HCCL at SemEval-2018 Task 8: An End-to-End System for Sequence Labeling from Cybersecurity Reports

Mingming Fu [1,2], **Xuemin Zhao** [1] , **Yonghong Yan** [1,2,3]
[1]The Key Laboratory of Speech Acoustics and Content Understanding
Institute of Acoustics, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Xinjiang Laboratory of Minority Speech and Language Information Processing
Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences
{fumingming,zhaoxuemin,yanyonghong}@hccl.ioa.ac.cn

Abstract

This paper describes HCCL team systems that participated in SemEval 2018 Task 8: SecureNLP (Semantic Extraction from cybersecurity reports using NLP). To solve the problem, our team applied a neural network architecture that benefits from both word and character level representaions automatically, by using combination of Bi-directional LSTM, CNN and CRF (Ma and Hovy, 2016). Our system is truly end-to-end, requiring no feature engineering or data preprocessing, and we ranked 4th in the subtask 1, 7th in the subtask 2 and 3rd in the SubTask2-relaxed.

1 Introduction

Recently, cybersecurity defense has also been recognized as one of the problem areas likely to be important both for advancing AI and for its long-run impact on society. In particular, natural language processing (NLP) has the potential for substantial contribution in cybersecurity and that this is a critical research area given the urgency and risks involved (Lim et al., 2017).

In SemEval 2018 Task 8 (Phandi et al., 2018), there are four subtask:

1. SubTask1: Classify if a sentence is useful for inferring malware actions and capabilities
2. SubTask2: predict the token labels in the sentences. The output needs to be in BIO format. There are 3 types of token labels: ”Action”, ”Entity”, and ”Modifier”.
3. SubTask3: predict the relations between the token labels
4. SubTask4: predict the attributes for each entity token

In this evaluation, our team submitted the results of Subtask 1 and Subtask 2. To tackle this problem, we treat subtask 2 as a sequence labeling problem. Most traditional high performance sequence labeling models are linear statistical models, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Luo et al., 2015), which rely heavily on hand-crafted features and taskspecific resources.

Recently, many neural network based methods have been successfully applied to sequence labeling task: Named Entity Recognition (Lample et al., 2016). In this paper, we present an end-to-end System (combined CNN, LSTM and CRF) for sequence labeling that uses no complicated handcrafted features or domain knowledge. LSTM is capable of learning long-term dependencies, which is beneficial to sequence modeling tasks. And character level CNN can get character-level representation. For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence. So we model label sequence jointly using a conditional random field (CRF), instead of decoding each label independently. Therefore, the system we proposed is based on CNN, Bi-directional LSTM and CRF. And in the SubTask2-relaxed our group ranked third. As for SubTask1, we proposed a ruled based method that if any token in the sentence is labled ”Action”, ”Entity”, or ”Modifier”, the sentence would be considered relevant. Our team ranked 4th in the subtask 1.

2 System Description

In this section, we describe the components (layers) of our end-to-end system. We design our model with CNN-BiLSTM-CRF that combined word level representation, character level representation and POS representation as feature input, and outperform than the baseline in subtask2-relaxed.

2.1 Feature Embedding

Feature representation as the meta input of neural network have received a great deal of attention, and there are many outstanding achievements. In our system, the word level embedding is trained by the Google's Word2Vec (Mikolov et al., 2013) tool. Previous studies (Santos and Guimaraes, 2015; Chiu and Nichols, 2015)have shown that CNN is an effective approach to extract morphological information (like the prefix or suffix of a word) from characters of words and encode it into neural representations. To get more diverse information, our team decided to use Part-Of-Speech(POS) as extra feature input.

Word level Embeddings: Taking into account the particularity of the cybersecurity, we use the evaluation data to train our own word embeddings. Word level embeddings are trained by Word2Vec[1], and we set embedding dim = 300.

Character level Embeddings: Character level embeddings are random initialization(trainable), and we set embedding dim = 30.

POS Embeddings: POS embeddings are random initialization(trainable), and we set embedding dim = 30.

