

# Neural Architectures for Fine-grained Entity Type Classification

Sonse Shimaoka<sup>†\*</sup>, Pontus Stenetorp<sup>‡</sup>, Kentaro Inui<sup>†</sup> and Sebastian Riedel<sup>‡</sup>

<sup>†</sup>Graduate School of Information Sciences, Tohoku University

<sup>‡</sup>Department of Computer Science, University College London

{simaokasonse, inui}@ecei.tohoku.ac.jp

{p.stenetorp, s.riedel}@cs.ucl.ac.uk

## Abstract

In this work, we investigate several neural network architectures for fine-grained entity type classification and make three key contributions. Despite being a natural comparison and addition, previous work on attentive neural architectures have not considered hand-crafted features and we combine these with learnt features and establish that they complement each other. Additionally, through quantitative analysis we establish that the attention mechanism learns to attend over syntactic heads and the phrase containing the mention, both of which are known to be strong hand-crafted features for our task. We introduce parameter sharing between labels through a hierarchical encoding method, that in low-dimensional projections show clear clusters for each type hierarchy. Lastly, despite using the same evaluation dataset, the literature frequently compare models trained using different data. We demonstrate that the choice of training data has a drastic impact on performance, which decreases by as much as 9.85% loose micro F1 score for a previously proposed method. Despite this discrepancy, our best model achieves state-of-the-art results with 75.36% loose micro F1 score on the well-established FIGER (GOLD) dataset and we report the best results for models trained using publicly available data for the OntoNotes dataset with 64.93% loose micro F1 score.

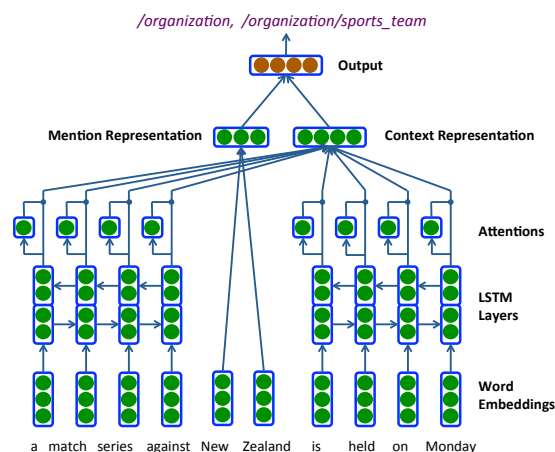


Figure 1: An illustration of the attentive encoder neural model predicting fine-grained semantic types for the mention “New Zealand” in the expression “a match series against New Zealand is held on Monday”.

## 1 Introduction

Entity type classification aims to label entity mentions in their context with their respective semantic types. Information regarding entity type mentions has proven to be valuable for several natural language processing tasks; such as question answering (Lee et al., 2006), knowledge base population (Carlson et al., 2010), and co-reference resolution (Recasens et al., 2013). A natural extension to traditional entity type classification has been to divide the set of types – which may be too *coarse-grained* for some applications (Sekine, 2008) – into a larger set of *fine-grained* entity types (Lee et al., 2006; Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014; Del Corro et al., 2015); for example *person* into *actor*, *artist*, etc.

Given the recent successes of attentive neural

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

models for information extraction (Globerson et al., 2016; Shimaoka et al., 2016; Yang et al., 2016), we investigate several variants of an attentive neural model for the task of fine-grained entity classification (e.g. Figure 1). This model category uses a neural attention mechanism – which can be likened to a soft alignment – that enables the model to focus on informative words and phrases. We build upon this line of research and our contributions are three-fold:

1. Despite being a natural comparison and addition, previous work on attentive neural architectures do not consider hand-crafted features. We combine learnt and hand-crafted features and observe that they complement each other. Additionally, we perform extensive analysis of the attention mechanism of our model and establish that the attention mechanism learns to attend over syntactic heads and the tokens prior to and after a mention, both which are known to be highly relevant to successfully classifying a mention.
2. We introduce label parameter sharing using a hierarchical encoding that improves performance on one of our datasets and the low-dimensional projections of the embedded labels form clear coherent clusters.
3. While research on fine-grained entity type classification has settled on using two evaluation datasets, a wide variety of training datasets have been used – the impact of which has not been established. We demonstrate that the choice of training data has a drastic impact on performance, observing performance decreases by as much as 9.85% loose Micro F1 score for a previously proposed method. However, even when comparing to models trained using different datasets we report state-of-the-art results of 75.36% loose micro F1 score on the FIGER (GOLD) dataset.

