B5IM2023

Master's Thesis

Neural Networks for Fine-grained Entity Type Classification

Sonse Shimaoka

February 10, 2017

Graduate School of Information Science
Tohoku University

A Master's Thesis
submitted to System Information Sciences,
Graduate School of Information Science,
Tohoku University
in partial fulfillment of the requirements for the degree of
MASTER of ENGINEERING

Sonse Shimaoka

Thesis Committee:

Professor Kentaro Inui	(Supervisor)
Professor Akinori Ito	
Professor Kengo Kinoshita	
Associate Professor Naoaki Okazaki	(Co-supervisor)
Research Associate Pontus Stenetorp	(Co-supervisor, University College London)

Neural Networks for Fine-grained Entity Type Classification*

Sonse Shimaoka

Abstract

In this thesis, we investigate several neural network architectures for fine-grained entity type classification and make four key contributions. First, we incorporate pre-trained word embeddings to enable our models to use the information of similarities between word meanings and establish that hand-crafted features and word embeddings complement each other. Second, we propose a novel attention-based neural network model for the task that unlike previously proposed models recursively composes representations of entity mention contexts. Through both qualitative and 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. Third, 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. Fourth, We introduce parameter sharing between labels through a hierarchical encoding method, that in low-dimensional projections show clear clusters for each type hierarchy.

Keywords:

Fine Grained Entity Type Classification, Neural Networks

*Master's Thesis, System Information Sciences, Graduate School of Information Sciences, Tohoku University, B5IM2023, February 10, 2017.

Contents

List of Figures

List of Tables

1 Introduction

In this section we explain the objective and background of the research and provide the structure of the thesis.

1.1 Natural Language Processing and Machine Learning

We begin with explaining the general academic field in which the subject of this thesis belongs, namely natural language processing. By doing so we intend to make readers understand how our work is positioned in a broader context. Moreover, we briefly describe machine learning, which currently is the dominant algorithmic approach to solve natural language processing tasks.

1.1.1 Natural Language Processing

Languages that we use in everyday life such as English and Japanese are our most important means of communication. These languages are called natural languages, as opposed to artificial languages such as programing languages. Natural Language Processing (NLP) is a sub field in computer science that deals with information processing of natural language data.

There are many tasks in NLP that are useful at a societal level. Below, we illustrate some of those tasks that today are widely used for a variety of services. **Machine translation** automatically translates text from one human language (like Japanese) to another (like English). This makes individuals capable of reading and writing text in many different languages, thus drastically increasing the potential of worldwide communication. **Text summarization** produces a readable summary of a document such as newspaper articles. This enables us to shorten the required time to comprehend important news, which is necessary in today's world where the society keeps changing at an unprecedented speed. By performing **sentiment analysis**, computers can automatically analyze the emotions put in given words. As an example, sentiment analysis can be used to gather opinions of approval and disapproval on specific issues on products. **Document classification** determines the categories of a given document. Thanks to this technology e-mail service providers can automatically filter emails that are classified as a spam. Alternatively, document classification can be used to automatically

organize news paper articles according to their topics (sports, politics, science, economy and so on). **Information extraction** aims to extract structured information from unstructured text. An example is the extraction from news articles about corporate mergers. Organizing such information in formal knowledge base makes it possible to process the information in rigid ways using database queries. **Information retrieval** concerns finding documents that contain particular information that the user is looking for. Search engines for finding web pages in the internet is a well-known application. **Question answering** is the task of producing human-readable answers to questions described in the form of text. IBM Watson[Fer+10] is the championing example of question answering system.

While above examples of NLP tasks are in themselves acknowledged to be useful for society, in order to perform them effectively, it is necessary to do more basic processing beforehand. We explain some of those downstream tasks below. Sentences, phrases, and words are considered to be important elements of natural language. The tasks of identifying each of those elements in text are generally called **segmentation**. Word segmentation is relatively easy for languages such as English where each word is separated with the previous and next words by spaces, but difficult for languages such as Japanese and Chinese, since there are no simple indicator to determine boundaries between words. Once sentences, phrases, words, and other useful elements are identified, a typical subsequent processing step is classifying those elements into several categories. For example, the task of classifying words into parts of speech (POS) such as noun, verb, adjective, determinant, is called **POS tagging**. Segmentation and classification can be executed either separately or jointly, depending on the situation. Segmenting phrases and classifying them into syntactic categories such as noun phrase, verb phrase, and adjective phrase are together called **chunking** or **shallow paring**. Entities are atomic elements in text that belong to particular semantic types including `person`, `organization`, `time`, and `location`. Identifying and classifying entities are respectively called **entity segmentation** and **entity type classification**. When those two tasks are put together, we call it **entity recognition**. In this thesis we focus on the advanced variant of entity classification, namely **fine-grained entity classification**. After identifying and classifying various elements in text, we can proceed to detect relations between them. For example, **dependency parsing** detects the grammatical subordination relation (dependency) between two words. Another example is

coreference resolution that finds a group of expressions that refer to the same entity in a text.

1.1.2 Machine Learning Approach

We have explained upstream and downstream tasks of NLP. Next, we discuss machine learning approach to solve these tasks. Machine learning is a group of algorithmic methods that automatically detects patterns in data, and then uses the uncovered patterns to predict future data or other outcomes of interest [Mur12].

