AN ABSTRACT OF THE DISSERTATION OF

Rasha Mohammad Obeidat for the degree of Doctor of Philosophy in Computer Science presented on June 20, 2019.

Title: Learning with Limited Labeled Data in Natural Language Processing

Abstract approved: _____

Xiaoli Fern

The advent of deep learning models leads to a substantial improvement in a wide range of NLP tasks, achieving state-of-art performances without any hand-crafted features. However, training deep models requires a massive amount of labeled data. Labeling new data as a new task or domain emerges consumes time and efforts and needs domain expertise. As a result, the approaches that address the data scarcity are getting increasing attention in recent years, including, but not limited to, transfer learning, zero-shot learning, and weak supervision. We present three different methods to learn from limited labeled data. In the first work, we present a *Transfer Learning* method to transfer the knowledge between two domains (source and target) with disparate labels. Our approach exploits the relationship between the source and the target labels to enhance the transfer of the learned knowledge. We apply our methods to two NLP tasks: Event Typing and Text Classification. In our second work, we address the problem of modeling the tasks with evolving type ontologies. We present a *Zero-shot Fine-Grained Entity Typing* (ZS-FGET) approach that exploits the *Wikipedia* description of the type to construct the representation of that type. Then, the type can be recognized requiring *zero* training examples. Since FGET deals with a large number of types organized into a hierarchy, Distant Supervision is employed to automatically collect training data, leading to significant label noise. Several methods have been proposed to tackle the problem of FGET, some of them suggest special ways to learn robustly from the data with noisy labels. Most of these methods are evaluated using three publicly available benchmark datasets: FIGER, OntoNotes, and BBN. However, there are some fundamental issues in the empirical evaluation of these methods. Critically, most existing evaluation only reports the overall performance on all types, which can be dominated by the performance on coarse types

and provides very little information regarding how well these methods work for the fine-grained types. This is further compounded by the fact that the testing sets for two of the three benchmarks actually have very poor coverage of the fine-grained types. In our final work, we present a new empirical study that re-evaluates the most recently proposed FGET methods by introducing new testing sets with significantly improved coverage for fine-grained types and examining not only the overall performance but also per-level, type-specific performance. Our analysis of the tested methods reveals new insights about these methods and suggests new directions for future improvement.

Learning with Limited Labeled Data in Natural Language Processing

by

Rasha Mohammad Obeidat

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 20, 2019
Commencement June 2020

Doctor of Philosophy dissertation of Rasha Mohammad Obeidat presented on June 20, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Rasha Mohammad Obeidat, Author

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor Prof. Xiaoli Fern for her continuous support of my Ph.D. research, motivation, and encouragement. I could not have imagined having a better advisor for my Ph.D. study at Oregon State University. My special thanks go to Prof. Prasad Tadepalli, who provided invaluable feedback about my research during my Ph.D. Journey. I'd also like to extend my gratitude to the rest of my Ph.D. committee: Prof. Liang Huang, Prof Stephen A Ramsey, Prof. Brett Tyler, and Prof. Sushma Naithani, for their insightful comments and continuous support.

I would like to express my deepest gratitude to all my friends, Evgenia, Noor, Hameed, Dima, Hamed and Reza for being a constant source of support and motivation.

This work would not have been accomplished without the love and support of my family. I would like to thank my parents, and my brothers, whose love and guidance are with me in whatever I pursue. I cannot begin to express my thanks to my soul mate, my dearest husband and my best friend, Ahmad, for your love, profound belief in my abilities, and the patience that cannot be underestimated. My kind gratitude goes to my children, Yara, Osama, Dana, and Muhammad, who are my source of unending inspiration.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

This work is dedicated to my beloved husband, Ahmad, and my children. You have made me stronger, better and more fulfilled than I could have ever imagined. I love you to the moon and back.

Chapter 1: Introduction

Traditional supervised learning methods typically require a significant amount of training data to learn effectively. It is also commonly assumed that the training data must include instances of all the classes that can be potentially tested on. However, obtaining sufficient labeled training data can become a key development bottleneck in supervised machine learning especially due to the emerging popularity of the deep neural models. Deep learning (DL) has become the dominating technique for NLP, achieving the state-of-art results for most tasks in the span of the past few years [48]. Deep learning models automatically learn high-quality and task-specific representations and thus reduce the need for feature engineering. Nevertheless, deep models are more complex than traditional models, have considerably more parameters, and thus require more labeled training data.

The reliance on massive amounts of hand-labeled training data is often not feasible due to several reasons. First, it is expensive in term of time, human efforts and the need for domain expertise. Also, the systems built on heavily annotated data are not scalable to deal with new domains or with larger sets of types. These systems are usually incapable of handling domain shift or ever-growing type taxonomies. In real-world applications, it is a common scenario that new types may emerge over time whereas existing types may be refined into finer-grained classes, or become obsolete. Re-annotating the labeled data partially or entirely to accommodate the new types is undesirable or even impractical. Moreover, annotating data manually can be error-prone when the task deals with a large number of types; as it is hard for a human to distinguish more than one hundred types consistently. [58].

