CONVOLUTIONAL NEURAL NETWORK BASED TRIANGULAR CRF FOR JOINT INTENT DETECTION AND SLOT FILLING

Puyang Xu, Ruhi Sarikaya

Microsoft Corporation

ABSTRACT

We describe a joint model for intent detection and slot filling based on convolutional neural networks (CNN). The proposed architecture can be perceived as a neural network (NN) version of the *triangular* CRF model (TriCRF), in which the intent label and the slot sequence are modeled jointly and their dependencies are exploited. Our slot filling component is a *globally* normalized CRF style model, as opposed to left-to-right models in recent NN based slot taggers. Its features are automatically extracted through CNN layers and shared by the intent model. We show that our slot model component generates state-of-the-art results, outperforming CRF significantly. Our joint model outperforms the standard TriCRF by 1% absolute for both intent and slot. On a number of other domains, our joint model achieves 0.7 - 1%, and 0.9 - 2.1% absolute gains over the independent modeling approach for intent and slot respectively.

Index Terms— Joint modeling, slot filling, convolutional neural network, triangular CRF

1. INTRODUCTION

Spoken language understanding (SLU) typically involves identifying a user's intent and extracting relevant semantic constituents from the natural language sentence – a task often called *slot filling*. While intent detection is a standard classification problem in which only one label is predicted for each sentence, slot filling is often formulated as a sequence labeling task, where a sequence of labels need to be assigned jointly.

Intent detection and slot filling are usually carried out separately. A number of standard classifiers can be used for intent detection, such as logistic regression and support vector machines. For slot filling, conditional random field (CRF) [1] is a proven technique and has been used extensively. As opposed to the left-to-right, factorized probabilistic models for sequence labeling, such as maximum entropy markov models (MEMM) [2], CRF directly models the global conditional distribution, and is often thought to overcome the *label bias* problem [1] facing locally normalized models.

We thank Anoop Deoras and Minwoo Jeong for sharing the ATIS datasets, and Ashley Fidler for preparing the other datasets.

Joint training of intent and slot models has been investigated in the literature. In [3], *triangular* CRF (TriCRF) was proposed for this purpose: On top of the standard CRF, an additional random variable indicating the topic/intent assignment of the sentence is introduced, with dependency links established between the added variable and the hidden label sequence. It was shown empirically that exploiting such dependencies can lead to improved results for both of the modeling tasks.

Similar to most classification techniques, TriCRF requires feature functions to be predefined by human before the model can be trained. Inspired by the success of neural network (NN) based deep architectures for various tasks, the goal of this work is to exploit the powerful feature learning mechanism enabled by NNs, and apply it to joint intent and slot modeling for SLU.

NN based techniques have been previously explored for SLU. In [4], intent detection (call routing) was shown to benefit substantially from the use of a multi-layer NN architecture. Recently, several NN based models were introduced for slot filling, either using feed-forward NNs [5] or recurrent NNs [6, 7]. However, no joint training was discussed in these work. In fact, due to the left-to-right, locally normalized model nature, it is not straightforward to adapt the recently proposed NN based slot models to simultaneously handle intent detection.

In this work, we first describe a *globally normalized* NN architecture for slot filling. The benefit of global normalization is two-fold: First, it overcomes the label bias problem, which is often perceived to be a weakness of local normalization. Second, since the sequence of tags are predicted in a single step, it allows other labels such as intent to be predicted simultaneously. The same features extracted through convolutional neural networks (CNN) can thus be shared by the two modeling tasks. To the best of our knowledge, this is the first attempt to train intent and slot models jointly based on NNs, and can be thought of as a continuous space version of the TriCRF model.

NN based CRF has been investigated in the literature. In [8, 9], conditional neural field and neural CRF were introduced which combined the NN based feature learning and the CRF models. Similar approaches have also been used for phone recognition with success [10, 11]. However, the fea-

tures were not derived from variable length word sequences. The experiments they presented were also limited to using linear chain CRF models, as opposed to TriCRF in this work. In [12], a multi-layer CRF was used for sequence labeling, but the feature extraction was also different, and no joint modeling was performed. Another related work is the multi-task learning framework presented in [13]. It also uses convolutional units for feature extraction. However, the task of sequence labeling was reduced to a set of independent local classification tasks (similar to 0^{th} order CRF), in which no dependency is assumed to be present among the hidden tags. In contrast, our architecture handles sequence labeling as a full blown CRF model, and also allows handling other tasks simultaneously.

