Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction

Pankaj Gupta
Corporate Technology, Siemens AG
CIS, University of Munich (LMU)
Munich, Germany
`gupta.pankaj.ext@siemens.com`
`pankaj.gupta@campus.lmu.de`

Hinrich Schütze
CIS, University of Munich (LMU)
`inquiries@cis.lmu.org`

Bernt Andrassy
Corporate Technology, Siemens AG
Munich, Germany
`bernt.andrassy@siemens.com`

Abstract

This paper proposes a novel context-aware joint entity and word-level relation extraction approach through semantic composition of words, introducing a Table Filling Multi-Task Recurrent Neural Network (TF-MTRNN) model that reduces the entity recognition and relation classification tasks to a table-filling problem and models their interdependencies. The proposed neural network architecture is capable of modeling multiple relation instances without knowing the corresponding relation arguments in a sentence. The experimental results show that a simple approach of piggybacking candidate entities to model the label dependencies from relations to entities improves performance.

We present state-of-the-art results with improvements of 2.0% and 2.7% for entity recognition and relation classification, respectively on CoNLL04 dataset.

1 Introduction

Relation classification is defined as the task of predicting the semantic relation between the annotated pairs of nominals (also known as relation arguments). These annotations, for example named entity pairs participating in a relation are often difficult to obtain. Traditional methods are often based on a pipeline of two separate subtasks: Entity Recognition (ER[1]) and Relation Classification (RC), to first detect the named entities and then performing relation classification on the detected entity mentions, therefore ignoring the underlying interdependencies and propagating errors from the entity recognition to relation classification. The two subtasks together are known as End-to-End relation extraction.

Relation classification is treated as a sentence-level multi-class classification problem, which often assume a single relation instance in the sentence. It is often assumed that entity recognition affects the relation classification, but it is not affected by relation classification. Here, we reason with experimental evidences that the latter is not true. For example, in Figure 1, relation *Work_For* exists between *PER* and *ORG* entities, *ORGBased_in* between *ORG* and *LOC*, while *Located_In* between *LOC* and *LOC* entities. Inversely, for a given word with associated relation(s), the candidate entity types can be detected. For example, in Figure 2, for a given relation, say *Located_in*, the candidate entity pair is (*LOC*, *LOC*). Therefore, the two tasks are interdependent and optimising a single network for ER and RC to model the interdependencies in the candidate entity pairs and corresponding relations is achieved via the proposed joint modeling of subtasks and a simple piggybacking approach.

Joint learning approaches (Roth and Yih, 2004; Kate and Mooney, 2010) built joint models upon complex multiple individual models for the subtasks. (Miwa and Sasaki, 2014) proposed a joint entity and relation extraction approach using a history-based structured learning with a table representation; however, they explicitly incorporate entity-relation label interdependencies, use complex features and search heuristics to fill table. In addition, their state-of-the-art method is structured prediction and not based on neural network frameworks. However, *deep learning* methods such as recurrent and convolutional neural networks (Zeng et al., 2014; Zhang and Wang, 2015; Nguyen and Grishman, 2015) treat relation

[1]Entity Recognition (ER) = Entity Extraction (EE); Relation Classification (RC) = Relation Extraction (RE)

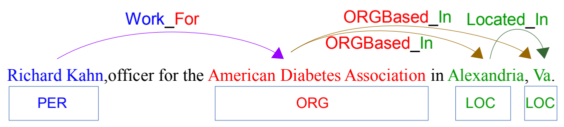

Work_For
ORGBased_In
Located_In
ORGBased_In
Richard Kahn,officer for the American Diabetes Association in Alexandria, Va.
PER
ORG
LOC
LOC

Figure 1: An entity and relation example (CoNLL04 data). *PER*: Person, *ORG*: Organization, *LOC*: Location. Connections are: *PER* and *ORG* by *Work_For*; *ORG* and *LOC* by *OrgBased_In*; *LOC* and *LOC* by *Located_In* relations.

	PER	**LOC**	**ORG**	**Other**
PER	KILL	Live_In	Work_For	⊥
LOC	Live_In	Located _In	ORG Based_In	⊥
ORG	Work_For	ORG Based_In	⊥	⊥
Other	⊥	⊥	⊥	⊥

Figure 2: Entity-Relation dependencies (CoNLL04 dataset).

	Richard	Kahn	,	officer	for	the	American	Diabetes	Association	in	Alexandria	,	Va	.
Richard	*B-PER*, ⊥													
Kahn	⊥	*L-PER*, ⊥												
,	⊥	⊥	*O*, ⊥											
officer	⊥	⊥	⊥	*O*, ⊥										
for	⊥	⊥	⊥	⊥	*O*, ⊥									
the	⊥	⊥	⊥	⊥	⊥	*O*, ⊥								
Ameriacan	⊥	⊥	⊥	⊥	⊥	⊥	*B-ORG*, ⊥							
Diabetes	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*I-ORG*, ⊥						
Association	⊥	*Work_For*	⊥	⊥	⊥	⊥	⊥	⊥	*L-ORG*, ⊥					
in	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥				
Alexandria	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*ORGBased_In*	⊥	*U-LOC*, ⊥			
,	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥		
Va	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*ORGBased_In*	⊥	*Located_In*	⊥	*U-LOC*, ⊥	
.	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥

Table 1: Entity-Relation Table for the example in Figure 1. Demonstrates the word-level relation classification via a Table-Filling problem. The symbol (⊥) indicates *no_relation* word pair. Relations are defined on the words, instead of entities. The diagonal entries have the entity types and ⊥ relations to the words itself, while the off-diagonal entries are the relation types.

classification as a sentence-level multi-class classification, and rely on the relation arguments provided in the sentence. Therefore, they are incapable in handling multiple relation instances in a sentence and can not detect corresponding entity mention pairs participating in the relation detected.

