An Attentive Neural Architecture for Fine-grained Entity Type Classification

Sonse Shimaoka[†*] **Pontus Stenetorp**[‡] **Kentaro Inui**[†] **Sebastian Riedel**[‡]
{simaokasonse,inui}@ecei.tohoku.ac.jp
{p.stenetorp,s.riedel}@cs.ucl.ac.uk
[†]Graduate School of Information Sciences, Tohoku University
[‡]Department of Computer Science, University College London

Abstract

In this work we propose a novel attention-based neural network model for the task of fine-grained entity type classification that unlike previously proposed models recursively composes representations of entity mention contexts. Our model achieves state-of-the-art performance with 74.94% loose micro F1-score on the well-established FIGER dataset, a relative improvement of 2.59% . We also investigate the behavior of the attention mechanism of our model and observe that it can learn contextual linguistic expressions that indicate the fine-grained category memberships of an entity.

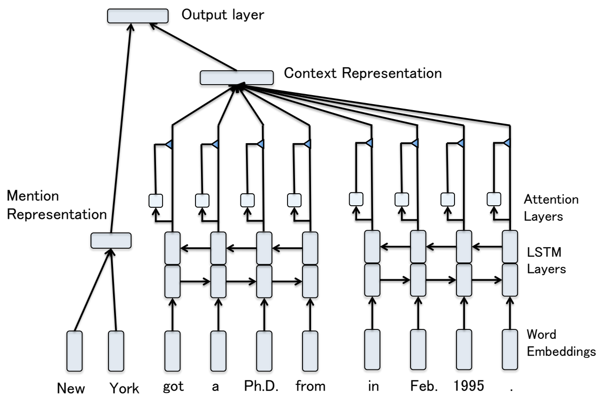


Figure 1: An illustration of our proposed model predicting fine-grained semantic types for the mention "New York" in the sentence "She got a Ph.D from New York in Feb. 1995.".

1 Introduction

Entity type classification is the task of assigning semantic types to mentions of entities in sentences. Identifying the types of entities is useful for various natural language processing tasks, such as relation extraction (Ling and Weld, 2012), question answering (Lee et al., 2006), and knowledge base population (Carlson et al., 2010). Unfortunately, most entity type classification systems use a relatively small number of types (e.g. `person`, `organization`, `location`, `time`, and `miscellaneous` (Grishman and Sundheim, 1996)) which may be too coarse-grained for some NLP applications (Sekine, 2008). To address this shortcoming, a series of recent work has investigated entity type classification with a large set of fine-grained types (Lee et al., 2006; Ling and Weld, 2012; Yosef et al., 2012; Yogatama et al., 2015; Del Corro et al., 2015).

Existing fine-grained entity type classification systems have used approaches ranging from sparse binary features to dense vector representations of entities to model the entity mention and its context. However, no previously proposed system has attempted to learn to recursively compose representations of entity context. For example, one can see that a phrase "got a Ph.D. from" is indicative of the next words being an educational institution, something which would be helpful for fine-grained entity type classification.

In this work our main contributions are two-fold:

1. A first model for fine-grained entity type classification that learns to recursively compose representations for the context of each mention and attains state-of-the-art performance on a

[*] This work was conducted during a research visit to University College London.

Proceedings of AKBC 2016, pages 69–74,
San Diego, California, June 12-17, 2016. ©2016 Association for Computational Linguistics

well-established dataset.

2. The observation that by incorporating an attention mechanism into our model, we not only achieve better performance, but also are able to observe that the model learns contextual linguistic expressions that indicate fine-grained category memberships of an entity.

2 Related Work

To the best of our knowledge, Lee et al. (2006) were the first to address the task of fine-grained entity type classification. They defined 147 fine-grained entity types and evaluated a conditional random fields-based model on a manually annotated Korean dataset. Sekine (2008) advocated the necessity of a large set of types for entity type classification and defined 200 types which served as a basis for future work on fine-grained entity type classification.