2. GLOBALLY NORMALIZED NN FOR SLOT FILLING

2.1. Local and Global Normalization

Discriminative models for sequence labeling attempts to model the conditional distribution of the hidden sequence Y given the observation X. In a locally normalized model, $P(Y|X)$ is factorized into the product of a sequence of local probabilities, as shown in (1),

$$P(Y|X) = \prod_{i=1}^{l} P(Y_i|X, Y_1, ..., Y_{i-1}), \quad (1)$$

where Y_i is the i^{th} observation in the sequence of length l.

There are different ways to parameterize each $P(Y_i|X)$, such as the maximum entropy models in MEMM, and the various kinds of NN models in the recently proposed slot filling approaches [5, 6, 7]. Despite the notorious label bias problem for such local models, the recent NN based local models have uniformly achieved superior slot tagging results.

As opposed to modeling local probabilities, globally normalized models directly specify the global conditional distribution, usually in an exponential framework,

$$P(Y|X) = \frac{e^{\sum_j f_j(X,Y)\theta_j}}{\sum_{Y'} e^{\sum_j f_j(X,Y')\theta_j}}. \quad (2)$$

Here, $f_j(X, Y)$ is the j^{th} feature function defined over the observation X and the hidden sequence Y, and θ_j is the corresponding feature weight. The denominator sums over all possible hidden sequences.

Many loss functions for optimization can be applied according to the probability definition in (2). The likelihood loss is convex thus can be minimized using gradient-based techniques. The gradient with respect to each θ_j usually requires computing the posterior probabilities $P(Y_i = y|X)$ or $P(Y_i = y, Y_{i-1} = y'|X)$, and these can be efficiently computed through forward-backward passes over the hidden trellis.

2.2. Feature Learning Through CNN

Note that in (2), each $f_j(Y, X)$ needs to be predefined. It is usually a binary function and describes a certain pattern (e.g. n-gram pattern) in the sequence. In our proposed architecture, the score for each sequence, instead of based on a set of predefined feature functions, will be derived from a CNN architecture over the word sequence.

CNN has been used extensively for various machine learning tasks. It is a way of controlling the number of parameters and capturing some of the translation invariance in the input data. CNN was previously used for extracting features over variable length word sequences in [13] (Figure 1). Similar as other NN based natural language processing techniques, each word in the vocabulary, instead of being treated as a discrete symbol, is represented as a fixed dimensional continuous vector. Such vectors are learned in the same way as other NN parameters, and can be initialized either randomly or using vectors obtained through auxiliary tasks such as language modeling.

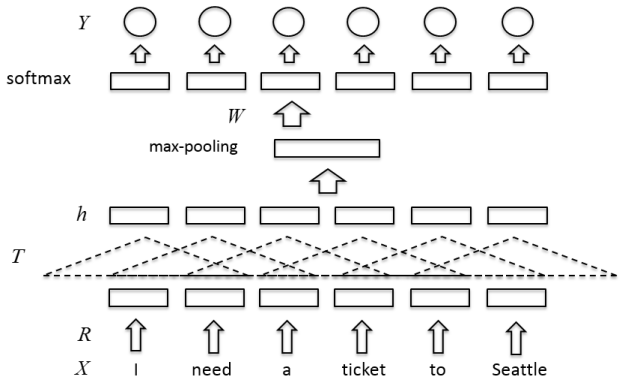

Fig. 1. The CNN architecture introduced in [13]. Note that each tag in Y is predicted separately through *softmax*.

As shown in the Figure 1, the word sequence is represented as the concatenated word vectors. Such word vectors may also encode positional information indicating which word in the sequence the tag will be predicted for. The feature transform T spans over a fix-sized n-gram window and slides over the input sequence. The obtained features usually undergo some kind of nonlinear activation function such as *tanh* before feeding into subsequent layers. In [13], a max-pooling layer was also used to reduce the number of parameters and ensure the feature layers have fixed dimensions such that classic NN layers can be applied on the top.

Note that in Figure 1, the final feature layer after max-pooling goes into an array of *softmax* output layers, each producing a distribution over tags for the current word position. This approach essentially follows the local modeling strategy illustrated in (1), and no dependency is assumed between adjacent tags.

In contrast, in our proposed sequence labeling architecture (Figure 2), no normalization is performed at each individual position. In fact, the top layer is essentially the same as a 1^{st} order CRF model – the only difference is that the features from the word sequence are automatically extracted continuous-valued features, instead of predefined indicator functions.