We tackle the limitations of joint and deep learning methods to detect entities and relations. The contributions of this paper are as follows:

1. We propose a novel Table Filling Multi-task Recurrent Neural Network to jointly model entity recognition and relation classification tasks via a unified multi-task recurrent neural network. We detect both entity mention pairs and the corresponding relations in a single framework with an entity-relation table representation. It alleviates the need of search heuristics and explicit entity and relation label dependencies in joint entity and relation learning. As far as we know, it is the first attempt to jointly model the interdependencies in entity and relation extraction tasks via multi-task recurrent neural networks.

   We present a word-level instead sentence-level relation learning via word-pair compositions utilising their contexts via Context-aware RNN framework. Our approach has significant advantage over state-of-the-art methods such as CNN and RNN for relation classification, since we do not need the marked nominals and can model multiple relation instances in a sentence.

2. Having named-entity labels is very informative for finding the relation type between them, and vice versa having the relation type between words eases problem of named-entity tagging. Therefore, a simple approach to piggyback candidate named entities for words (derived from the associated relation type(s) for each word) to model label dependencies improves the performance of our system. In addition, the sequential learning approach in the proposed network learns entity and relation label dependencies via sharing model parameters and representations, instead modeling them explicitly.

3. Our approach outperforms the state-of-the-art method by 2.0% and 2.7% for entity recognition and relation classification, respectively on CoNLL04 dataset.

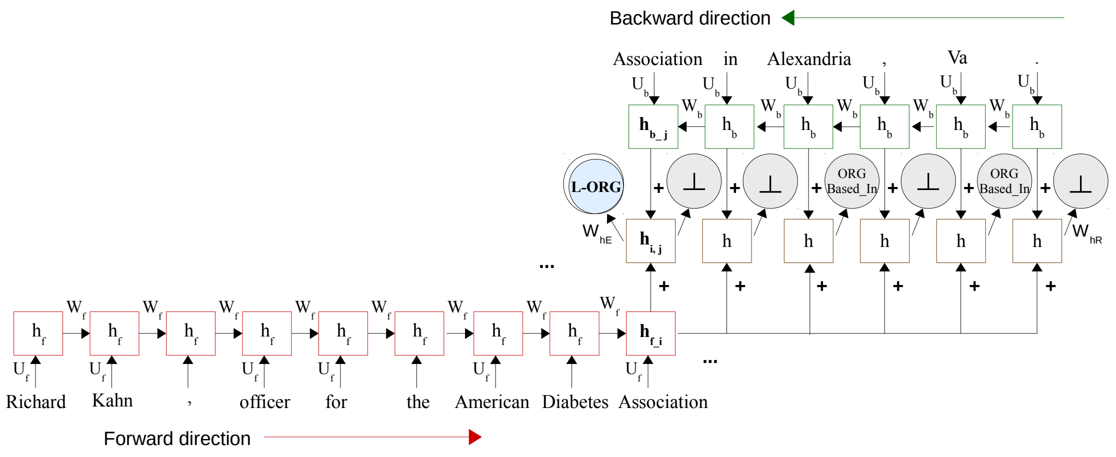


Figure 3: The Table Filling Multi-Task Recurrent Neural Network (TF-MTRNN) for joint entity and word-level relation extraction. Overlapping circle: Entity labels; Single circle: Relation label. In the above illustration, the word *Association* at $t = i$ (where; $t = 0, ..., i, ..., N$) from forward network is combined with each of the remaining words in the sequence (Figure 1), obtained from backward network at each time step, $j = i, ..., N$. Similarly, perform all possible word pair compositions to obtain Table 1. *ORGBased_In* relation in each word-pairs: (*Association*, *Alexandria*) and (*Association*, *Va*).

2 Methodology

2.1 Entity-Relation Table

As the backbone of our model we adopt the table structure proposed by Miwa and Sasaki (2014), shown in Table 1. This structure allows an elegant formalization of joint entity and relation extraction because both entity and relation labels are defined as instances of binary relations between words w_i and w_j in the sentence. An entity label is such a binary relationship for $i = j$, i.e., a cell on the diagonal. A relation label is such a binary relationship for $i \neq j$, i.e., an off-diagonal cell. To eliminate redundancy, we stipulate that the correct label for the pair (w_i, w_j) is relation label r if and only if $i \neq j$, w_i is the last word of a named entity e_i, w_j is the last word of a named entity e_j and $r(e_i, e_j)$ is true.[2] We introduce the special symbol $\perp$ for "no_relation", i.e., no relation holds between two words.

Apart from the fact that it provides a common framework for entity and relation labels, another advantage of the table structure is that modeling multiple relations per sentence comes for free. It simply corresponds to several (more than one) off-diagonal cells being labeled with the corresponding relations.