Ling and Weld (2012) defined a set of 112 types based on Freebase and created a training dataset from Wikipedia using a distant supervision method inspired by Mintz et al. (2009). For evaluation, they created a small manually annotated dataset of newspaper articles and also demonstrated that their system, FIGER, could improve the performance of a relation extraction system by providing fine-grained entity type predictions as features. Yosef et al. (2012) organised 505 types in a hierarchical taxonomy, with several hundreds of types at different levels. Based on this taxonomy they developed a multi-label hierarchical classification system. In Yogatama et al. (2015) the authors proposed to use label embeddings to allow information sharing between related labels. This approach lead to improvements on the FIGER dataset, and they also demonstrated that fine-grained labels can be used as features to improve coarse-grained entity type classification performance. Del Corro et al. (2015) introduced the most fine-grained entity type classification system to-date, it operates on the the entire WordNet hierarchy with more than $16,000$ types.

While all previous models relied on hand-crafted features, Dong et al. (2015) defined 22 types and created a two-part neural classifier. They used a recurrent neural networks to recursively obtain a vector representation of each entity mention and used a fixed-size window to capture the context of each mention. The key difference between our work and theirs lies in that we use recursive neural networks to compose context representations and that we employ an attention mechanism to allow our model to focus on relevant expressions.

3 Models

3.1 Task Formulation

We formulate the entity type classification problem as follows. Given an entity mention and its left and right context, our task is to predict its types. Formally, the input is $l_1, ..., l_C, m_1, ..., m_M, r_1, ..., r_C$, where C is the window size of the left and right context, l_i and r_i represents a word in those contexts, M is the window size of the mention, and m_i is a mention word. If a context or a mention extends beyond the sentence length, a padding symbol is used in-place of a word. Given this input we compute a probability $y_k \in \mathbb{R}$ for each of the K types.

At inference, the type k is predicted if y_k is greater than 0.5 or y_k is the maximum value $\forall k \in K$. The motivation of the former is that it acts as a cut-off, while the latter enforces the constraint that each mention is assigned at least one type.

3.2 General Model

While both mentions and contexts play important roles in determining the types, the complexity of learning to represent them are different. During initial experiments, we observed that our model could learn from mentions significantly easier than from the context, leading to poor model generalization. This motivated us to use different models for modeling mentions and contexts. Specifically, all of our models described below firstly compute a mention representation $v_m \in \mathbb{R}^{D_m \times 1}$ and context representation $v_c \in \mathbb{R}^{D_c \times 1}$ separately, and then concatenate them to be passed to the final logistic regression layer with weight matrix $W_y \in \mathbb{R}^{K \times (D_m + D_c)}$:

$$y = \frac{1}{1 + \exp\left(-W_y \begin{bmatrix} v_m \\ v_c \end{bmatrix}\right)} \tag{1}$$

Note that we did not include a bias term in the above formulation since the type distribution in the training and test corpus could potentially be significantly different due to domain differences. That is, in logistic regression, a bias fits to the empirical distribution of types in the training set, which would lead to bad performance on a test set that has a different type distribution.

The loss L for a prediction y when the true labels are encoded in a binary vector $t \in \{0,1\}^{K \times 1}$ is the following cross entropy loss function:

$$L(y,t) = \sum_{k=1}^{K} -t_k \log(y_k) - (1-t_k)\log(1-y_k) \tag{2}$$

3.3 Mention Representation

Mention representations are computed by averaging all the embeddings of the words in the mention. Let the vocabulary be V and the function $u : V \mapsto \mathbb{R}^{D_m \times 1}$ be a mapping from a word to its embedding. Formally, the mention representation v_m is obtained as follows.

$$v_m = \frac{1}{M}\sum_{i=1}^{M} u(m_i) \tag{3}$$

During our experiments we were surprised by the fact that unlike the observations made by Dong et al. (2015), complex neural models did not work well for learning mention representations compared to the simpler model described above. One possible explanation for this would be labeling discrepancies between the training and test set. For example, the label `time` is assigned to days of the week (e.g. "Friday", "Monday", and "Sunday") in the test set, but not in the training set, whereas explicit dates (e.g. "Feb. 24" and "June 4th") are assigned the `time` label in both the training and test set. This may be harmful for complex models due to their tendency to overfit on the training data.