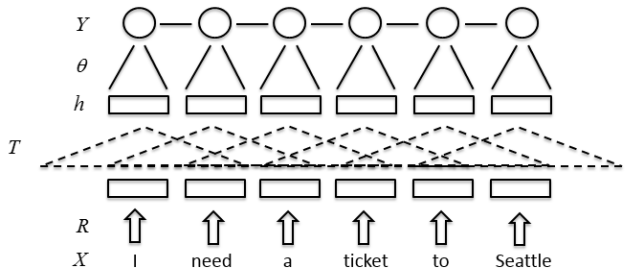

Fig. 2. Globally normalized conditional model based on CNN. The top layer is essentially the same as CRF.

Similar as (2), the global conditional distribution can be written as

$$P(Y|X) = \frac{e^{\sum_i \left(t(Y_{i-1},Y_i)+\sum_j h_{ij}(X_i,R,T)\theta_j(Y_i)\right)}}{\sum_{Y'} e^{\sum_i \left(t(Y'_{i-1},Y'_i)+\sum_j h_{ij}(X_i,R,T)\theta_j(Y'_i)\right)}}. \quad (3)$$

Note that the total score of the sequence is factorized into the sum of scores at each word position i. $t(Y_{i-1}, Y_i)$ is the tag transition score from Y_{i-1} to Y_i. $h_{ij}(X_i, R, T)$ denotes the j^{th} element in the feature vector extracted out of the n-gram window centered at X_i, and $\theta_j(Y_i)$ is the corresponding feature weight associated with the tag Y_i. It is worth pointing out that the described framework can also incorporate external features such as entity and syntactic features – either by augmenting the word feature vectors or adding these features directly at the topmost layer of the architecture.

Training of the topmost layer is the same as training a standard CRF model. The bottom layers can be trained using the well-known back propagation algorithm. Remember that the back propagation is essentially the chain rule of derivatives. We can easily take the derivative of the loss function with respect to each $h_{ij}(X_i, R, T)$ at the top layer, and obtain the derivative with respect to T and R by applying the chain rule.

3. NN BASED TRIANGULAR CRF

3.1. Intent Detection Using CNN

The intent classification component in our proposed NN architecture (Figure 3) differs from the NN based intent classifier introduced in [4], in which case the word sequence is first converted into a feature vector (e.g. a n-gram count vector), upon which standard NN layers are built. In this work, convolutional layers are employed for feature extraction directly from the embedded word sequence.

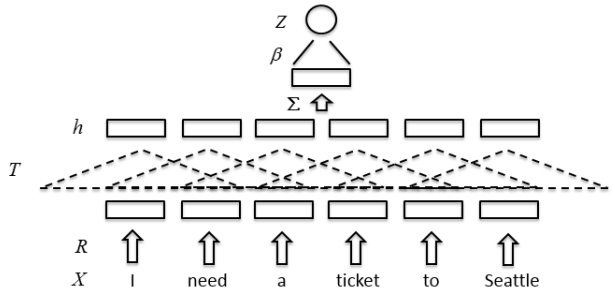

Fig. 3. CNN based intent classifier. The top layer is the same as standard linear classifiers such as logistic regression. The feature vectors in h are summed.

As depicted in Figure 3, the feature vectors extracted by the convolutional units at different word positions, are summed up before the intent type Z is predicted. The probability of each Z is specified according to (4),

$$P(Z|X) = \frac{e^{\sum_i \left(\sum_j h_{ij}\beta_j(Z)\right)}}{\sum_{Z'} e^{\sum_i \left(\sum_j h_{ij}\beta_j(Z')\right)}}. \quad (4)$$

Note that the max-pooling layer can also be used here instead of the summation. We have not observed much difference between the two approaches in terms of model accuracy.

3.2. Joint Slot Filling and Intent Detection

We can easily combine Figure 2 and Figure 3 into a single architecture. (Figure 4).

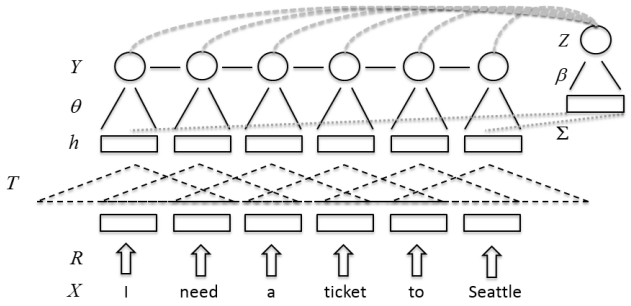

Fig. 4. Joint slot filling and intent detection based on CNN. The curly dashed lines at the top capture the dependency between intent and slot as in TriCRF.

Note that the convolutional layers and the resulting feature vectors h are shared by the two tasks. The global nature of our sequence labeling component allows the system

to simultaneously handle multiple labeling and classification tasks.

