 2002, pp. 473–480.

[4] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv:1508.01991*, 2015.

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of NAACL-HLT*, 2016, pp. 260–270.

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on EMNLP*, 2014, pp. 1532–1543.

[7] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of ACL-2016*, vol. 1, 2016, pp. 1064–1074.

[8] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[9] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," *arXiv:1705.00108*, 2017.

[10] L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han, "Empower Sequence Labeling with Task-Aware Neural Language Model," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[11] L. Luo, Z. Yang, P. Yang, Y. Zhang, L. Wang, H. Lin, and J. Wang, "An attention-based bilstm-crf approach to document-level chemical named entity recognition," *Bioinformatics*, vol. 34, no. 8, pp. 1381–1388, 2017.

[12] S. K. Sienčnik, "Adapting word2vec to named entity recognition," in *Proceedings of the 20th nordic conference of computational linguistics, nodalida 2015, may 11-13, 2015, vilnius, lithuania*, no. 109. Linköping University Electronic Press, 2015, pp. 239–243.

[13] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proceedings of the 2017 Conference on EMNLP*, 2017, pp. 1506–1515.

[14] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *Proceedings of the 54th Annual Meeting of the ACL (Volume 2: Short Papers)*, vol. 2, 2016, pp. 231–235.

[15] K. Hashimoto, Y. Tsuruoka, R. Socher *et al.*, "A joint many-task model: Growing a neural network for multiple nlp tasks," in *Proceedings of the 2017 Conference on EMNLP*, 2017, pp. 1923–1933.

[16] E. Strubell, P. Verga, D. Andor, D. Weiss, and A. McCallum, "Linguistically-informed self-attention for semantic role labeling," in *Proceedings of the 2018 Conference on EMNLP*, 2018, pp. 5027–5038.

[17] H. M. Alonso and B. Plank, "When is multitask learning effective? semantic sequence prediction under varying data conditions," in *Proceedings of the 15th Conference of the EACL: Volume 1, Long Papers*, vol. 1, 2017, pp. 44–53.

[18] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," 2017.

[19] J. Bingel and A. Søgaard, "Identifying beneficial task relations for multi-task learning in deep neural networks," in *Proceedings of the 15th Conference of the EACL: Volume 2, Short Papers*, vol. 2, 2017, pp. 164–169.

[20] M. Zhang, Y. Zhang, and D.-T. Vo, "Gated neural networks for targeted sentiment analysis," in *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 3087–3093.

[21] W. Xue and T. Li, "Aspect based sentiment analysis with gated convolutional networks," in *Proceedings of the 56th Annual Meeting of the ACL (Volume 1: Long Papers)*, vol. 1, 2018, pp. 2514–2523.

[22] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of HLT-NAACL-2003-Volume 4*. ACL, 2003, pp. 142–147.

[23] E. F. Tjong Kim Sang and S. Buchholz, "Introduction to the conll-2000 shared task: Chunking," in *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on CoNLL*. ACL, 2000, pp. 127–132.

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

[25] I. Loshchilov and F. Hutter, "Sgdr: stochastic gradient descent with restarts," *ICLR*, 2017.

[26] G. Luo, X. Huang, C.-Y. Lin, and Z. Nie, "Joint entity recognition and disambiguation," in *Proceedings of the 2015 Conference on EMNLP*, 2015, pp. 879–888.

[27] M. Rei, "Semi-supervised multitask learning for sequence labeling," *ACL*, 2017.

[28] S. Ruder, "An overview of multi-task learning in deep neural networks," *CoRR*, vol. abs/1706.05098, 2017.

[29] J. Kazama, T. Makino, Y. Ohta, and J. Tsujii, "Tuning support vector machines for biomedical named entity recognition," in *Proceedings of the ACL-02 Workshop on Natural Language Processing in the Biomedical Domain - Volume 3*, ser. BioMed '02, 2002, pp. 1–8.

[30] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and whats next," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 473–483.

[31] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

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

[33] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.