3.4 Context Representation

We compare three methods for computing context representations.

3.4.1 Averaging Encoder

Applying the same averaging approach as for the mention representation for both the left and right context. Thus, the concatenation of those two vectors becomes the representation of the context:

$$v_c = \frac{1}{C}\sum_{i=1}^{C} \begin{bmatrix} u(l_i) \\ u(r_i) \end{bmatrix} \tag{4}$$

3.4.2 LSTM Encoder

The left and right context are encoded recursively using an LSTM cell (Hochreiter and Schmidhuber, 1997). Given an input embedding $u_i \in \mathbb{R}^{D_m \times 1}$, the previous output $h_{i-1} \in \mathbb{R}^{D_h \times 1}$, and the previous cell state $s_{i-1} \in \mathbb{R}^{D_h \times 1}$, the high-level formulation of the recursive computation by an LSTM cell is as follows:

$$h_i, s_i = lstm(u_i, h_{i-1}, s_{i-1}) \tag{5}$$

For the left context, the model reads sequences $l_1, ..., l_C$ from left to right to produce the outputs $\overrightarrow{h_1^l}, ..., \overrightarrow{h_C^l}$. For the right context, the model reads sequences $r_C, ..., r_1$ from right to left to produce the outputs $\overleftarrow{h_1^r}, ..., \overleftarrow{h_C^r}$. Then the representation v_c is obtained by concatenating $\overrightarrow{h_C^l}$ and $\overleftarrow{h_1^r}$:

$$v_c = \begin{bmatrix} \overrightarrow{h_C^l} \\ \overleftarrow{h_1^r} \end{bmatrix} \tag{6}$$

A more detailed formulation of the LSTM used in this work can be found in Sak et al. (2014).

3.4.3 Attentive Encoder

While an LSTM can encode sequential data, it still finds it difficult to learn long-term dependencies. Inspired by recent work using attention mechanisms for natural language processing (Hermann et al., 2015; Rocktäschel et al., 2015), we circumvent this problem by introducing a novel attention mechanism. We also hypothesize that by incorporating an attention mechanism the model can recognize informative expressions for the classification and make the model behavior more interpretable.

The computation of the attention mechanism is as follows. Firstly, for both the right and left context, we encode the sequences using bi-directional LSTMs (Graves, 2012). We denote the outputs as $\overrightarrow{h_1^l}, \overleftarrow{h_1^l}, ..., \overrightarrow{h_C^l}, \overleftarrow{h_C^l}$ and $\overrightarrow{h_1^r}, \overleftarrow{h_1^r}, ..., \overrightarrow{h_C^r}, \overleftarrow{h_C^r}$.

For each output layer of the bi-directional LSTMs, we compute a scalar value $\tilde{a}_i \in \mathbb{R}$ using a two-layer feed forward neural network $e_i \in \mathbb{R}^{D_a \times 1}$ and weight matrices $W_e \in \mathbb{R}^{D_a \times 2D_h}$ and $W_a \in \mathbb{R}^{1 \times D_a}$. We then normalize these scalar values such that they sum to 1. We refer to these normalized scalar values $a_i \in \mathbb{R}$ as attentions. Lastly, we take a weighted sum of the output layers of the bidirectional LSTMs as the representation of the context weighted by the attentions a_i:

$$\begin{aligned} e_i^l &= \tanh\left(W_e \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix}\right) && (7) \\ \tilde{a}_i^l &= \exp(W_a e_i^l) && (8) \\ a_i^l &= \frac{\tilde{a}_i^l}{\sum_{i=1}^{C} \tilde{a}_i^l + \tilde{a}_i^r} && (9) \\ v_c &= \sum_{i=1}^{C} a_i^l \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i^r} \\ \overleftarrow{h_i^r} \end{bmatrix} && (10) \end{aligned}$$

The equations for computing e_i^r, $\tilde{a}_i^r$, and a_i^r were omitted for brevity and the overall picture of our proposed model is illustrated in Figure 1.