2.2 The Table Filling Multi-Task RNN Model

Formally, our task for a sentence of length n is to label $n(n+1)/2$ cells. The challenge is that the labeling decisions are highly interdependent. We take a deep learning approach since deep learning models have recently had success in modeling complex dependencies in NLP. More specifically, we apply recurrent neural networks (RNNs) (Elman, 1990; Jordan, 1986; Werbos, 1990) due to their success on complex NLP tasks like machine translation and reasoning.

To apply RNNs, we order the cells of the table into a sequence as indicated in Figure 4 and label – or "fill" – the cells one by one in the order of the sequence. We call this approach *table filling*.

More specifically, we use a bidirectional architecture (Vu et al., 2016b), a forward RNN and a backward RNN, to fill each cell (i, j) as shown in Figure 3. The forward RNN provides a representation of the history $w_1, \ldots, w_i$. The backward network provides a representation of the following context $w_j, \ldots, w_{|s|}$. The figure shows how the named entity tag for "Association" is computed. The forward RNN is shown as the sequence at the bottom. h_{f_i} is the representation of the history and h_{b_j} is the representation of the following context. Both are fed into $h_{i,j}$ which then predicts the label L-ORG. In this case, $i = j$. The prediction of a relation label is similar, except that in that case $i \neq j$.

[2] Relation types (excluding $\perp$) exist only in the word pairs with entity types: (L-*, L-*), (L-*, U-*), (U-*, L-*) or (U-*, U-*), where * indicates any entity type encoded in BILOU (Begin, Inside, Last, Outside, Unit) scheme.

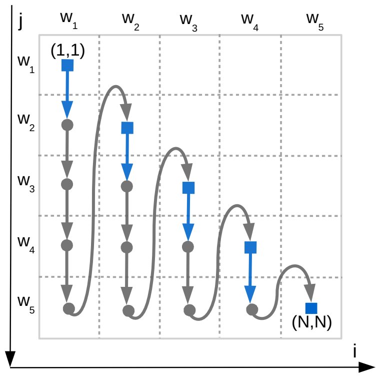


Figure 4: Table Filling/Decoding Order. Filled squares in blue represent both entity and relation label assignments, while filled circles in gray represent only relation label assignments, analogous to entries in Table 1. (i, j) is the cell index in the table, where i and j are the word indices in the sequence.

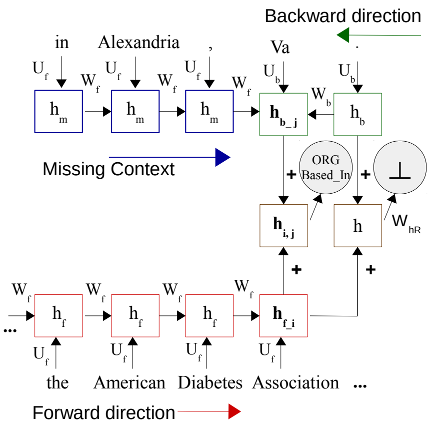


Figure 5: The context-aware TF-MTRNN model. (...) indicates the remaining word pair compositions (Table 1).

Our proposed RNN based framework jointly models the entity and relation extraction tasks to learn the correlations between them, by sharing the model parameters and representations. As illustrated in Figure 3, we use two separate output nodes and weight matrices each for entity and relation classification. An entity label is assigned to a word, while a relation is assigned to a word pair; therefore, EE is performed only when the same words from forward and backward networks are composed.

Dynamics of the proposed TF-MTRNN architecture (Figure 3) are given below:

$$\begin{aligned} s_{R_{i,j\in i:N}} = g(W_{hR}h_{i,j}); \qquad s_{E_{i,j=i}} = g(W_{hE}h_{i,j}); \qquad h_{i,j} = h_{f_i} + h_{b_j} \\ h_{f_i} = f(U_f w_i + W_f h_{f_{i-1}}); \qquad h_{b_j} = f(U_b w_j + W_b h_{b_{j+1}}) \end{aligned} \tag{1}$$

where i and j are the time-steps of forward and backward networks, respectively. ith word in the sequence is combined with every jth word, where $j = i, ..., N$ (i.e. combined with itself and the following words in the sequence). N is the total number of words in the sequence. For a given sequence, $s_{R_{i,j}}$ and $s_{E_{i,j}}$ represent the output scores of relation and entity recognition for ith and jth word from forward and backward networks, respectively. Observe that EE is performed on the combined hidden representation $h_{i,j}$, computed from the composition of representations of the same word from forward and backward networks, therefore $i = j$ and resembling the diagonal entries for entities in Table 1. h_{f_i} and h_{b_j} are hidden representations of forward and backward networks, respectively. W_{hR} and W_{hE} are weights between hidden layers ($h_{i,j}$) and the output units of relation and entity, respectively. f and g are activation and loss functions. Applying $argmax$ to $s_{R_{i,j\in i:N}}$ and $s_{E_{i,j=i}}$ gives corresponding table entries for relations and entities, in Table 1 and Figure 4.

2.3 Context-aware TF-MTRNN model

In Figure 3, we observe that when hidden representations for the words *Association* and *Va* are combined, the middle context i.e. all words in the sequence occurring between the word pair in composition are missed. Therefore, we introduce a third direction in the network (Figure 5) with missing context (i.e. *in Alexandria ,*) to accumulate the full context in combined hidden vectors ($h_{i,j}$).