## 2 Related Work

Our work primarily draws upon two strains of research, fine-grained entity classification and attention mechanisms for neural models. In this section we introduce both of these research directions.

By expanding a set of coarse-grained types into a set of 147 fine-grained types, Lee et al. (2006)

were the first to address the task of fine-grained entity classification. Their end goal was to use the resulting types in a question answering system and they developed a conditional random field model that they trained and evaluated on a manually annotated Korean dataset to detect and classify entity mentions. Other early work include Sekine (2008), that emphasised the need for having access to a large set of entity types for several NLP applications. The work primarily discussed design issues for fine-grained set of entity types and served as a basis for much of the future work on fine-grained entity classification.

The first work to use distant supervision (Mintz et al., 2009) to induce a large – but noisy – training set and manually label a significantly smaller dataset to evaluate their fine-grained entity classification system, was Ling and Weld (2012) who introduced both a training and evaluation dataset FIGER (GOLD). Arguing that fine-grained sets of types must be organised in a very fine-grained hierarchical taxonomy, Yosef et al. (2012) introduced such a taxonomy covering 505 distinct types. This new set of types lead to improvements on FIGER (GOLD), and they also demonstrated that the fine-grained labels could be used as features to improve coarse-grained entity type classification performance. More recently, continuing this very fine-grained strategy, Del Corro et al. (2015) introduced the most fine-grained entity type classification system to date, covering the more than 16,000 types contained in the WordNet hierarchy.

While initial work largely assumed that mention assignments could be done independently of the mention context, Gillick et al. (2014) introduced the concept of context-dependent fine-grained entity type classification where the types of a mention is constrained to what can be deduced from its context and introduced a new OntoNotes-derived manually annotated evaluation dataset. In addition, they addressed the problem of label noise induced by distant supervision and proposed three label cleaning heuristics. Building upon the noise reduction aspects of this work, Ren et al. (2016) introduced a method to reduce label noise even further, leading to significant performance gains on both the evaluation dataset of Ling and Weld (2012) and Gillick et al. (2014).

Yogatama et al. (2015) proposed to map hand-crafted features and labels to embeddings in or-

der to facilitate information sharing between both related types and features. A pure feature learning approach was proposed by Dong et al. (2015). They defined 22 types and used a two-part neural classifier that used a recurrent neural network to obtain a vector representation of each entity mention and in its second part used a fixed-size window to capture the context of a mention. A recent workshop paper (Shimaoka et al., 2016) introduced an attentive neural model that unlike previous work obtained vector representations for each mention context by composing it using a recurrent neural network and employed an attention mechanism to allow the model to focus on relevant expressions in the mention context. Although not pointed in Shimaoka et al. (2016), the attention mechanism used differs from previous work in that it does not condition the attention. Rather, they used global weights optimised to provide attention for every fine-grained entity type classification decision.

To the best of our knowledge, the first work that utilised an attention architecture within the context of NLP was Bahdanau et al. (2014), that allowed a machine translation decoder to attend over the source sentence. Doing so, they showed that adding the attention mechanism significantly improved their machine translation results as the model was capable of learning to align the source and target sentences. Moreover, in their qualitative analysis, they concluded that the model can correctly align mutually related words and phrases. For the set of neural models proposed by Hermann et al. (2015), attention mechanisms are used to focus on the aspects of a document that help the model answer a question, as well as providing a way to qualitatively analyse the inference process. Rocktäschel et al. (2015) demonstrated that by applying an attention mechanism to a textual entailment model, they could attain state-of-the-art results, as well as analyse how the entailing sentence would align to the entailed sentence.

Our work differs from previous work on fine-grained entity classification in that we use the *same publicly available training data* when comparing models. We also believe that we are the first to consider the *direct combination of hand-crafted features and an attentive neural model*.