2.2 Model

We provide a brief description of CNN, LSTM and CRF, and present a hybrid sequence labeling architecture. This architecture is similar to the ones presented by (Ma and Hovy, 2016).

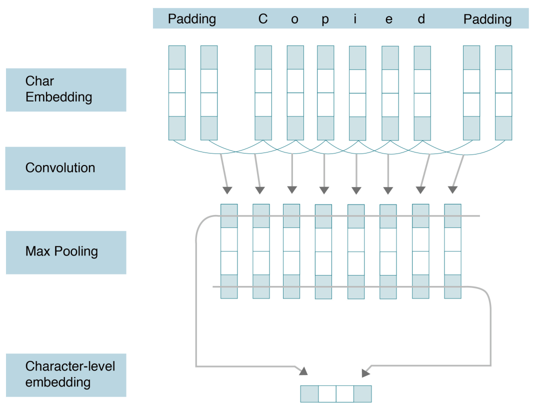

Figure 1: The CNN network for extracting character-level embedding of words.

[1]https://code.google.com/archive/p/word2vec/

2.2.1 CNN

Figure 1 shows the CNN we use to extract character-level representation of a given word. The CNN is similar to the (Chiu and Nichols, 2015), except that we use only character embeddings as the inputs to CNN, without character type features. A dropout layer (Srivastava et al., 2014) is applied before character embeddings are input to CNN.

2.2.2 LSTM

Recurrent neural networks (RNNs) are a family of neural networks that operate on sequential data. Although RNN can, in theory, learn long dependencies, in practice they fail to do so and tend to be biased towards their most recent inputs in the sequence (Bengio et al., 1994). Long Short-term Memory Network (LSTM) have been designed to combat this issue by incorporating a memory-cell and have been shown to capture long-range dependencies. They do so using several gates that control the proportion of the input to give to the memory cell, and the proportion from the previous state to forget (Hochreiter and Schmidhuber, 1997). We use the following implementation:

We will refer to the former as the forward LSTM and the latter as the backward LSTM. This forward and backward LSTM pair is referred to as a bidirectional LSTM (Graves and Schmidhuber, 2005; Dyer et al., 2015).The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively.

2.2.3 CRF

For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence. Therefore, we model label sequence jointly using a conditional random field (CRF) (Lafferty et al., 2001), instead of decoding each label independently.

For a sequence CRF model (only interactions between two successive labels are considered), training and decoding can be solved efficiently by adopting the Viterbi algorithm.

2.2.4 CNN-BiLSTM-CRF

Finally, we construct our neural network model by feeding the output vectors of BiLSTM into a

CRF layer. Figure 2 illustrates the architecture of our network in detail. For each word, the character-level is computed by the CNN in Figure 1 with character embeddings as inputs, and we use NLTK[2] to get POS information. Then the character-level representation vector the POS representation vector are concatenated with the word embedding vector to feed into the BiLSTM network. Finally, the output vectors of BiLSTM are fed to the CRF layer to jointly decode the best label sequence. As shown in Figure 2, dropout layers are applied on both the input and output vectors of BiLSTM. Experimental results show that using dropout significantly improve the performance of our model.

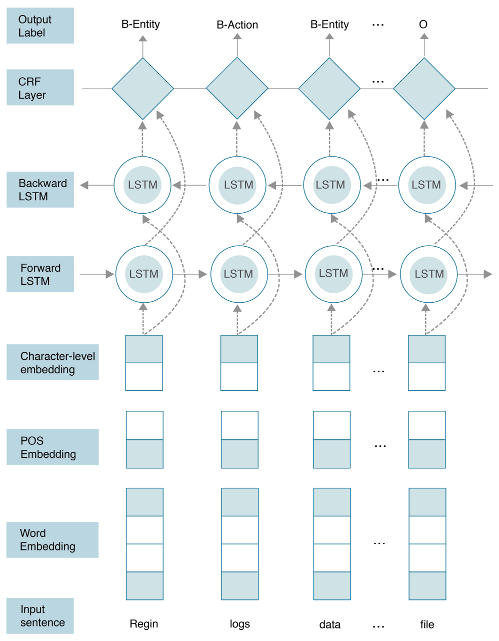

Figure 2: Main architecture of the network. Concatenated feature embeddings are given to BiLSTM.