Dynamics of the context-aware TF-MTRNN is similar to Eq. 1, except h_{b_j}, in Figure 5:

$$\begin{aligned} h_{b_j} = f(U_b w_j + W_b h_{b_{j+1}} + U_f h_{m_{t=T}}) \\ h_{b_{j+1}} = f(U_b w_{j+1} + W_b h_{b_{j+2}}); \qquad h_{m_t} = f(U_f w_t + W_f h_{m_{t-1}}) \end{aligned} \tag{2}$$

where h_{b_j} is the hidden representation in backward network obtained from the combination of jth word and contexts from backward network and from missing direction, $t = (i + 1, ..., T = j - 1)$, where i and j are the time-steps for forward and backward networks, respectively. $h_{m_{t=i}}$ is initialized with zeros similar to forward and backward networks. There is no missing context when $i = 0$ and $j = 0$ i.e. w_t is NULL and therefore, we introduce an artificial word *PADDING* and use its embedding to initialise w_t.

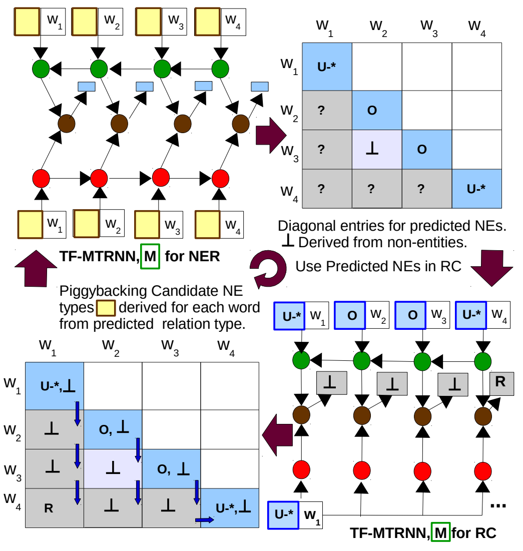
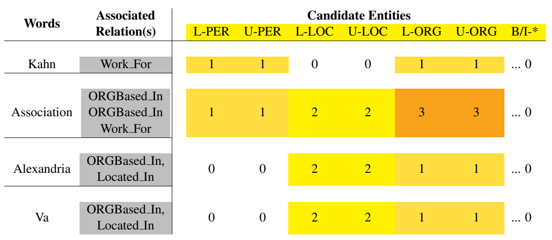

TF-MTRNN, M for NER
Diagonal entries for predicted NEs.
⊥ Derived from non-entities.
Use Predicted NEs in RC
Piggybacking Candidate NE types derived for each word from predicted relation type.
TF-MTRNN, M for RC
U-*
O
?
⊥
R
U-*,⊥
O, ⊥

Figure 6: End-to-End Relation Extraction by jointly modeling entity and relation in a unified Multi-task RNN framework, M (TF-MTRNN) and filling an Entity-Relation table. Entity-relation interdependencies modeled by parameter sharing and piggybacking (Section 2.4 and Figure 7). NE: Named Entity; U-* and O: NE in BILOU format; ?:Relation to determine.



Words	Associated Relation(s)	Candidate Entities						
		L-PER	U-PER	L-LOC	U-LOC	L-ORG	U-ORG	B/I-*
Kahn	Work_For	1	1	0	0	1	1	... 0
Association	ORGBased_In ORGBased_In Work_For	1	1	2	2	3	3	... 0
Alexandria	ORGBased_In, Located_In	0	0	2	2	1	1	... 0
Va	ORGBased_In, Located_In	0	0	2	2	1	1	... 0

Figure 7: Piggybacking approach to model label dependencies from relations to entities. We do not list all words due to space limitation. * indicates any entity type. Highlight for counts indicate candidate entity importance for corresponding words.

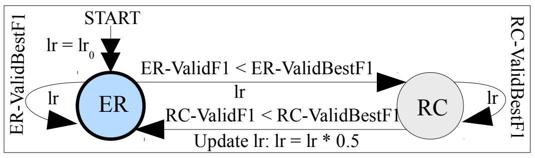


Figure 8: State Machine driven Multi-task Learning. ER:Entity Recognition; RC:Relation Classification; lr:learning rate; $ER-ValidBestF1$:Best entity recognition F_1 score on validation set.

2.4 Piggybacking for Entity-Relation Label Dependencies

Having named-entity labels is very informative for finding the relation type between them, and vice versa having the relation type between words eases problem of named-entity tagging. We model these label interdependencies during the end-to-end relation extraction in Figure 6, where the input vector at time step, t is given by -

$$input_t = \{C_{RE}, E_{ER}, W_{emb}\} \tag{3}$$

where C_{RE} is the count vector to model relation to entity dependencies, E_{ER} is the one-hot vector for predicted entities to model entity to relation dependencies and W_{emb} is the word embedding vector. Therefore, the input vector at each time step, t is the concatenation of these three vectors.

To model *entity to relation* dependency, the TF-MTRNN model, M for NER (Figure 6) first computes entity types, which are represented by diagonal entries of entity-relation table. Each predicted entity type E_{ER} (filled blue-color boxes) is concatenated with its corresponding word embedding vector W_{emb} and then input to the same model, M for relation classification.