With the combined architecture, our model specifies the conditional distribution of Y and Z jointly,

$$P(Y,Z|X) = \frac{e^{\sum_i \left(t(Y_{i-1},Y_i)+g(Y_i,Z)+\sum_j(\theta_j(Y_i)+\beta_j(Z))h_{ij}\right)}}{Z(X)}. \quad (5)$$

Same as in TriCRF, the feature function $g(Y_i, Z)$ is used to model the interaction between intent and slot – it essentially acts as an intent dependent slot prior for the model. More detailed per-intent modeling is possible within TriCRF, such as adding intent independent slot transition features and slot emission features, in which case learning can become challenging given limited amounts of training data. The denominator $Z(X)$ is the normalizer and sums over all possible combinations of Y and Z.

Similar as before, the top layer of the proposed architecture can be trained in the same way as the standard TriCRF, and the layers below can be trained using the back propagation. As described in [3], the posterior probability $P(Y, Z|X)$ can be written as $P(Z|X)P(Y|X, Z)$. Both of these quantities can be obtained from multiple forward-backward passes through the hidden trellis – one pass for each Z. Therefore, the training complexity is theoretically $|Z|$ times larger than CRF models. Recall that the complexity for training CRF models is quadratic in the number of slot types. Since the number of observed slot types for each intent in the training data is usually much smaller than the total number of slots, we can achieve significant speedup if we allow only a subset of slot types to appear for each intent. We also observed no degradation of performance applying this constraint in our experiments. At test time, the best Y, Z combination can be obtained similarly by running multiple viterbi decoding passes.

4. EXPERIMENTAL RESULTS

4.1. Model Training

We use stochastic gradient descent to learn the model parameters. A development set is used to track the training progress. The learning rate starts at 0.05 and halves every time the result on the development set ceases to improve for 5 consecutive iterations. The training stops when the learning rate falls below 0.0001. All the NN parameters are *randomly* initialized as opposed to derived from other tasks.

We use the rectifier activation function at layer h, and also regularize the features using the dropout technique (with probability 0.5) [14].

The size of the word vectors, the hidden layers, and the convolutional n-gram windows are tuned minimally on the development set. Unless stated otherwise, we generally use 30 dimensional word vectors, and extract 110 dimensional hidden features from the 5-gram window centered around the current position. While additional features can be easily incorporated into our architecture, we only use lexical features (features automatically extracted from word sequences) for the experiments.

4.2. Slot Filling Results

The ATIS dataset has been used extensively in the SLU research community. To evaluate our slot filling model, we use the same ATIS corpus that was used in [5], which consists of 4978 utterances in the training set, and 893 utterances in the test set. We also have a set of 491 utterances as a development set.

We mainly compare with the three recently introduced NN based slot filling models [5, 6, 7]. While it is not straightforward for these models to simultaneously handle intent classification, they all produced the new state-of-the-art slot filling results in the literature. CRF results are also presented as it is often perceived to be a powerful sequence tagger.

Table 1. *F1 scores on ATIS compared with recent state-of-the-art NN techniques (lexical features only).*

Model	F1 score
CRF	91.00
DBN [5] (discriminative embedding)	93.20
RNN [6](ATIS embedding)	94.11
RNN [7] (Wikipedia embedding)	94.26
This work (random embedding)	**94.35**

The CRF result is obtained using CRF++ [15]. L2 regularization is used and the regularization weight is optimized on the development set. As shown in Table 1, the proposed NN architecture significantly outperforms the CRF model, and is also similar to the recent state-of-the-art approaches. It is worth mentioning that all the recent NN methods presented here used word embedding obtained from external tasks or resources, while our word vectors are initialized randomly.

4.3. Joint Modeling Results

The goal of this set of experiments is to compare with the standard TriCRF model and demonstrate the advantage of performing joint modeling within the proposed NN architecture.

We obtained the same dataset used in the original TriCRF paper [3]. It is also an ATIS corpus consisting of 5138 utterances with both slot and intent annotated. There are in total 21 types of intents and 110 types of slots. We use the same 10-fold cross validation setup as in [3]. Instead of training on 9 folds and testing on the remaining 1 fold, we leave another 1 fold out as the development set in order to track the training progress. The intent error rate and the slot F1 score averaged over the 10 folds are demonstrated in Table 2.