The field of machine learning is usually divided into several sub fields such as supervised learning, unsupervised learning, and reinforcement learning. This thesis mainly make use of supervised learning, which is the most widely used form of machine learning in practice. The goal of supervised learning is to learn a mapping from input variable x to output variable t, using training set $D = \{(x_i, t_i)|i = 1, ..., N\}$ that contains pairs of input and output variables where N is the number of pairs. A simple example is to learn an estimator of the gender given the weight and height of a person. In this case, input variable x is a 2 dimensional real vector and output variable t is a binary variable where $t = 1$ represents the person being female and $t = 0$ represents the person being male. In general, the domains of x and t can be arbitrarily complex. For example, x can be an image, a sentence, an email message, a time series, a molecular shape, a graph and so on. Similarly, the form of t can be in principal anything, but most methods assume t is either a categorical variable from some finite set (such as female or male) or real-valued scalar variable. If t is categorical, the problem is called classification, which covers most NLP problems. Common examples of classification problem includes document classification, cancer detection, image classification, and sentiment analysis. If t is real-valued, the problem is called regression. Well-known instances of regression problems are stock price prediction and real estate value estimation.

Supervised learning is typically done in three steps. The first step is to define a model that has parameters to be estimated. A model is formalized as a parametric function $y = f(x; \theta)$ with parameter θ. The second step is to define a loss function $L(D; \theta)$ that compares the outputs $y_i = f(x_i; \theta)$ of the model and the actual output variables t_i in the training set $D = \{(x_i, t_i)|i = 1, ..., N\}$. The third step is to conduct optimization that minimizes the loss function with respect to the parameter.

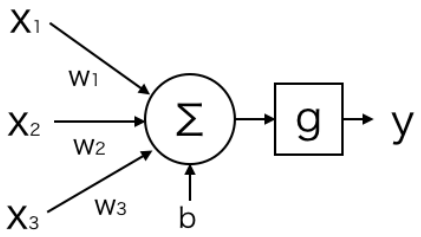

Figure 1: Computation of a single neuron.

Recently, a particular family of parametric function called neural network has attracted a great deal of attention. An artificial neural network – commonly referred to as a neural network – is a network of simple computational units called neurons. The typical computation taking place in a neuron is as follows. First, it takes real-valued input vector x and computes an intermediate scalar value $a = w^T x + b$ where the parameter w, called weight is a vector of the same dimension as x and the parameter b – called bias – is a scalar. a is called preactivation value. The output of the neuron y is computed by applying a nonlinear function g, called activation function, to the preactivation a: $y = g(a)$. Activation function can be any differentiable nonlinear function. The overall computation of a neuron that takes three input variables x_1, x_2, x_3 and produce the output y is shown in Fig 1.

What a single neuron can compute is limited to some simple domain, but when several neurons are connected together, the overall network can perform more complex computation. Fig 2 illustrates an example of such neural networks. As seen in the figure, neural networks are mostly designed to have layerwise structure. Starting from the input layer, each layer takes the output vector of the previous layer, executes the computation, and then passes the resulting output vector to the next layer. The final layer produces the overall output of the network. In this particular example, the output layer consist of a single neuron, and the sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$ is used as the activation function. This setting of the output layer is used to perform a binary classification problem. That is, the output y of the final layer takes value between 0 and 1, so this can be interpreted as the probability of the input classified as $t = 1$. When we make the actual classification decision, we can use this probability with a cut off value of 0.5 such that if $y > 0.5$, the input is classified as $t = 1$ and otherwise as $t = 0$.

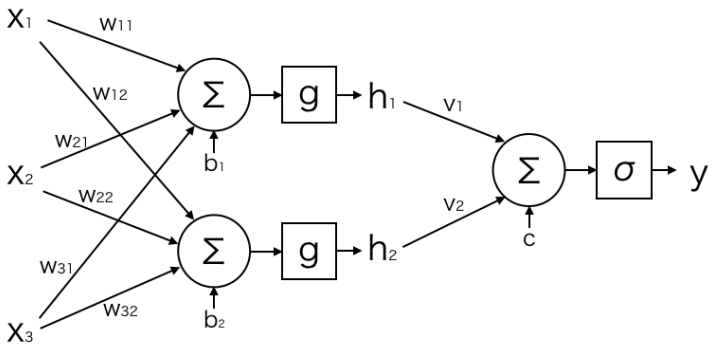

Figure 2: Computation of a neural network.

A loss function is needed to estimate how good the model output is compared to the actual output values. The choice of an appropriate loss function depends on the type of the model. In the case of the neural network that does a binary classification with the sigmoid activation function, the best choice is the following cross-entropy:

$$L(D;\theta) = \frac{1}{N}\sum_{i=1}^{N} -t_i \log(y_i) - (1-t_i)\log(1-y_i)$$

Note that above loss function decreases when the values of t_i and y_i are close and increases when they are separated. Thus this function measures the performance of the model appropriately. From a probabilistic perspective, this function can be seen as the average negative log likelihood $NLL(D;\theta) = -\frac{1}{N}\log\prod_{i=1}^{N} P(t_i = 1|x_i;\theta)$.

In order to minimize the loss function in neural networks, variants of the gradient decent algorithm are usually used. The key insight of the gradient decent is that the value of the loss function decreases if we adjust the parameters minutely along the (negative) direction represented by the gradient of the loss function with respect to the current parameters. In gradient decent, at first the parameters are initialized with random variables, and then the parameters are iteratively moved toward the direction of the gradients until the loss function converges to the locally optimal value. This iteration can be written as follows:

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial L(D;\theta)}{\partial \theta}\bigg|_{\theta=\theta_{old}}$$

Where α is a small positive scalar called the learning rate.