To model *relation to entity* dependency, we derive a list of possible candidate entity tags for each word participating in a relation(s), except for $\perp$ relation type. Each word associated with a relation type(s) is determined from relation classification (RC) step (Figure 6). Figure 7 illustrates the entity type count vector for each word of the given sentence (Figure 1). For example, the word *Alexandria* participates in the relation types: *ORGBased_In* and *Located_In*. Possible entity types are {*U-ORG*, *L-ORG*, *U-LOC*, *L-LOC*} for *ORGBased_In*, while {*U-LOC*, *L-LOC*} for *Located_In*. We then compute a count vector C_{RE} from these possible entity types. Therefore, *U-LOC* and *L-LOC* each with occurrence 2, while *U-ORG* and *L-ORG* each with occurrence 1 (Figure 7). The candidate entity types as count vector (filled-yellow color box) for each word is piggybacked to model, M for entity learning by concatenating it with corresponding word embedding vector W_{emb}. This simple approach of piggybacking the count vectors of candidate entities enables learning label dependencies from *relation to entity* in order to improve entity extraction. In addition, multi-tasking by sharing parameters and adapting shared embeddings within a unified network enables learning label interdependencies.

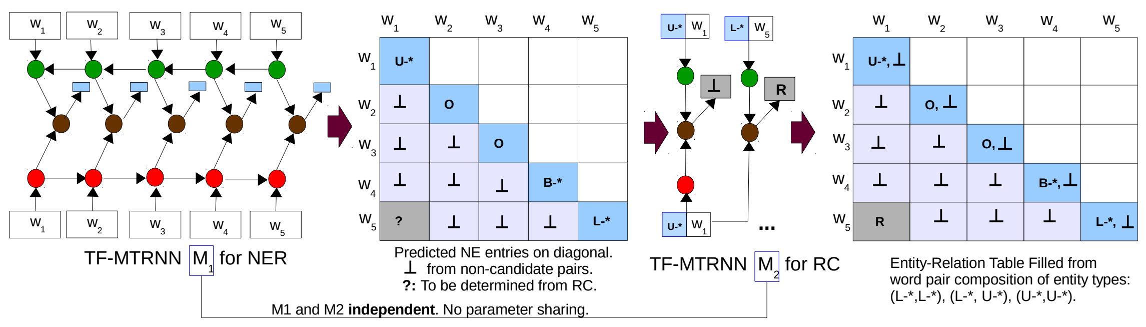

TF-MTRNN M1 for NER
Predicted NE entries on diagonal.
⊥ from non-candidate pairs.
?: To be determined from RC.
TF-MTRNN M2 for RC
Entity-Relation Table Filled from word pair composition of entity types: (L-*,L-*), (L-*, U-*), (U-*,U-*).
M1 and M2 independent. No parameter sharing.

Figure 9: Pipeline Approach in End-to-End Relation Extraction.

2.5 Ranking Bi-directional Recurrent Neural Network (R-biRNN)

Ranking loss has been used in neural architectures (dos Santos et al., 2015) and (Vu et al., 2016b) to handle *artificial* classes. In our experiments, for a given sentence x with class label y^+, the competitive class c^- is chosen the one with the highest score among all competitive classes during SGD step. The basic principle is to learn to maximize the distance between the true label y^+ and the best competitive label c^- for a given data point x. We use the ranking loss to handle the two artificial classes i.e. 'O' and $\perp$ in entity and relation types, respectively. The ranking objective function is defined as-

$$\begin{aligned} L = \log(1+\exp(\gamma(m^+ - s_\theta(x)_{y^+}))) + \log(1+\exp(\gamma(m^- + s_\theta(x)_{c^-})));\\ c^- = \arg\max_{c\epsilon C; c\neq y^+} s_\theta(x)_c \end{aligned} \quad (4)$$

where $s_\theta(x)_{y^+}$ and $s_\theta(x)_{c^-}$ are the scores for positive y^+ and the most competitive c^- classes. γ controls the penalization of the prediction errors while hyperparameters m^+ and m^- are the margins for the true and competitive classes. We set $\gamma = 2, m^+ = 2.5, m^- = 0.5$, following (Vu et al., 2016b).

The unified architecture (Figure 3) can be viewed as being comprised of two individual models, each for NER and RE (Figure 6). We illustrate that the R-biRNN (Figure 12 in Appendix A) is integrated in TF-MTRNN (Figure 3) and therefore, the unified model leverages R-biRNN (Vu et al., 2016b) effectiveness for entity extraction, where the full context information is availed from the forward and backward network at each input word vector along with the ranking loss at each output node. Figure 12 corresponds to the diagonal entries for named entities in Table 1 and enables entity-entity label dependencies (Miwa and Sasaki, 2014) via sequential learning.

3 Model Training

3.1 End-to-End Relation Extraction

In CoNLL04, more than 99% of the whole word pairs lie in the no_relation class. Therefore, named-entity candidates are required to choose the candidate word pairs in relation learning. In Figure 6 and Figure 9, we demonstrate the joint and pipeline approach for end-to-end relation extraction.