Feature	Description	Example
Head	Syntactic head of the mention	Obama
Non-head	Non-head words of the mention	Barack, H.
Cluster	Brown cluster for the head token	1110, ...
Characters	Character trigrams for the mention head	:ob, oba, ...
Shape	Word shape of the mention phrase	Aa A. Aa
Role	Dependency label on the mention head	subj
Context	Words before and after the mention	B:who, A:first
Parent	The head's lexical parent	picked
Topic	The LDA topic of the document	LDA:13

Table 1: Hand-crafted features, based on those of Gillick et al. (2014), used by the sparse feature and hybrid model variants in our experiments. The features are extracted for each entity mention and the example mention used to extract the example features in this table is "... who [Barack H. Obama] first picked ...".

### 3 Models

In this section we describe the neural model variants used in this paper as well as a strong feature-based baseline from the literature. We pose fine-grained entity classification as a multi-class, multi-label classification problem. Given a mention in a sentence, the classifier predicts the types  $t \in \{1, 0\}^K$  where  $K$  is the size of the set of types. Across all the models, we compute a probability  $y_k \in \mathbb{R}$  for each of the  $K$  types using logistic regression. Variations of the models stem from the ways of computing the input to the logistic regression.

At inference time, we enforce the assumption that at least one type is assigned to each mention by first assigning the type with the largest probability. We then assign any additional types based on the condition that their corresponding probabilities must be greater than a threshold of 0.5, which was determined by tuning it using development data.

#### 3.1 Sparse Feature Model

For each entity mention  $m$ , we create a binary feature indicator vector  $f(m) \in \{0, 1\}^{D_f}$  and feed it to the logistic regression layer. The features used are described in Table 1, which are comparable to those used by Gillick et al. (2014) and Yogatama et al. (2015). It is worth noting that we aimed for this model to resemble the independent classifier model in Gillick et al. (2014) so that it constitutes as a meaningful well-established baseline; however, there are two noteworthy differences. Firstly, we use the more commonly used clustering method of Brown et al. (1992), as opposed to Uszkoreit and Brants (2008), as Gillick

et al. (2014) did not make the data used for their clusters publicly available. Secondly, we learned a set of 15 topics from the OntoNotes dataset using the LDA (Blei et al., 2003) implementation from the popular gensim software package,<sup>1</sup> in contrast to Gillick et al. (2014) that used a supervised topic model trained using an unspecified dataset. Despite these differences, we argue that our set of features is comparable and enables a fair comparison given that the original implementation and some of the data used is not publicly available.

### 3.2 Neural Models

The neural models from Shimaoka et al. (2016) processes embeddings of the words of the mention and its context; and we adopt the same formalism when introducing these models and our variants. First, the mention representation  $v_m \in \mathbb{R}^{D_m \times 1}$  and context representation  $v_c \in \mathbb{R}^{D_c \times 1}$  are computed separately. Then, the concatenation of these representations is used to compute the prediction:

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

Where  $W_y \in \mathbb{R}^{K \times (D_m + D_c)}$  is the weight matrix.

Let the words in the mention be  $m_1, m_2, \dots, m_{|m|}$ . Then the representation of the mention is computed as follows:

$$v_m = \frac{1}{|m|} \sum_{i=1}^{|m|} u(m_i) \quad (2)$$

Where  $u$  is a mapping from a word to an embedding. This relatively simple method for composing the mention representation is motivated by it being less prone to overfitting.

Next, we describe the three methods from Shimaoka et al. (2016) for computing the context representations; namely, Averaging, LSTM, and Attentive Encoder.

#### 3.2.1 Averaging Encoder

Similarly to the method of computing the mention representation, the Averaging encoder computes the averages of the words in the left and right context. Formally, let  $l_1, \dots, l_C$  and  $r_1, \dots, r_C$  be the words in the left and right contexts respectively, where  $C$  is the window size. Then, for each sequence of words, we compute the average of the

corresponding word embeddings. Those two vectors are then concatenated to form the representation of the context  $v_c$ .