4 Experiment

4.1 Dataset

To train and evaluate our model we use the publicly available FIGER dataset with 112 fine-grained types from Ling and Weld (2012). The sizes of our datasets are $2,600,000$ for training, $90,000$ for development, and 563 for testing. Note that the train and development sets were created from Wikipedia, whereas the test set is a manually annotated dataset of newspaper articles.

4.2 Pre-trained Word Embeddings

The only features used by our model are pre-trained word embeddings that were not updated during training to help the model generalize for words not appearing in the training set. Specifically, we used the freely available 300 dimensional cased word embeddings trained on 840 billion tokens from the Common Crawl supplied by Pennington et al. (2014). As embeddings for out-of-vocabulary words, we used the embedding of the "unk" token from the pre-trained embeddings.

4.3 Evaluation Criteria

Following Ling and Weld (2012), we evaluate the model performances by strict, loose macro, and loose micro measures. For the i-th instance, let the set of the predicted types be $\hat{T}_i$, and the set of the true types be T_i. Then the precisions and recall for each measure are computed as follows.

- strict

$$Precision = Recall = \frac{1}{N} \sum_{i=1}^{N} \delta(\hat{T}_i = T_i) \quad (11)$$

- loose macro

$$Precision = \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{T}_i \cap T_i|}{|\hat{T}_i|} \quad (12)$$

$$Recall = \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{T}_i \cap T_i|}{|T_i|} \quad (13)$$

- loose micro

$$Precision = \frac{\sum_{i=1}^{N} |\hat{T}_i \cap T_i|}{\sum_{i=1}^{N} |\hat{T}_i|} \quad (14)$$

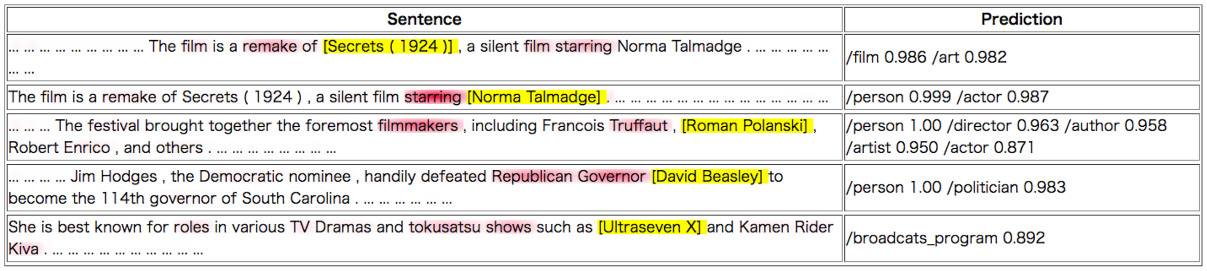
$$Recall = \frac{\sum_{i=1}^{N} |\hat{T}_i \cap T_i|}{\sum_{i=1}^{N} |T_i|} \quad (15)$$

Where N is the total number of instances.

4.4 Hyperparameter Settings

As hyperparameters, all three models used the same $D_m = 300$ dimensional word embeddings, the hidden-size of the LSTM was set to $D_h = 100$, and the hidden-layer size of the attention module was set to $D_a = 50$. We used Adam (Kingma and Ba, 2014) as our optimization method with a learning rate of 0.005 with a mini-batch size of $1,000$. As a regularizer we used dropout with probability 0.5 applied to the mention representation.