The gradients of complex neural networks may seem to be difficult to compute, but there is an efficient algorithm called backpropagation that does this job mechanically. Ultimately, backpropagation is just another name for the chain rule used in the basic calculus. Backpropagation is supported by most of software frameworks for neural networks such as Tensorflow [Mar+15].

1.2 Entity type classification

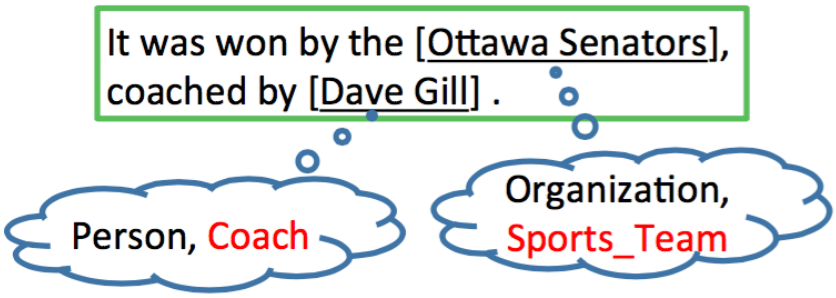

Figure 3: Fine-grained Entity Type Classification. Traditional coarse-grained labels are colored in black. Fine-grained labels are colored in red.

Next, we explain fine-grained entity classification, which is the task we focus in this thesis.

Entities are atomic elements in text that belong to particular semantic types such as `person`, `organization`, `time`, and `location`. Entity type classification aims to label entity mentions in their context with their respective semantic types. For example, in the sentence "It was won by the Ottawa Senators, coached by Dave Gill.", our goal is to label "Ottawa Senators" as `organization` and "Dave Gill" as `person`.

1.3 Applications

Information regarding entity type mentions have proven to be valuable for several natural language processing tasks.

Information Extraction

Information extraction (IE) is the task of extracting structured information from unstructured text [GS96]. One way of using entity type information in IE is constraint checking of predicate arguments [Car+10]. Consider a hypothetical situation where we want to extract the relation that a person plays for a sport team. It is reasonable to assume that the first argument of this relation should be categorized as person and the second argument should be categorized as organization. Given this assumption, we can use an entity type classification system to decide whether or not the type of an entities satisfy the constraints of the relation argument, which in turn would lead to a performance increase in precision.

Alternatively, entity type information can be directly added to the features of an existing machine learning based information extraction system. It is shown that those additional features significantly increases the performance of a relation extractor [LW12].

Question Answering

Question answering is another important application of entity type classification. Especially in factoid question answering system in which the form of the answer is restricted to be a noun phrase, knowing the semantic types of entity mentions is crucial for correctly answering questions [Lee+06; Fer+10]. For example if the question begins with the pronoun "Who", the answer is very likely to belong to person, which means that entities mentions classified as person are seen as plausible candidates of the answer.

Coreference Resolution

Coreference resolution is the task of linking entity mentions in a document that refer to the same entity. Entity type information is also useful for this task since the system can utilize the fact that if mentions are coreferent, they are very likely to share the same entity type [RMP13].

Entity Linking

Entity linking aims to link entity mentions in text with their corresponding entities in a knowledge base. Entity types of mentions are used in this task to indicate whether the

type of the entity mention in text is consistent with the type of the candidate entity in a knowledge base [SWH15].

Information Retrieval

Finally entity type information enables an advanced form of information retrieval that supports users in querying documents not only by superficial string-based matching, but also by more abstract category-based matching [JW14].

1.4 Fine-grained entity type classification

Unfortunately, most entity type classification systems have focused on a limited number of semantic types. When one of the earliest entity recognition tasks was introduced in the MUC-7 conference, only types of `person`, `location` and `organization` were considered [CR97]. Then, since type of `miscellaneous` was added in CoNLL03 [TD03], these 4 types have become the mainstream choice of the entity recognition systems including widely used Stanford Named Entity Recognizer [FGM05] .

In contrast to using a limited number of coarse-grained types, a series of recent work has investigated entity type classification with a large set of fine-grained types [Lee+06; LW12; Yos+12; YGL15; Del+15]. For example, fine-grained entity type classification might assign `sport_team` to "Ottawa Senators" in addition to `organization` and `coach` to "Dave Gill" in addition to `person` as seen in Figure 3.

Using fine-grained entity types, as compared to using traditional coarse-grained types, is expected to be beneficial for a wide spectrum of applications [Sek08]. For example if we want to perform entity linking to the sentence "Michael Jordan is a leading researcher in machine learning.", it would be better to label the mention "Michael Jordan" as `scientist` in addition to `person` in order to avoid the mistake of linking the mention to wrong knowledge base entity such as a basket ball player [SWH15].

1.5 Characteristics of Fine-grained Entity Type Classification

There are several issues that are specific to fine-grained entity type classification.

Hierarchical structure of entity types

First, fine-grained types are typically subtypes of the standard coarse types. For example `artist` is a subtype of `person` and `author` is a subtype of `artist`. This means the type space forms a tree-structured is-a hierarchy. Later we introduce a simple method to exploit this hierarchical nature of the type space that enables the information sharing among different types.

Collapse of the mutual exclusion assumption

Second, the assumption of the standard entity type classification that the labels of entities are mutually exclusive, does not hold. For example Magic The Gathering is both `game` and `product`. This means that it is more natural to formulate the task as a multi-class, multi-label classification problem, rather than a multi-class single label classification as in the case of normal entity type classification.