#### 3.2.2 LSTM Encoder

For the LSTM Encoder, the left and right contexts are encoded by an LSTM (Hochreiter and Schmidhuber, 1997). The high-level formulation of an LSTM can be written as:

$$h_i, s_i = lstm(u_i, h_{i-1}, s_{i-1}) \quad (3)$$

Where  $u_i \in \mathbb{R}^{D_m \times 1}$  is an input embedding,  $h_{i-1} \in \mathbb{R}^{D_h \times 1}$  is the previous output, and  $s_{i-1} \in \mathbb{R}^{D_h \times 1}$  is the previous cell state.

For the left context, the LSTM is applied to the sequence  $l_1, \dots, l_C$  from left to right and produces the outputs  $\vec{h}_1^l, \dots, \vec{h}_C^l$ . For the right context, the sequence  $r_C, \dots, r_1$  is processed from right to left to produce the outputs  $\vec{h}_1^r, \dots, \vec{h}_C^r$ . The concatenation of  $\vec{h}_C^l$  and  $\vec{h}_1^r$  then serves as the context representation  $v_c$ .

#### 3.2.3 Attentive Encoder

An attention mechanism aims to encourage the model to focus on salient local information that is relevant for the classification decision. The attention mechanism variant used in this work is defined as follows. First, bi-directional LSTMs (Graves, 2012) are applied for both the right and left context. We denote the output layers of the bi-directional LSTMs as  $\vec{h}_1^l, \vec{h}_1^r, \dots, \vec{h}_C^l, \vec{h}_C^r$  and  $\overleftarrow{h}_1^l, \overleftarrow{h}_1^r, \dots, \overleftarrow{h}_C^l, \overleftarrow{h}_C^r$ .

For each output layer, a scalar value  $\tilde{a}_i \in \mathbb{R}$  is computed using a feed forward neural network with the hidden layer  $e_i \in \mathbb{R}^{D_a \times 1}$  and weight matrices  $W_e \in \mathbb{R}^{D_a \times 2D_h}$  and  $W_a \in \mathbb{R}^{1 \times D_a}$ :

$$e_i^l = \tanh\left(W_e \begin{bmatrix} \vec{h}_i^l \\ \overleftarrow{h}_i^l \end{bmatrix}\right) \quad (4)$$

$$\tilde{a}_i^l = \exp(W_a e_i^l) \quad (5)$$

Next, the scalar values are normalised such that they sum to 1:

$$a_i^l = \frac{\tilde{a}_i^l}{\sum_{i=1}^C \tilde{a}_i^l + \tilde{a}_i^r} \quad (6)$$

These normalised scalar values  $a_i \in \mathbb{R}$  are referred to as attentions. Finally, we compute the sum of the output layers of the bidirectional

<sup>1</sup><http://radimrehurek.com/gensim/>

LSTMs, weighted by the attentions  $a_i$  as the representation of the context:

$$v_c = \sum_{i=1}^C a_i^l \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i^r} \\ \overleftarrow{h_i^r} \\ \overleftarrow{h_i^r} \end{bmatrix} \quad (7)$$

An illustration of the attentive encoder model variant can be found in Figure 1.

### 3.3 Hybrid Models

To allow model variants to use both human background knowledge through hand-crafted features as well as features learnt from data, we extended the neural models to create new hybrid model variants as follows. Let  $v_f \in \mathbb{R}^{D_l \times 1}$  be a low-dimensional projection of the sparse feature  $f(m)$ :

$$v_f = W_f f(m) \quad (8)$$

Where  $W_f \in \mathbb{R}^{D_l \times D_f}$  is a projection matrix. The hybrid model variants are then defined as follows:

$$y = \frac{1}{1 + \exp \left( -W_y \begin{bmatrix} v_m \\ v_c \\ v_f \end{bmatrix} \right)} \quad (9)$$

These models can thus draw upon learnt features through  $v_m$  and  $v_c$  as well as hand-crafted features using  $v_f$  when making classification decisions. While existing work on fine-grained entity type classification have used either sparse, manually designed features or dense, automatically learnt embedding vectors, our work is the first to propose and evaluate a model using the combination of both features.