3 Experiments and Results

3.1 Training

For model presented, we train our networks using the back-propagation algorithm updating our parameters on every training batch, using Adam with a learning rate of 0.001 and a gradient clipping of 5.0. Our CNN-BiLSTM-CRF model uses a single layer for the forward and backward LSTMs whose dimensions are set to 300. Tuning this dimension did not significantly impact model performance. We set the dropout rate to 0.5. Using higher rates negatively impacted our results, while smaller rates led to longer training time. The models were implemented in TensorFlow[3] and experiments were run on K80 GPU.

3.2 Result

In this work, our team submitted the subtask 1 and subtask 2 results. The results of all the teams are shown in Table 2.

For subtask1, its goal is to classify if a sentence is relevant for inferring malware actions and capabilities. We make use of the result in subtask2 for this subtask and consider a sentence to be relevant as long as it has an annotated token label. Table 2 shows that our system is ranked 4th and behave better than baseline for subtask1.

For subtask2, our CNN-BiLSTM-CRF model is then trained to predict token labels from cybersecurity reports. From Table 2 we can see that in subtask2, our system is slightly worse than the baseline. However, our system has a 22.5% improvement in subtask2-relaxed than baseline.

3.3 Error Analysis

For subTask1, a lot of non-malware sentences are regarded as malware sentences. May be due to the fact that we use the subTask2 output to estimate whether the current sentence is non-malware sentence or malware sentence, so the errors of subTask2 will affect subTask1. And both non-malware sentences and malware sentences contain annotated tokens.

For subTask2, we find that many unannotated tokens are labeled as annotated tokens and annotated tokens are not labeled. By analyzing the data, we found that the same words occurring as both unannotated and annotated tokens in the sentences, which might make our system achieve a low F-score.

4 Conclusion

In this paper we presented the system we used to compete in the SemEval-2018 Semantic Extraction from cybersecurity reports using NLP competition. Our goal is to implement a deep learning based end-to-end system that can solve cross domain sequence labeling issues without complicated feature engineering.

For future work, it would be interesting to explore systems that can solve the problem of self-

[2]http://www.nltk.org/

[3]https://www.tensorflow.org/

Team	SubTask1	SubTask2	SubTask2-relaxed
Team 1	0.57	0.23	0.31
Team 2	0.57	0.28	0.36
Team 3	0.52	0.29	0.39
Our Team	**0.52 (4)**	**0.22 (7)**	**0.38 (3)**
Team 5	0.52	0.16	0.25
Team 6	0.50	0.25	0.36
Team 7	0.49	0.28	0.39
Team 8	0.18	0.22	0.32
Team 9	0.15	0.21	0.28
Baseline	0.51	0.23	0.31

Table 1: Results on subtask1 and subtask2.

adaptation between different domains. And transfer learning might be a way to handle the lack of labeled data.

5 Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Nos. 11590770-4, 61650202, U1536117, 61671442, 11674352, 11504406, 61601453), the National Key Research and Development Program (Nos. 2016YFB0801203, 2016YFC0800503) and the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No. 2016A03007-1).

References

Yoshua Bengio, Patrice Y Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Jason P C Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4(0):357–370.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *meeting of the association for computational linguistics*, pages 334–343.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. 4:2047–2052.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

John D Lafferty, Andrew Mccallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. pages 260–270.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. Malwaretextdb: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1557–1567.

Gang Luo, Xiaojiang Huang, Chin Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv: Computation and Language*.

Peter Phandi, Amila Silva, and Wei Lu. 2018. Semeval-2018 Task 8: Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP). In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Cicero Nogueira Dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv: Computation and Language*, pages 25–33.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.