Context dependent labeling

Third, typically the set of acceptable labels for a mention is constrained to only those that are relevant to the local context [Gil+14; YGL15]. For example in the sentence "Madonna starred as Breathless Mahoney in the film Dick Tracy.", the most appropriate label for the mention "Madonna" is `actress`, since the sentence talks about her role in a film. In the majority of other cases, "Madonna" is likely to be labeled as a `musician`. The importance of the context gives us the motivation to use complex machine learning models to effectively process the contextual information.

Difficulty of preparing annotated data

Forth, compared to the traditional coarse grained entity classification, it is much more difficult to prepare an enough amount of annotated data to train the machine learning models. To tackle this issue, most works has taken the approach of distant supervision [Min+09] to automatically generate the training data [LW12; Gil+14]. We also follow this approach and use freely available public datasets to train our models.

Feature	Decription	Example
Tokens	The tokens of the segment.	"Eton"
Word Shape	The word shape of the tokens in the segment.	"Aa" for "Eton" and "A0" for "CS446".
Part-of-Speech tags	The part-of-speech tags of the segment.	"NNP"
Length	The length of the segment.	1
Contextual unigrams	The tokens in a contextual window of the segment.	"victory", "for", "."
Contextual bigrams	The contextual bigrams including the segment.	"victory for", "for Eton" and "Eton ."
Brown clusters	The cluster id of each token in the segment (using the first 4, 8 and 12-bit prefixes).	"4_1110", "8_11100111", etc.
Head of the segment	The head of the segment following the rules by Collins (1999).	"HEAD_Eton"
Dependency	The Stanford syntactic dependency (De Marneffe, MacCartney, and Manning 2006) involving the head of the segment.	"prep_for:seal:dep"
ReVerb patterns	The frequent lexical patterns as meaningful predicates collected in ReVerb.	"seal_victory_for:dep"

Figure 4: List of features used in FIGER system [LW12]. The mention "Eto" in the sentence"CJ Ottaway scored his celebrated 108 to seal victory for Eton " is taken as a running example.

1.6 Existing machine learning approaches

Next we consider the existing machine learning approach to fine-grained entity type classification. Existing systems for the task have mainly used simple linear models such as logistic regression [Gil+14] and perceptron [LW12], using high dimensional sparse hand-crafted features as input. For example in the model from Ling and Weld (2012), to classify the entity mention in a sentence, they firstly extracted features using the features shown in Fig 4.

Based on those features, a perceptron is used to classify labels.

1.7 Contributions

We address three issues that are limited in the previous works.

Introducing pre-trained word embeddings

First, previous models only used sparse handcrafted features. Models that use sparse features only cannot exploit similarities between features. For example, the words "corpolation" and "campany" are semantically similar whereas the words "corpolation" and "avocado" are dissimilar. Since the occurrence of each word is represented as a binary value in the sparse feature vectors, semantically similar words and dissim-

ilar words are treated equally. Therefore previous models fail to make use of semantic similarities between words, which might undermine the performance.

We incorporate pre-trained word embeddings to allow information sharing between words. Word embeddings aims at quantifying and categorizing semantic similarities between linguistic items based on their distributional properties in large samples of language data [TRB10]. Inspite of the recent successes in incorporating pre-trained word embeddings in NLP systems, there are few works that used those embeddings to fine-grained entity type classification. We use openly available word embeddings extensively and show this feature alone can lead to a significant performance improvement to the previous state of the art model without using any hand-crafted features. Moreover we use both word embeddings and hand-crafted features and observe that they complement each other, resulting in the best performance compared to using only one of those.

RNNs and attention mechanism to sequentially process the context

Second, the previous works fail to incorporate the word ordering information. In those works, the words are represented by a bag-of-words features, thus the information about the word ordering is missed. For example, one can see that a phrase "got a degree from" is indicative of the next words being an educational institution, something which would be helpful for fine-grained entity type classification, but the meaning of this phrase seems not to be successfully represented if the word ordering information is not preserved. Especially, we saw that the context plays an important role in fine-grained entity type classification, but no previously proposed system has attempted to learn to recursively compose representations of entity context.

To address this, we introduce a recurrent neural network to sequentially process the mention context, thus make it possible to use the word ordering information.

Next, while RNNs 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 [Her+15; Roc+15], we circumvent this problem by introducing a novel attention mechanism. 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.

Using the same dataset to train and evaluate

Third, despite using the same evaluation dataset, the literature frequently compare models trained using different dataset.

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.

Hierarchical label encoding

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.

1.8 Structure of the thesis

This thesis is organized as follows. In this section, we explained the objective and background of our research. In section 2, we explain the previous work on fine-grained entity classification and attentive neural networks. From section 3, we introduce the machine learning models. In section 4, the settings and results of experiments are presented. Finally, we conclude our research in section 5.

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 [Min+09] 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 organized 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, Gillick, and Lazic (2015) proposed to map hand-crafted features and labels to embeddings in order 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.

To the best of our knowledge, the first work that utilised an attention architecture within the context of NLP was Bahdanau, Cho, and Bengio (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*.

3 Models

In this section we describe the neural model variants used in this thesis as well as a strong feature-based baseline from the literature.

3.1 Shared settings across all models

First, we explain general characteristics that are used in all the models used in this work including the formulation of the task, computation of label probabilities based on logistic regression, and the inference procedure.