### 3.4 Hierarchical Label Encoding

Since the fine-grained types tend to form a forest of type hierarchies (e.g. musician is a subtype of artist, which in turn is a subtype of person), we investigated whether the encoding of each label could utilise this structure to enable parameter sharing. Concretely, we compose the weight matrix  $W_y$  for the logistic regression layer as the product of a learnt weight matrix  $V_y$  and a constant sparse binary matrix  $S$ :

$$W_y^T = V_y S \quad (10)$$

We encode the type hierarchy formed by the set of types in the binary matrix  $S$  as follows. Each type is mapped to a unique column in  $S$ , where membership at each level of

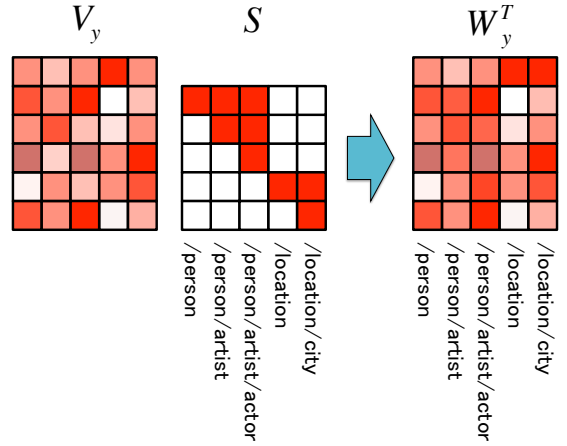


Figure 2: Hierarchical label encoding illustration.

Work	W2M	W2M+D	W2.6M	GN1	GN2
Ling and Weld (2012)	✓				
Gillick et al. (2014)				×	
Yogatama et al. (2015)					×
Ren et al. (2016)	✓	×			
Shimaoka et al. (2016)			✓		

Table 2: Training datasets used and its availability. W2M and W2.6M are Wikipedia-based, +D indicates denoising, and GN1/GN2 are two company-internal Google News datasets. The symbols ✓ and × indicates publicly available and unavailable data.

its type hierarchy is marked by a 1. For example, if we use the set of types defined by Gillick et al. (2014), the column for /person could be encoded as  $[1, 0, \dots]$ , /person/artist as  $[1, 1, 0, \dots]$ , and /person/artist/actor as  $[1, 1, 1, 0, \dots]$ . This encoding scheme is illustrated in Figure 2.

This enables us to share parameters between labels in the same hierarchy, potentially making learning easier for infrequent types that can now draw upon annotations of other types in the same hierarchy.

## 4 Experiments

### 4.1 Datasets

Despite the research community having largely settled on using the manually annotated datasets FIGER (GOLD) (Ling and Weld, 2012) and OntoNotes (Gillick et al., 2014) for evaluation, there is still a remarkable difference in the data used to train models (Table 2) that are then evaluated on the same manually annotated datasets. Also worth noting is that some data is not even publicly available, making a

fair comparison between methods even more difficult. For evaluation, in our experiments we use the two well-established manually annotated datasets FIGER (GOLD) and OntoNotes, where like Gillick et al. (2014), we discarded pronominal mentions, resulting in a total of 8,963 mentions. For training, we use the automatically induced publicly available datasets provided by Ren et al. (2016). Ren et al. (2016) aimed to eliminate label noise generated in the process of distant supervision and we use the “raw” noisy data<sup>2</sup> provided by them for training our models.

## 4.2 Pre-trained Word Embeddings

We use pre-trained word embeddings that were not updated during training to help the model generalise to words not appearing in the training set (Rocktäschel et al., 2015). For this purpose, 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). For words not present in the pre-trained word embeddings, we use the embedding of the “unk” token.

## 4.3 Evaluation Criteria

We adopt the same criteria as Ling and Weld (2012), that is, we evaluate the model performance by strict accuracy, loose macro, and loose micro scores.