The context window size was set to $C = 15$ and mention window size was set to $M = 5$. It should be noted that our approach is not restricted to using fixed window sizes, rather this is an implementation detail arising from current limitations of the machine learning library used when handling dynamic-width recurrent neural networks. For each epoch we iterated over the training data set ten times and then evaluated the model performance on the development set. After training we picked up the best model



Sentence	Prediction
… … … … … … … … … The film is a remake of [Secrets (1924)] , a silent film starring Norma Talmadge . … … … … … … …	/film 0.986 /art 0.982
The film is a remake of Secrets (1924) , a silent film starring [Norma Talmadge] . … … … … … … … … … … … … … …	/person 0.999 /actor 0.987
… … … The festival brought together the foremost filmmakers , including Francois Truffaut , [Roman Polanski] , Robert Enrico , and others . … … … … … … … …	/person 1.00 /director 0.963 /author 0.958 /artist 0.950 /actor 0.871
… … … … Jim Hodges , the Democratic nominee , handily defeated Republican Governor [David Beasley] to become the 114th governor of South Carolina . … … … … … … …	/person 1.00 /politician 0.983
She is best known for roles in various TV Dramas and tokusatsu shows such as [Ultraseven X] and Kamen Rider Kiva . … … … … … … … … … …	/broadcats_program 0.892

Figure 2: Examples of our model attending over contexts for a given mention.

Models	P	R	F1
Ling and Weld (2012)	-	-	69.30
Yogatama et al. (2015)	**82.23**	64.55	72.35
Averaging Encoder	68.63	69.07	68.65
LSTM Encoder	72.32	70.36	71.34
Attentive Encoder	73.63	**76.29**	**74.94**

Table 1: Loose Micro Precision (P), Recall (R), and F1-score on the test set

Models	Strict	Loose Macro	Loose Micro
Ling and Weld (2012)	52.30	69.90	69.30
Yogatama et al. (2015)	-	-	72.25
Averaging Encoder	51.89	72.24	68.65
LSTM Encoder	55.60	73.95	71.34
Attentive Encoder	**58.97**	**77.96**	**74.94**

Table 2: Strict, Loose Macro and Loose Micro F1-scores

on the development set as our final model and report the performance on the test set. Our model implementation was done in Python using the TensorFlow (Abadi et al., 2015) machine learning library.

4.5 Results

The performance of the various models are summarized Tables 1 and 2. We see that the Averaging base line performs well in spite of its relative simplicity, the LSTM model shows some improvements, and the attention model performs better than any previously proposed method. In Figure 2, we visualize the attentions for several instances that were manually selected from the development set. It is clear that our proposed model is attending over expressions relevant for the entity types such as immediately adjacent to the mention such as “starring” and “Republican Governor”, as well as more distant expressions such as “filmmakers”.

5 Conclusion

In this paper, we proposed a novel state-of-the-art neural network architecture with an attention mechanism for the task of fine-grained entity type classification. We also demonstrated that the model can successfully learn to attend over expressions that are important for the classification of fine-grained types.

Acknowledgments

This work was supported by CREST-JST, JSPS KAKENHI Grant Number 15H01702, a Marie Curie Career Integration Award, and an Allen Distinguished Investigator Award. We would like to thank the anonymous reviewers and Koji Matsuda for their helpful comments and feedback.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Conference on Empirical Methods in Natural Language Processing*, pages 868–878. ACL.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1243–1249. AAAI Press.

Alex Graves. 2012. *Supervised sequence labelling*. Springer.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Changki Lee, Yi-Gyu Hwang, Hyo-Jung Oh, Soojong Lim, Jeong Heo, Chung-Hee Lee, Hyeon-Jin Kim, Ji-Hyun Wang, and Myung-Gil Jang. 2006. Fine-grained named entity recognition using conditional random fields for question answering. In *Information Retrieval Technology*, pages 581–587. Springer.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *In Proc. of the 26th AAAI Conference on Artificial Intelligence*. Citeseer.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342.

Satoshi Sekine. 2008. Extended named entity ontology with attribute information. In *LREC*, pages 52–57.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*, pages 26–31.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *24th International Conference on Computational Linguistics*, pages 1361–1370. ACL.