3.1.1 Task Formulation

We pose fine-grained entity classification as a multi-class, multi-label classification problem [TKV09] given an entity mention and its left and right context. We process this input to compute a probability $y_k \in \mathbb{R}$ for each of the K types.

3.1.2 Logistic Regression

Across all the models, we compute a probability $y_k \in \mathbb{R}$ for each of the K types using logistic regression.

In the logistic regression, we only used a weight matrix and did not use a bias vector to compute the pre-activation 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:

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

3.1.3 Inference Procedure

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. The motivation of the former is that it enforces the constraint that each mention is assigned at least one type, while the latter acts as a cut-off.

Variations of the models stem from the ways of computing the input to the logistic regression. We introduce three variants: sparse feature models, neural models, and hybrid models. Each models are explained below.

3.2 Sparse Feature Model

Most previous works in fine-grained entity type classification used high dimensional sparse features as input to the machine learning model. Sparse Feature Model presented here is designed to be similar to those previous models, so that the comparison across different models appropriately fits to the research context.

In this model, we create a binary feature indicator vector $f(m) \in \{0,1\}^{D_f}$ for an entity mention m, 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, Gillick, and Lazic (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 [BNJ03] implementation from the popular gensim software package,[1] 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.

3.3 Neural Models

The neural models processes embeddings of the words of the mention and its context. While both mentions and contexts play important roles in determining the types, the complexity of learning to represent them are different. During initial experiments,

[1] `http://radimrehurek.com/gensim/`

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 ...".

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 $W_y \in \mathbb{R}^{K \times (D_m + D_c)}$:

$$y = \frac{1}{1 + \exp\left(-W_y \begin{bmatrix} v_m \\ v_c \end{bmatrix}\right)} \tag{2}$$

3.3.1 Mention Representation

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

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

Where u is a mapping from a word to an embedding. 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.

Next, we describe the three methods for computing the context representations; namely, Averaging, LSTM, and Attentive Encoder.

3.3.2 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 $l_1, ..., l_C$ and $r_1, ..., r_C$ 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.

3.3.3 LSTM Encoder

Long short-term memory (LSTM) networks have recently applied to various sequential tasks including machine translation and language modeling and showed remarkable successes. In order to handle expressions that have long term dependencies to the category membership of the entity mention, we employ LSTM networks.

For the LSTM Encoder, the left and right contexts are encoded by an LSTM [HS97]. The high-level formulation of an LSTM can be written as:

$$h_i, s_i = lstm(u_i, h_{i-1}, s_{i-1}) \tag{4}$$

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 $l_1, ..., l_C$ from left to right and produces the outputs $\overrightarrow{h_1^l}, ..., \overrightarrow{h_C^l}$. For the right context, the sequence $r_C, ..., r_1$ is processed from right to left to produce the outputs $\overleftarrow{h_1^r}, ..., \overleftarrow{h_C^r}$. The concatenation of $\overrightarrow{h_C^l}$ and $\overleftarrow{h_1^r}$ then serves as the context representation v_c:

$$v_c = \begin{bmatrix} \overrightarrow{h_C^l} \\ \overleftarrow{h_1^r} \end{bmatrix} \tag{5}$$

A complete definition of the LSTM variant used in this work can be found in Sak, Senior, and Beaufays (2014).

3.3.4 Attentive Encoder

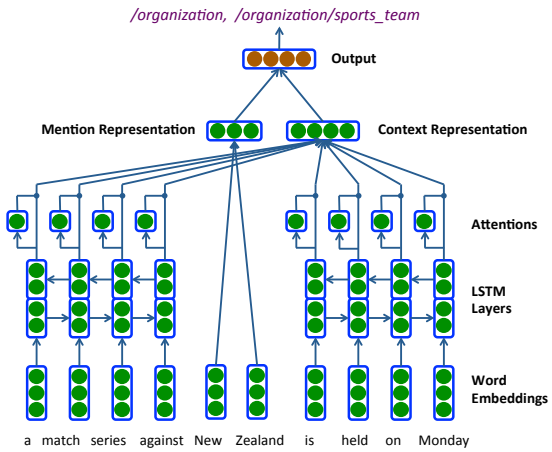

Figure 5: An illustration of the attentive encoder neural model predicting fine-grained semantic types for the mention "New Zealand" in the expression "a match series against New Zealand is held on Monday".

While LSTM networks can handle sequential data, it is still difficult to learn long term dependencies. We concur this problem by introducing a novel attention mechanism. We also hypothesize that by incorporating an attention mechanism, the model can recognize 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 [Gra12] are applied for both the right and left context. We denote the output layers of the bi-directional LSTMs 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, 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 $W_a \in \mathbb{R}^{1 \times D_a}$:

$$e_i^l = \tanh\left(W_e \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix}\right) \tag{6}$$

$$\tilde{a}_i^l = \exp(W_a e_i^l) \tag{7}$$

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

$$a_i^l = \frac{\tilde{a}_i^l}{\sum_{i=1}^{C} \tilde{a}_i^l + \tilde{a}_i^r} \tag{8}$$

These normalized scalar values $a_i \in \mathbb{R}$ are referred to as attentions. Finally, we compute the sum of the output layers of the bidirectional LSTMs, weighted by the attentions a_i as the representation of the context:

$$v_c = \sum_{i=1}^{C} a_i^l \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i^r} \\ \overleftarrow{h_i^r} \end{bmatrix} \tag{9}$$

An illustration of the attentive encoder model variant can be found in Figure 5.