In Figure 6, the candidate relation pairs are chosen by filtering out the non-entities pairs. Therefore, in entity-relation table, we insert 'no_relation' label for the non-entities pairs and RC is not performed. Note that a word pair is chosen for RC in which at least one word is an entity. It allows the model M to correct itself at NER by piggybacking candidate named entities (Figure 7). In addition, it reduces a significant number of non-relation word pairs and does not create a bias towards the no_relation class. However, in Figure 9, the two independent models, M$_1$ and M$_2$ are trained for NER and RC, respectively. In pipeline approach, the only relation candidates are word pairs with (U-*, U-*),(L-*, L-*) or (U-*, L-*) entity types. Therefore, only w$_1$ and w$_5$ from word sequence are composed in M$_2$ for RC subtask.

		CoNLL04 Dataset					
	Features	NER			RE		
		P	R	F_1	P	R	F_1
separate	basic	.865	.902	.883	.360	.403	.376
separate	+POS	.877	.906	.892	.440	.376	.395
separate	+CF	.906	.914	.910	.454	.390	.410
separate	+CTX				.499	.434	.453
pipeline	basic				.641	.545	.589
pipeline	+POS				.663	.555	.604
pipeline	+CF				.661	.585	.621
pipeline	+CTX				.736	.616	.671
joint	basic	.885	.889	.888	.646	.531	.583
joint	+POS	.904	.908	.906	.673	.531	.594
joint	+CF	.913	.914	.914	.691	.562	.620
joint	+CTX				.745	.595	.661
joint	+p'backing	.925	.921	.924	.785	.630	.699
joint	+ensemble	.936	.935	.936	.832	.635	.721

Figure 10: CoNLL04 dataset: Performance on test set for NER and RE; RE in pipeline always used predicted NEs. POS: part-of-speech; CF: capital features; CTX: context awareness (Figure 5); p'backing: piggybacking predicted and candidate entities in RE and NER, respectively; ensemble: majority vote.

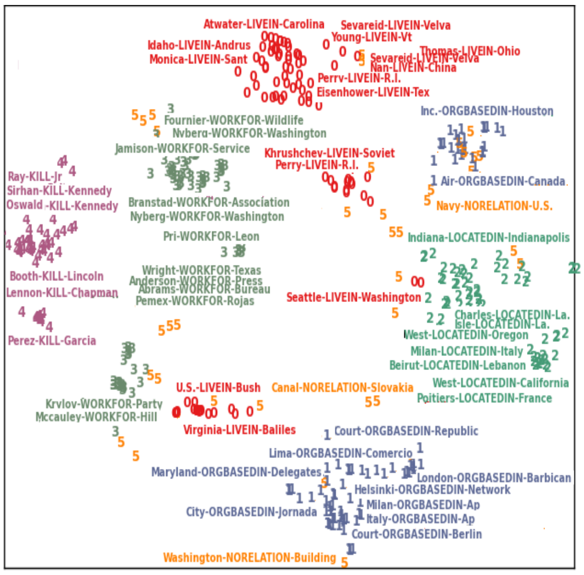


Figure 11: T-SNE view of the semantic entity-relation space for the combined hidden representations of each word pair composition. Relations: (0:*LIVEIN*, 1:*ORGBASEDIN*, 2:*LOCATEDIN*, 3:*WORKFOR*, 4:*KILL*, 5:*NORELATION*). Entity-pair and relation denoted by E1-RELATION-E2 and/or count in [0-5]. 5: misclassified entity-pairs.

3.2 Word Representation and Features

Each word is represented by concatenation of pre-trained 50-dimensional word embeddings[3] (Turian et al., 2010) with N-gram, part-of-speech (POS), capital feature (CF: all-capitalized; initial-capitalized) and piggybacked entity vectors (Section 2.4). The word embeddings are shared across entity and relation extraction tasks and are adapted by updating them during training. We use 7-gram $(w_{t-3}w_{t-2}w_{t-1}w_tw_{t+1}w_{t+2}w_{t+3})$ obtained by concatenating corresponding word embeddings.

3.3 State Machine driven Multi-tasking

Multi-task training is performed via switching across multiple tasks in a block of training steps. However, we perform switches between ER and RC subtasks based on the performance of each task on the common validation set and update learning rate only when task is switched from RC to ER (Figure 8). ER is the task to start for multi-tasking and ER/RC is switched in the following training step, when their $ValidF1$ score is not better than $BestValidF1$ score of previous steps on the validation set.

4 Evaluation and Analysis

4.1 Dataset and Experimental Setup

We use CoNLL04[4] corpus of Roth and Yih (2004). Entity and relation types are shown in Figure 2. There are 1441 sentences with at least one relation. We randomly split these into training (1153 sentences) and test (288 sentences), similar to Miwa and Sasaki (2014). We release this train-test split at `https://github.com/pgcool/TF-MTRNN/tree/master/data/CoNLL04`. We introduce the pseudo-label $\perp$ "no_relation" for word pairs with no relation.

To tune hyperparameters, we split (80-20%) the training set (1153 sentences) into $train$ and validation (dev) sets. All final models are trained on $train$+dev. Our evaluation measure is F_1 on entities and relations. An entity is marked correct if NE boundaries and entity type[5] are correct. A relation for a word pair is marked correct if the NE boundaries and relation type are correct. However, in separate approach, a relation for a word pair is marked correct if the relation type is correct.

[3]with a special token PADDING. Also, used when there is no missing context.

[4]conll04.corp at `cogcomp.cs.illinois.edu/page/resource_view/43`

[5]For multi-word entity mention, an entity is marked correct if atleast one token is tagged correctly.