## 4.4 Hyperparameter Settings

Values for the hyperparameters were obtained from preliminary experiments by evaluating the model performance on the development sets. Concretely, all neural and hybrid models used the same  $D_m = 300$ -dimensional word embeddings, the hidden-size of the LSTM was set to  $D_h = 100$ , the hidden-layer size of the attention module was set to  $D_a = 100$ , and the size of the low-dimensional projection of the sparse features was set to  $D_l = 50$ . We used Adam (Kingma and Ba, 2014) as our optimisation method with a learning rate of 0.001, a mini-batch size of 1,000, and iterated over the training data for five epochs. As a regularizer we

<sup>2</sup> Although Ren et al. (2016) provided both “raw” data and code to “denoise” the data, we were unable to replicate the performance benefits reported in their work after running their pipeline. We have contacted them regarding this as we would be interested in comparing the benefit of their denoising algorithm for each model, but at the time of writing we have not yet received a response.

Model	Acc.	Macro	Micro
Hand-crafted	51.33	71.91	68.78
Averaging	46.36	71.03	65.31
Averaging + Hand-crafted	52.58	72.33	70.04
LSTM	55.60	75.15	71.73
LSTM + Hand-crafted	57.02	76.98	73.94
Attentive	54.53	74.76	71.58
Attentive + Hand-crafted	<b>59.68</b>	<b>78.97</b>	<b>75.36</b>
FIGER (Ling and Weld, 2012)	52.30	69.90	69.30
FIGER (Ren et al., 2016)	47.4	69.2	65.5

Table 3: Performance on FIGER (GOLD) for models using the *same* W2M training data.

Model	Data	Acc.	Macro	Micro
Attentive + Hand-crafted	W2M	59.68	<b>78.97</b>	<b>75.36</b>
Attentive (Shimaoka et al., 2016)	W2.6M	58.97	77.96	74.94
FIGER + PLE (Ren et al., 2016)	W2M+D	<b>59.9</b>	76.3	74.9
HYENA + PLE (Ren et al., 2016)	W2M+D	54.2	69.5	68.1
K-WASABIE (Yogatama et al., 2015)	GN2	n/a	n/a	72.25

Table 4: Performance on FIGER (GOLD) for models using *different* training data.

used dropout (Hinton et al., 2012) with probability 0.5 applied to the mention representation and sparse feature representation. The context window size was set to  $C = 10$  and if the length of a context extends beyond the sentence length, we used a padding symbol in-place of a word. After training, we picked the best model on the development set as our final model and report their performance on the test sets. Our model implementation was done in Python using the TensorFlow (Abadi et al., 2015) machine learning library.

## 4.5 Results

When presenting our results, it should be noted that we aim to make a clear separation between results from models trained using different datasets.

### 4.5.1 FIGER (GOLD)

We first analyse the results on FIGER (GOLD) (Tables 3 and 4). The performance of the baseline model that uses the sparse hand-crafted features is relatively close to that of the FIGER system of Ling and Weld (2012). This is consistent with the fact that both systems use linear classifiers, similar sets of features, and training data of the same size and domain.

Looking at the results of neural models, we observe a consistent pattern that adding hand-crafted features boost performance significantly, indicating that the learnt and hand-crafted features complement each other. The effects of adding the hier-

Model	Acc.	Macro	Micro
Hand-crafted	48.16	66.33	60.16
Averaging	46.17	65.26	58.25
Averaging + Hier	47.15	65.53	58.25
Averaging + Hand-crafted	51.57	70.61	64.24
Averaging + Hand-crafted + Hier	<b>51.74</b>	<b>70.98</b>	64.91
LSTM	49.20	66.72	60.52
LSTM + Hier	48.96	66.51	60.70
LSTM + Hand-crafted	48.58	68.54	62.89
LSTM + Hand-crafted + Hier	50.42	69.99	64.57
Attentive	50.32	67.95	61.65
Attentive + Hier	51.10	68.19	61.57
Attentive + Hand-crafted	49.54	69.04	63.55
Attentive + Hand-crafted + Hier	50.89	70.80	<b>64.93</b>
FIGER (Ren et al., 2016)	36.90	57.80	51.60

Table 5: Performance on OntoNotes for models using the *same* W2M training data.

archical label encoding were inconsistent, sometimes increasing, sometimes decreasing performance. We thus opted not to include them in the results table due to space constraints and hypothesise that given the size of the training data, parameter sharing may not yield any large performance benefits. Among the neural models, we see that the averaging encoder perform considerably worse than the others. Both the LSTM and attentive encoder show strong results and the attentive encoder with hand-crafted features achieves the best performance among all the models we investigated.