3.4 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 $f(m)$:

$$v_f = W_f f(m) \tag{10}$$

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

$$y = \frac{1}{1 + \exp\left(-W_y \begin{bmatrix} v_m \\ v_c \\ v_f \end{bmatrix}\right)} \tag{11}$$

These models can thus draw upon learnt features through v_m and v_c as well as handcrafted 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.

3.5 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:

$$W_y^T = V_y S \tag{12}$$

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 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 $[1, 0, \ldots]$, `/person/artist` as $[1, 1, 0, \ldots]$, and `/person/artist/actor` as $[1, 1, 1, 0, \ldots]$. This encoding scheme is illustrated in Figure 6.

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.

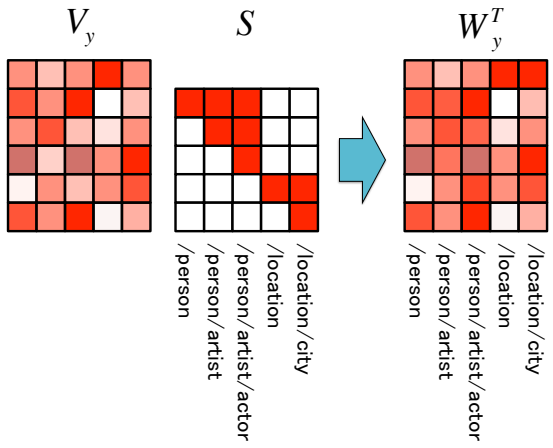

Figure 6: Hierarchical label encoding illustration.

Work	W2M	W2M+D	GN1	GN2
[LW12]	✓			
[Gil+14]			×	
[YGL15]				×
[Ren+16]	✓	×		
This work	✓			

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

4 Experiments

4.1 Datasets

Despite the research community having largely settled on using the manually annotated datasets FIGER (GOLD) [LW12] and OntoNotes [Gil+14] 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[2] 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 generalize for words not appearing in the training set [Roc+15]. 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, Socher, and Manning (2014). For words not present in the pre-trained word embeddings, we use the embedding of the "unk" token.

4.3 Evaluation Criteria

Following Ling and Weld ([LW12]), 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

$$Precision = Recall = \frac{1}{N}\sum_{i=1}^{N}\delta(\hat{T}_i = T_i) \tag{13}$$

- loose macro

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

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

[2] 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.

- loose micro

$$Precision = \frac{\sum_{i=1}^{N} |\hat{T}_i \cap T_i|}{\sum_{i=1}^{N} |\hat{T}_i|} \tag{16}$$

$$Recall = \frac{\sum_{i=1}^{N} |\hat{T}_i \cap T_i|}{\sum_{i=1}^{N} |T_i|} \tag{17}$$

Where N is the total number of instances.

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` [KB14] 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 used dropout [Hin+12] 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 [Mar+15] 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

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	**59.68**	**78.97**	**75.36**
FIGER [LW12]	52.30	69.90	69.30
FIGER [Ren+16]	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	**78.97**	**75.36**
FIGER + PLE [Ren+16]	W2M+D	**59.9**	76.3	74.9
HYENA + PLE [Ren+16]	W2M+D	54.2	69.5	68.1
K-WASABIE [YGL15]	GN2	n/a	n/a	72.25

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

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 boosts performance significantly, indicating that the learnt and hand-crafted features complement each other. 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+16], 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.

4.5.2 OntoNotes

Secondly, we discuss the results on OntoNotes (Tables 5, and 6). Again, we see consistent performance 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. As a reason for these results, 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 to detect entities), 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

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	**51.74**	**70.98**	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	**64.93**
FIGER [Ren+16]	36.90	57.80	51.60

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

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+16]	W2M+D	**57.2**	**71.5**	66.1
HYENA + PLE [Ren+16]	W2M+D	54.6	69.2	62.5
Hand-crafted [Gil+14]	GN1	n/a	n/a	70.01
K-WASABIE [YGL15]	GN2	n/a	n/a	**72.98**

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

explained by the fact that the type ontology used in OntoNotes is more well-formed than its FIGER counterpart. While we do not attain state-of-the-art performance when considering models using different training data. We do note that in terms of F1-scores we perform within 1 point of the state of the art, 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 set of corresponding 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 hesitate to speculate as to exactly why this drop is so dramatic, but similar observations have been made for entity linking [LSW15]. 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 7) and comparing the non-hierarchical and hierarchical label encodings, we can observe that the hierarchical encoding forms clear distinct clusters.

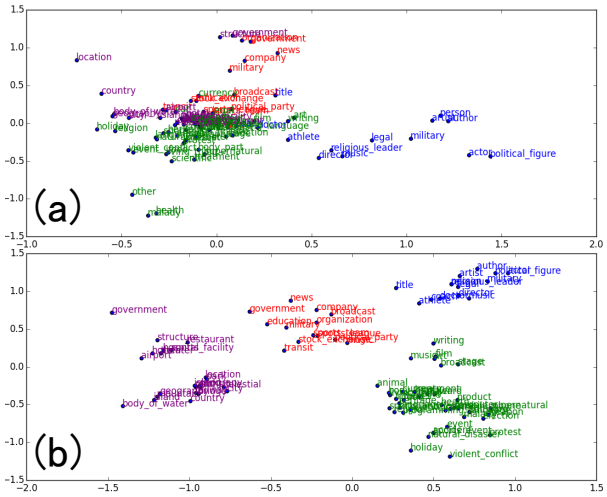

Figure 7: PCA projections of the label embeddings learnt from the OntoNotes dataset where subtypes share the same color as their parent type. Sub-figure (a) uses the non-hierarchical encoding, while sub-figure (b) uses the hierarchical encoding.