	Roth&Yih			Kate&Mooney			Miwa&Sasaki			TF-MTRNN		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Person	.891	.895	.890	.921	.942	.932	.931	.948	.939	.932	.988	.959
Location	.897	.887	.891	.908	.942	.924	.922	.939	.930	.974	.956	.965
Organization	.895	.720	.792	.905	.887	.895	.903	.896	.899	.873	.939	.905
(Average)	.894	.834	.858	.911	.924	. 917	.919	.927	.923	.926	.961	.943
Live_In	.591	.490	.530	.664	.601	.629	.819	.532	.644	.727	.640	.681
OrgBased_In	.798	.416	.543	.662	.641	.647	.768	.572	.654	.831	.562	.671
Located_In	.539	.557	.513	.539	.557	.513	.821	.549	.654	.867	.553	.675
Work_For	.720	.423	.531	.720	.423	.531	.886	.642	.743	.945	.671	.785
Kill	.775	.815	.790	.775	.815	.790	.933	.797	.858	.857	.894	.875
(Average)	.685	.540	.581	.672	.607	.622	.845	.618	.710	.825	.664	.737

Table 2: State-of-the-art comparison for EE and RE on CoNLL04 dataset.

4.2 Results

Figure 10 shows results for NER[6] and RE. All models use n-grams for $n = 7$ (Section 3.2). Embedding dimensionality is 50. The notation "+" (e.g., +POS) at the beginning of a line indicates that the model of this line is the same as the model on the previous line except that one more model element (e.g., POS) is added. The separate NER model performs NER only. The separate RE model performs RE only, without access to NER results. The pipeline RE model takes the results of the separate NER model and then performs RE. The joint model is trained jointly on NER and RE. For compactness, we show the results of *two different models* (an NER model and an RE model) in the separate part of the table; in contrast, results for a *single model* – evaluated on both NER and RE – are shown in the joint part.

We make the following observations based on Figure 10. (i) All of our proposed model elements (POS, CF, CTX, piggybacking, ensemble) improve performance, in particular CTX and piggybacking provide large improvements. (ii) Not surprisingly, the pipeline RE model that has access to NER classifications performs better than the separate RE model. (iii) The joint model performs better than separate and pipeline models, demonstrating that joint training and decoding is advantageous for joint NER and RE. (iv) Majority voting[7] (ensemble) results in a particularly large jump in performance and in the overall best performing system; F_1 is .936 for NER and .721 for RE, respectively.

4.3 Comparison with Other Systems

Our end-to-end relation extraction system outperform the state-of-the-art results. We compare the entity and relation extraction performance of our model with other systems (Roth and Yih, 2007; Kate and Mooney, 2010; Miwa and Sasaki, 2014). (Roth and Yih, 2007) performed 5-fold cross validation on the complete corpus (1441 sentences), while (Miwa and Sasaki, 2014) performed 5-cross validation on the data set, obtained after splitting the corpus. We report our results on the test set from random split (80-20%) of the corpus, similar to (Miwa and Sasaki, 2014). Since, the standard splits were not available, we cannot directly compare the results, but our proposed model shows an improvement of 2.0% and 2.7% in F_1 scores for entity and relation extraction tasks, respectively (Table 2).

[6]Our NER model reports 86.80% F1 score, comparable to 86.67% from (Lample et al., 2016) on CoNLL03 shared task using the standard NER evaluation script with strict multi-word entity evaluation, and adapted for BILOU encoding.

[7]Randomly pick one of the most frequent classes, in case of a tie

4.4 Word pair Compositions (T-SNE)

Using t-SNE (der Maaten and Hinton, 2008), we visualize the hidden representations obtained on the composition of hidden vectors of every two words (word pair) in the sentence via TF-MTRNN model. In Figure 11, we show all data points i.e. word pair compositions, leading to natural relations (except $\perp$ denoted by 5). We observe that the entity mention pairs with common relation types form clusters corresponding to each relation in the semantic entity-relation space. We observe that the relation clusters with common entity type lie close to each other, for example, *KILL* has (*PER*, *PER*) entity pairs, which is close to relation cluster *LIVEIN* and *WORKFOR*, in which one of the entities i.e. *PER* is common. While, *KILL* relation cluster is at a distance from *LOCATEDIN* cluster, since they have no common entity.

4.5 Hyperparamter Settings

We use stochastic gradient descent with L2 regularization with a weight of .0001. The initial learning rate for entity and relation extraction is .05 with hidden layer size 200. The learning rate update and task switching is driven by the state machine (Figure 8). Models are trained for 40 iterations performing stochastic gradient descent. We initialize the recurrent weight matrix to be identity and biases to be zero. We use Capped Rectified Linear units (CappedReLu) and ranking loss with default parameters (section 2.5). The entity vectors C_{RE} and E_{ER} are initialized with zero when NER is performed for the first time in entity and relation extraction loop (Figure 6). The models are implemented in Theano (Bergstra et al., 2010; Bastien et al., 2012).