Table 2. *Averaged intent error rate and slot F1 score on ATIS compared with independent training and standard TriCRF (10-fold cross validation).*

	Intent Error	Slot F1
Indep training [3]	7.91	90.67
Indep training (CNN)	6.65	92.43
TriCRF [3]	6.93	94.42
CNN TriCRF	**5.91**	**95.42**

In addition to the independent training results reported in [3], which made use of the standard CRF models, we also present the independent training results using the CNN based individual component in our joint architecture. As indicated by the table, both the CNN based intent classifier and slot tagger outperform the standard models in [3] by significant margins. When joint training is performed, both intent and slot models benefit substantially. Moreover, the proposed CNN based TriCRF outperforms the standard TriCRF by 1% absolute on both tasks, although it is trained on one less fold of data.

Another favorable condition for the CNN based TriCRF is that it usually produces much more compact models. In the TriCRF model presented in [3], a large number of features are used for slot filling in each n-gram window – n-grams, skip n-grams, and also features such as prefix, suffix which are not directly used by our CNN based model. In our approach, all features in the n-gram window are succinctly represented as an automatically learned continuous vector. For intent classification, the same feature vector can be used, without requiring a new set of features to be defined as in the standard TriCRF.

4.4. More Domains From Real Applications

The benefit of joint training is quite obvious on the ATIS dataset. However, it was found in [16], that joint training may not always be advantageous (using the standard TriCRF), especially for real-world systems at early stages of the development. It was observed that while TriCRF produces slightly better slot filling results, the two-pass (independent) training yields significantly better intent classification accuracy. Therefore, the goal of this set of experiments, is to evaluate the efficacy of the proposed CNN based TriCRF on a number of datasets collected for real dialogue applications.

The baseline we are comparing with, is the independent training strategy consisting of support vector machine (SVM) based intent classifier, and the linear chain CRF based slot tagger (using CRF++). Only lexical features (n-grams) are used in the experiments.

We have datasets from 4 different domains (communication, calendar, alarm and notes), each containing around 10,000 utterances. The datasets are each partitioned into training, testing and development sets according to an 80/10/10 division.

Table 3. *Intent error rate and slot F1 score on 4 domains using CNN TriCRF compared with independent training using SVM and CRF (lexical feature only).*

Domain	Intent Error		Slot F1	
	Indep	CNN TriCRF	Indep	CNN TriCRF
Comm.	7.2	6.4	88.6	90.0
Calendar	7.2	6.4	85.9	88.0
Alarm	7.8	7.1	87.6	89.3
Notes	6.5	5.5	85.7	86.6

As Table 3 demonstrates, the advantage of CNN based TriCRF is very consistent across the 4 domains for both tasks, differing from the findings presented in [16]. We observe 0.7% - 1% absolute improvement in intent accuracy, and 0.9 - 2.1% absolute improvement in slot F1. The advantage here could be the result of two reasons, namely the CNN based modeling, and the joint training. Even if the dependency between intent and slot is not useful, in which case the joint training brings little benefit, the proposed architecture can benefit from the CNN based feature learning and yield superior results than the existing separate training strategy (SVM + CRF).

5. CONCLUSION

We proposed a NN version of the TriCRF model, which simultaneously handles intent detection and slot filling. The slot filling component is globally normalized. The features are extracted through CNN layers and shared by the two tasks. Experimental results showed that the slot filling component in our architecture is state-of-the-art, and the joint model outperforms TriCRF by 1% absolute for both intent and slot. We also applied the joint model to a number of real world domains and demonstrated consistent improvement over the existing independent modeling strategy.

6. REFERENCES

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

[2] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *ICML*, 2000, pp. 591–598.

[3] M. Jeong and G.G. Lee, "Triangular-chain conditional random fields," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 1287–1302, September 2008.

[4] R. Sarikaya, G.E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *ICASSP*, 2011.

[5] A. Deoras and R. Sarikaya, "Deep belief network based semantic taggers for spoken language understanding," in *INTERSPEECH*, 2013.

[6] K. Yao, J. Zweig, M. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *INTERSPEECH*, 2013.

[7] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *INTERSPEECH*, 2013.

[8] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," in *NIPS*, 2009.

[9] T.M.T. Do and T. Artieres, "Neural conditional random fields," *JMLR*, 2010.

[10] A. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *INTERSPEECH*, 2010.

[11] G. Andrew and J. Bilmes, "Sequential deep belief networks," in *ICASSP*, 2012.

[12] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep-structured conditional random fields," *IEEE Journal of Selected Topics in Signal Processing*, 2010.

[13] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML*, 2008.

[14] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv*, 2012.

[15] H. Kudo, "Crf++: Yet another crf toolkit.," in *http://crfpp.googlecode.com/svn/trunk/doc/index.html*, 2009.

[16] Y. Wang, "Strategies for statistical spoken language understanding with small amount of data an empirical study," in *INTERSPEECH*, 2010.