4.7 Attention Analysis

4.7.1 Qualitative Analysis

We visualise the values of attentions over the words in the contexts of mentions in Figure 8. Here, all the examples are selected from the development set of FIGER (GOLD). Firstly, we observe the model is able to find out informative words that are located far from the mention, namely, "species" in example 3, and "starring" in example 5. Secondly, it is clear that the model can handle not only a single word but also a group of multipul words that together form relevant expression. Some of the examples of this type is "is a term meaning" in example 1 and "live DVD titled" in example 6. Thirdly, in example 4 we see the model's successful finding of "five-match" that would have served as a crucial information for appropriately classifying the mention "Australlia", which might usually classified as for example "/location/country", into the types "/organization" and "/organization/sports_team". Finally, we point out that the expressions that are treated seriously by the model have a wide syntactic variety such as nouns, verbs, determinants, and propositions.

Id	Sentence	Prediction
0	coast until the fall , when he was appointed a [captain] in the Alabama Cavalry on November 12 .	/title 0.992
1	Vajrayana Buddhism -LRB- the religion of Tibet -RRB- is a [Sanskrit] term meaning " place of peace/tranquility/happiness " .	/language 0.566
2	anthology of ghost stories that was compiled and translated into [Afrikaans] by South African author Francois Bloemhof .	/language 0.524
3	Although [Shepherd 's Beaked Whale] is an exception , most species have only one or	/living_thing 0.897
4	the 1947-48 season to play a five-match Test series against [Australia] .	/organization/sports_team 0.968 /organization 0.881
5	British comedy film starring Googie Withers , Tyrell Davis and [Rex Harrison] .	/person 0.976 /person/actor 0.973
6	Hall in Amsterdam , Netherlands for a live DVD titled [Live from Amsterdam] .	/music 0.309
7	He returned to the Riverina in 1913 and died of [endocarditis] in Hay , survived by his daughter .	/disease 0.979
8	The city lies 85km south-west of [Bobo-Dioulasso] , on the Abidjan - Ouagadougou railway .	/location 0.999 /location/city 0.958
9	Gregorian University in Rome , from where he obtained his [doctorate in canon law] .	/education/educational_degree 0.661 /education 0.523

Figure 8: Visualisation of the attention over several mentions in their context.

4.7.2 Quantitative 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 indicative 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.

Type	Parent	Before	After	Frequent Words
/location	0.319	0.228	0.070	in, at, born
/location/body_of_water	0.420	0.057	0.057	on, east, along
/organization	0.324	0.178	0.119	at, the, by
/organization/sports_team	0.199	0.089	0.02	league, footballer, against
/person	0.246	0.248	0.045	by, president, band
/person/musician	0.213	0.224	0.043	band, group, bands
/person/athlete	0.207	0.197	0.034	coach, champion, match
/person/politician	0.259	0.423	0.032	president, governor, minister
/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.

5 Conclusions and Future Work

In this thesis, 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 reproducability of our work, we publish our code used in the experiments

at https://github.com/shimaokasonse/NFGEC.

Acknowledgements

I would like to express my deep appreciation to Professor Kentaro Inui, Associate Professor Naoaki Okazaki, Professor Akinori Ito, Professor Kengo Kinoshita and Dr. Pontus Stenetorp for their supervision and guidance as the thesis committee of my master thesis.

I would also like to thank Dr. Pontus Stenetorp, Professor Kentaro Inui, and Dr. Sebastian Riedel who are the co-authors of the publications that this thesis is based upon. Without their supervision, constant help, and inspirational ideas, this thesis would not have been possible.

I would also like to thank all the Inui-Okazaki laboratory members, for the valuable supports during my master course period. The daily opinion exchange with them clearly had a positive influence on my way of thinking.

I also cannot omit an expression of thanks to all the members of the Machine Reading laboratry at University College London, "read" by Dr. Sebastian Riedel. Although my stay at the lab was as short as 2 months, the experience I got there was truly irreplaceable.

Finally I would like to express my deepest appreciation to Dr. Pontus Stenetorp who is the closest mentor to this thesis.

References

[Bro+92] P.F. Brown et al. “Class-based n-gram models of natural language”. In: *Computational linguistics* 18.4 (1992), pp. 467–479.

[GS96] Ralph Grishman and Beth Sundheim. “Message Understanding Conference-6: A Brief History.” In: *COLING*. Vol. 96. 1996, pp. 466–471.

[CR97] Nancy Chinchor and Patricia Robinson. “MUC-7 named entity task definition”. In: *Proceedings of the 7th Conference on Message Understanding*. 1997, p. 29.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *the Journal of machine Learning research* 3 (2003), pp. 993–1022.

[TD03] Erik F Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics. 2003, pp. 142–147.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. “Incorporating non-local information into information extraction systems by gibbs sampling”. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 363–370.

[Lee+06] Changki Lee et al. “Fine-grained named entity recognition using conditional random fields for question answering”. In: *Information Retrieval Technology*. Springer, 2006, pp. 581–587.

[Sek08] Satoshi Sekine. “Extended Named Entity Ontology with Attribute Information.” In: *LREC*. 2008, pp. 52–57.