5 Related Work

Recurrent and convolutional neural networks (Zeng et al., 2014; Nguyen and Grishman, 2015; Zhang and Wang, 2015; Vu et al., 2016a) have delivered competitive performance for sentence-level relation classification. Socher et al. (2012) and Zhang and Wang (2015) proposed recurrent/recursive type neural networks to construct sentence representations based on dependency parse trees. However, these sentence-level state-of-the-art methods do not model the interdependencies of entity and relation, do not handle multiple relation instances in a sentence and therefore, can not detect entity mention pairs for the sentence-level relations. Our approach is a joint entity and word-level relation extraction capable to model multiple relation instances, without knowing nominal pairs.

Existing systems (Roth and Yih, 2004; Kate and Mooney, 2010; Miwa and Sasaki, 2014) are complex feature-based models for joint entity and relation extraction. The most related work to our method is (Miwa and Sasaki, 2014); however they employ complex search heuristics (Goldberg and Elhadad, 2010; Stoyanov and Eisner, 2012) to fill the entity-relation table based on structured prediction method. They explicitly model the label dependencies and their joint approach is not based on neural networks. Multi-task learning (Caruana, 1998) via neural networks (Zhang and Yeung, 2012; Seltzer and Droppo, 2013; Dong et al., 2015; Li and J, 2014; Collobert and Weston, 2008) have been used to model relationships among the correlated tasks. Therefore, we present a unified neural network based multi-task framework to model the entity-relation table for end-to-end relation extraction.

6 Conclusion

We proposed TF-MTRNN, a novel architecture that jointly models entity and relation extraction, and showed how an entity-relation table is mapped to a neural network framework that learns label interdependencies. We introduced word-level relation classification through composition of words; this is advantageous in modeling multiple relation instances without knowing the corresponding entity mentions in a sentence. We also introduced context-awareness in RNN network to incorporate missing information, and investigated piggybacking approach to model entity-relation label interdependencies.

Experimental results show that TF-MTRNN outperforms state-of-the-art method for both entity and relation extraction tasks.

Acknowledgements
This research was supported by Siemens AG CT MIC - Machine Intelligence, Munich Germany.

References

Frèdèric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.*

James Bergstra, Olivier Breuleux, Frèdèric Bastien, Pascal Lamblin, Guillaume Desjardins Razvan Pascanu, Joseph Turian, David Warde-Farley, , and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. *In Proceedings of the Python for Scientific Computing Conference (SciPy).*

Rich Caruana. 1998. Multitask learning. *Springer.*

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing:deep neural networks with multitask learning. *In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland.*

L Van der Maaten and G Hinton. 2008. Visualizing data using t-sne. *Proceedinggs of JMLR.*

Daxiang Dong, Wei He Hua Wu, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *In Proceedings of the Association for Computational Linguistics.*

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2).

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

M. Jordan. 1986. Serial order: A parallel distributed processing approach. *Published in Tech. Rep. No. 8604. San Diego: University of California, Institute for Cognitive Science.*

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. *In Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition.

Qi Li and Heng J. 2014. Incremental joint extraction of entity mentions and relations. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 402–412.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1858–1869.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. *In Proceedings of the NAACL Workshop on Vector Space Modeling for NLP.*

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. *In Hwee Tou Ng and Ellen Riloff, editors, HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8.

Dan Roth and Wen-Tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *In Hwee Tou Ng and Ellen Riloff, editors, HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004).*

Michael L Seltzer and Jasha Droppo. 2013. Multi-task learning in deep neural networks for improved phoneme recognition. *In Proceedings of the IEEE International Confonference on Acoustics, Speech, and Signal Processing (ICASSP).*

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. In proceedings of emnlp/conll. *Association for Computational Linguistics.*

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. *In Proceedings of COLING 2012*, page 25192534.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016a. Combining recurrent and convolution neural networks for relation classification. *In Proceedings of the NAACL*.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016b. Bi-directional recurrent neural network with ranking loss for spoken language understanding. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*.

Paul J Werbos. 1990. Backpropagation through time:what it does and how to do it. *In Proceedings of the IEEE*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING*.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *In ArXiv*.

Y. Zhang and D.-Y. Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536*.

Appendix A. R-biRNN discussed in section 2.5.

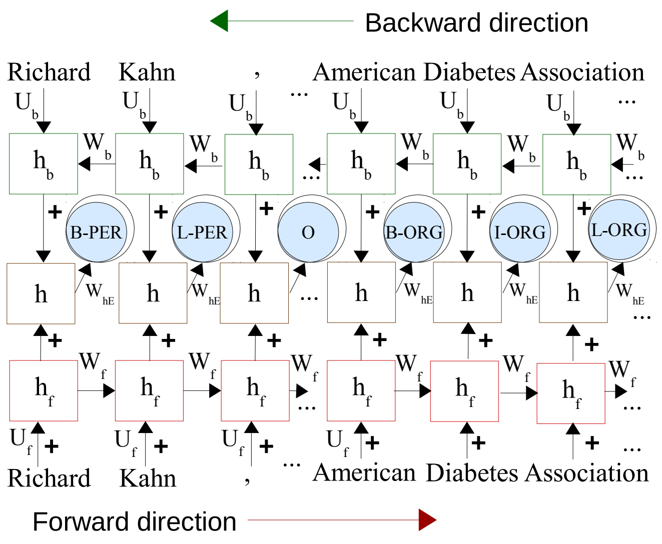


Figure 12: R-biRNN. Disintegrating TF-MTRNN (Figure 3) to illustrate that it is comprised of R-biRNN for entity learning. (...) indicates remaining words in the sentence (Figure 1).