When comparing our best model to previously introduced models trained using different training data, we find that we achieve state-of-the-art results both in terms of loose macro and micro scores. The closest competitor, FIGER + PLE (Ren et al., 2016), achieves higher accuracy at the expense of lower F1 scores, we suspect that this is due to an accuracy focus in their label pruning strategy. It is worth noting that we achieve state-of-the-art results without the need for any noise reduction strategies. Also, even with 600,000 fewer training examples, our variant with hand-crafted features of the attentive model from Shimaoka et al. (2016) outperforms its feature-learning variant.

In regards to the impact of the choice of training set, we observe that the model introduced in Shimaoka et al. (2016) drops as much as 3.36 points of loose micro score when using a smaller dataset. Thus casting doubts upon the comparability of results of fine-grained entity classification models using different training data.

#### 4.5.2 OntoNotes

Secondly, we discuss the results on OntoNotes (Tables 5, and 6). Again, we see consistent per-

Model	Data	Acc.	Macro	Micro
Averaging + Hand-crafted + Hier	W2M	51.74	70.98	64.91
Attentive + Hand-crafted + Hier	W2M	50.89	70.80	64.93
FIGER + PLE (Ren et al., 2016)	W2M+D	<b>57.2</b>	<b>71.5</b>	66.1
HYENA + PLE (Ren et al., 2016)	W2M+D	54.6	69.2	62.5
Hand-crafted (Gillick et al., 2014)	GN1	n/a	n/a	70.01
K-WASABIE (Yogatama et al., 2015)	GN2	n/a	n/a	<b>72.98</b>

Table 6: Performance on OntoNotes for models using *different* training data.

formance improvements when the sparse hand-crafted features are added to the neural models. In the absence of hand-crafted features, the averaging encoder suffer relatively poor performance and the attentive encoder achieves the best performance. However, when the hand-crafted features are added, a significant improvement occurs for the averaging encoder, making the performance of the three neural models much alike. We speculate that some of the hand-crafted features such as the dependency role and parent word of the head noun, provide crucial information for the task that cannot be captured by the plain averaging model, but can be learnt if an attention mechanism is present. Another speculative reason is that because the training dataset is noisy compared to FIGER (GOLD) (since FIGER (GOLD) uses anchors to detect entities whereas OntoNotes uses an external tool), and the size of the dataset is small, the robustness of the simpler averaging model becomes clearer when combined with the hand-crafted features.

Another interesting observation can be seen for models with the hierarchical label encoding, where it is clear that consistent performance increases occur. This can be explained by the fact that the type ontology used in OntoNotes is more well-formed than its FIGER counterpart. While we do not obtain state-of-the-art performance when considering models using different training data, we do note that in terms of F1-score we perform within 1 point of the state of the art. This being achieved despite having trained our models on different non-proprietary noisy data.

Once again we have an opportunity to study the impact of the choice of training data by comparing the results of the hand-crafted features of Gillick et al. (2014) to our own comparable set of features. What we find is that the performance drop is very dramatic, 9.85 points of loose micro score. Given that the training data for the previously introduced model is not publicly available, we hesi-

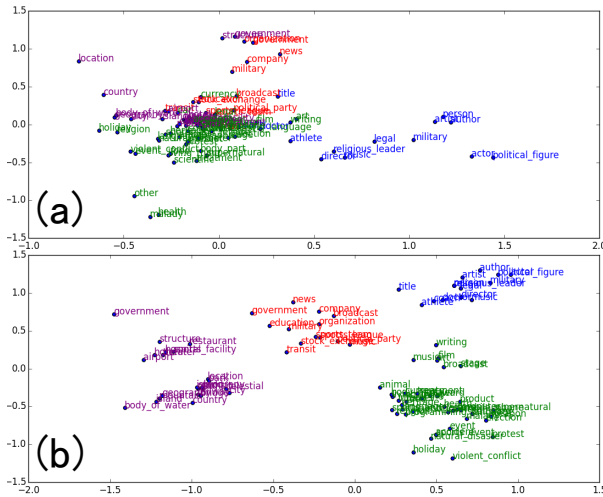


Figure 3: PCA projections of the label embeddings learnt from the OntoNotes dataset where subtypes share the same color as their parent type. Sub-figure (a) uses the non-hierarchical encoding, while sub-figure (b) uses the hierarchical encoding.