[UB08] Jakob Uszkoreit and Thorsten Brants. “Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation”. In: *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 755–762. URL: http://www.aclweb.org/anthology/P/P08/P08-1086.

[Min+09] Mike Mintz et al. “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*. Association for Computational Linguistics. 2009, pp. 1003–1011.

[TKV09] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. “Mining multi-label data”. In: *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 667–685.

[Car+10] Andrew Carlson et al. “Coupled semi-supervised learning for information extraction”. In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pp. 101–110.

[Fer+10] David Ferrucci et al. “Building Watson: An overview of the DeepQA project”. In: *AI magazine* 31.3 (2010), pp. 59–79.

[TRB10] Joseph Turian, Lev Ratinov, and Yoshua Bengio. “Word representations: a simple and general method for semi-supervised learning”. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 384–394.

[Gra12] Alex Graves. *Supervised sequence labelling*. Springer, 2012.

[Hin+12] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[LW12] Xiao Ling and Daniel S Weld. “Fine-grained entity recognition”. In: *In Proc. of the 26th AAAI Conference on Artificial Intelligence*. Citeseer. 2012.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. 2012.

[Yos+12] Mohamed Amir Yosef et al. “HYENA: Hierarchical Type Classification for Entity Names”. In: *24th International Conference on Computational Linguistics*. ACL. 2012, pp. 1361–1370.

[RMP13] Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. “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*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 627–633. URL: http://www.aclweb.org/anthology/N13-1071.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[Gil+14] Dan Gillick et al. “Context-Dependent Fine-Grained Entity Type Tagging”. In: *arXiv preprint arXiv:1412.1820* (2014).

[JW14] Dragan Milchevski Johannes Hoffart and Gerhard Weikum. “Stics: Searching with strings, things, and cats”. In: *The 37th Annual ACM SIGIR CONFERENCE*. 2014, pp. 1247–1248.

[KB14] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global Vectors for Word Representation.” In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[SSB14] Hasim Sak, Andrew W Senior, and Françoise Beaufays. “Long short-term memory recurrent neural network architectures for large scale acoustic modeling.” In: *INTERSPEECH*. 2014, pp. 338–342.

[Del+15] Luciano Del Corro et al. “FINET: Context-Aware Fine-Grained Named Entity Typing”. In: *Conference on Empirical Methods in Natural Language Processing*. ACL. 2015, pp. 868–878.

[Don+15] Li Dong et al. “A hybrid neural model for type classification of entity mentions”. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press. 2015, pp. 1243–1249.

[Her+15] Karl Moritz Hermann et al. “Teaching machines to read and comprehend”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1684–1692.

[LSW15] Xiao Ling, Sameer Singh, and Daniel Weld. “Design Challenges for Entity Linking”. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 315–328. ISSN: 2307-387X.

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: `http://tensorflow.org/`.

[Roc+15] Tim Rocktäschel et al. “Reasoning about Entailment with Neural Attention”. In: *arXiv preprint arXiv:1509.06664* (2015).

[SWH15] Wei Shen, Jianyong Wang, and Jiawei Han. “Entity linking with a knowledge base: Issues, techniques, and solutions”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2015), pp. 443–460.

[YGL15] Dani Yogatama, Dan Gillick, and Nevena Lazic. “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*. 2015, pp. 26–31.

[Ren+16] Xiang Ren et al. “Label Noise Reduction in Entity Typing by Heterogeneous Partial-Label Embedding”. In: *arXiv preprint arXiv:1602.05307* (2016).

List of Publications

Awards

1. Best Paper Award, The 20th Annual Meeting of the Association for Natural Language Processing, March 2014.

2. Japan Science and Technology Agency Chairman Award, The 3rd Science Intercollegiate, March 2014.

3. Best Paper Award, The 19th Annual Meeting of the Association for Natural Language Processing, August 2013.

International Conferences Papers

1. Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui and Sebastian Riedel. Neural Architectures for Fine-grained Entity Type Classication. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), April 2017. to appear

2. Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui and Sebastian Riedel. An Attentive Neural Architecture for Fine-grained Entity Type Classification. In Proceedings of the 5th Workshop on Automated Knowledge Base Construction, pp. 69-74, June 2016.

3. Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki and Kentaro Inui. Finding The Best Model Among Representative Compositional Models. In Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing (PACLIC), December 2014.

Domestic Conferences Papers

1. Yuto Shinoda, Sonse Shimaoka, and Kentaro Inui. Decorating a Sentence with Emojis by a Neural Network (in Japanese). Proceedings of the 23rd Annual Meeting of the Association for Natural Language Processing, March 2017, to appear.

2. Sonse Shimaoka, Shota Sato, Akira Sasaki, Kazuaki Inada, Satoshi Sekine, and Kentaro Inui. Extracting Future Predictions using Conditional Random Fields (in Japanese). Proceedings of the 7th Text Mining Symposium, pp.57-67, September 2015.

3. Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Generating Phrase Embedding using Dependencies (in Japanese). Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing, pp.1055-1058, March 2014.

4. Sonse Shimaoka, Masayasu Muraoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Distributional Meaning Representation of Words and Phrases with Gaussian Distribution (in Japanese). Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing, pp.1051-1054, March 2014.

5. Sonse Shimaoka, Kazeto Yamamoto, and Kentaro Inui. Learning of Word Embedding by Auto Encoder (in Japanese). Proceedings of the 19th Annual Meeting of the Association for Natural Language Processing, pp.612-619, March 2013.