tate to speculate as to exactly why this drop is so dramatic, but similar observations have been made for entity linking (Ling et al., 2015). This clearly underlines how essential it is to compare models on an equal footing using the same training data.

#### 4.6 PCA visualisation of label embeddings

By visualising the learnt label embeddings (Figure 3) and comparing the non-hierarchical and hierarchical label encodings, we can observe that the hierarchical encoding forms clear distinct clusters.

#### 4.7 Attention Analysis

While visualising the attention weights for specific examples have become commonplace, it is still not clear exactly what syntactic and semantic patterns that are learnt by the attention mechanism. To better understand this, we first qualitatively analysed a large set of attention visualisations and observed that head words and the words contained in the phrase forming the mention tended to receive the highest level of attention. In order to quantify this notion, we calculated how frequently the word strongest attended over for all mentions of a specific type was the syntactic head or the words before and after the mention in its phrase. What we found through our analysis (Table 7) was that our attentive model without hand-crafted features does indeed learn that head words and the phrase surrounding the mention are highly indica-

Type	Parent	Before	After	Frequent Words
/location	0.319	0.228	0.070	in, at, born
/organization	0.324	0.178	0.119	at, the, by
/art/film	0.207	0.429	0.021	film, films, in
/music	0.259	0.116	0.018	album, song, single
/award	0.583	0.292	0.083	won, a, received
/event	0.310	0.188	0.089	in, during, at

Table 7: Quantitative attention analysis.

tive of the mention type, without any explicit supervision. Furthermore, we believe that this in part might explain why the performance benefit of adding hand-crafted features was smaller for the attentive model compared to our other two neural variants.

## 5 Conclusions and Future Work

In this paper, we investigated several model variants for the task of fine-grained entity type classification. The experiments clearly demonstrated that the choice of training data – which until now been ignored for our task – has a significant impact on performance. Our best model achieved state-of-the-art results with 75.36% loose micro F1 score on FIGER (GOLD) despite being compared to models trained using larger datasets and we were able to report the best results for any model trained using publicly available data for OntoNotes with 64.93% loose micro F1 score. The analysis of the behaviour of the attention mechanism demonstrated that it can successfully learn to attend over expressions that are important for the classification of fine-grained types. It is our hope that our observations can inspire further research into the limitations of what linguistic phenomena attentive models can learn and how they can be improved.

As future work, we see the re-implementation of more methods from the literature as a desirable target, so that they can be evaluated after utilising the same training data. Additionally, we would like to explore alternative hierarchical label encodings that may lead to more consistent performance benefits.

To ease the reproducibility of our work, we make our code used for the experiments available at <https://github.com/shimaokasonse/NFGEC>.

## 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 Dan Gillick for answering several questions related to his 2014 paper.

## 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.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.
- P.F. Brown, P.V. Desouza, R.L. Mercer, V.J. Della Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.
- 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.
- Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.
- Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Berlin, Germany, August. Association for Computational Linguistics.
- Alex Graves. 2012. *Supervised sequence labelling*. Springer.
- 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.
- Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- 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.
- Xiao Ling, Sameer Singh, and Daniel Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.
- 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.
- Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 627–633, Atlanta, Georgia, June. Association for Computational Linguistics.

- Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. *arXiv preprint arXiv:1602.05307*.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.
- Satoshi Sekine. 2008. Extended named entity ontology with attribute information. In *LREC*, pages 52–57.
- Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. *arXiv preprint arXiv:1604.05525*.
- Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, pages 755–762, Columbus, Ohio, June. Association for Computational Linguistics.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June. Association for Computational Linguistics.
- 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.