Driven by the need to alleviate the supervision bottleneck, learning paradigms that address the data scarcity problem become increasingly popular. They can be divided, by the data limitations they address and the type of data they leverage, into the following categories: (1) *Transfer Learning* [54, 69, 60] aims to enhance learning in a new task or domain through the transfer of knowledge learned from one or more related tasks or domains; (2) *Zero-shot* and *few-shot learning*[71, 76, 30, 81] try to recognize classes with a few or zero training examples; (3) *Weakly supervised learning* [97] attempts to construct predictive models effectively from data collected via *Weak Supervision*, in which low-cost approaches (e.g., Distant supervision [45], Crowd-

sourcing [33, 5]) are employed to collect labeled data, with the caveat that they tend to introduce low-quality annotations with significant label noise.

In this thesis, we present three works to address the problem of learning from limited training data. First, we present a *Transfer Learning* method that exploits the label structure to transfer the learned knowledge between tasks that have related but disparate label sets. Second, We propose a *Description-based Zero-shot entity typing* approach that utilizes a type description to recognize new types without the need for collecting training examples. Finally, we highlight several critical flaws in the evaluation of Fine-grained Entity Typing (FGET) in prior work, which relies on benchmark datasets that are originally collected by distant supervision, and present a rigorous re-evaluation of recently proposed models for this task.

1.1 Overview

Transfer Learning. Traditional supervised learning typically tries to learn a mapping from the input space to the output (label) space, with a common assumption that the training data and the unseen test data are drawn from the same distribution. It also assumes the existence of a sufficient amount of labeled training data that includes examples for all classes (that one might encounter during testing). If one of these assumptions doesn't hold, supervised learning fails to learn a robust model. Collecting new data each time a new domain or task is introduced, or a new class is added is an undesirable or infeasible process. Transfer learning provides a solution to this problem by leveraging the knowledge extracted from one or more source tasks or domains and transferring the learned knowledge to a target task or domain. The desire of making the machine, like a human, be able to intellectually apply previously learned experiences and knowledge in solving new problems motivates the study of transfer learning. The difference between traditional supervised learning and transfer learning is illustrated in Figure 1.1.

We start first by defining the domain and the task following the definition by [54]. A domain $\mathcal{D}$ consists of the feature space $\mathcal{X}$ and the marginal probability distribution $P(X)$ (i.e., $\mathcal{D} = \{\mathcal{X}, P(X)\}$) where $X \in \mathcal{X}$. Given a domain $\{\mathcal{X}, P(X)\}$, a task consists of the label space $\mathcal{Y}$ and an objective predictive function $f(.)$ ($\mathcal{T} = \{\mathcal{Y}, f(.)\}$). Transfer learning aims to enhance the learning of the target prediction function $f_t : \mathcal{X}_t \rightarrow \mathcal{Y}_t$ by utilizing the source domain $\mathcal{D}_s$ and the task $\mathcal{T}_s$. Base on different settings of $\mathcal{D}_s$, $\mathcal{D}_t$, $\mathcal{T}_s$, and $\mathcal{T}_t$, We can define the following transfer learning scenarios as follows:

- $\mathcal{X}_s \neq \mathcal{X}_t$. The feature space of the source and the target domains are heterogeneous. One

example is transferring learning between textual datasets form different languages, this is called *Cross-lingual Transfer Learning* [12, 86, 96]. Another example is transfer learning between domains of different modalities such as image vs. text [98, 10]

- $P(X_s) \neq P(X_t)$. The marginal probability distributions of source and target domains are different, .e.g, Part of Speech Tagging (POS) of newswire and biomedical domains. This is also known as domain adaptation. Significant research has been devoted to domain adaptation, which typically assumes that the source and the target tasks deal with the same label space [4, 14, 53, 20, 25].

- $P(Y_s|X_s) \neq P(Y_t|X_t)$. It assumes that the feature spaces and the marginal distributions of the source and the target domains are the same, but the class distributions are imbalanced. This is known as *imbalanced domain adaptation* [72]. Approaches that minimize the conditional distribution discrepancy between domains are usually used to handle the class imbalance [72, 44, 75].

- $\mathcal{Y}_s \neq \mathcal{Y}_t$. The label spaces between the two tasks are different, regardless of whether the source and target domains are the same or not. Examples include two text classification tasks that use various label schemes (e.g., 20Newsgroups vs. Reuters) [46], or transfer the learning from data set annotated with POS tags to a Named Entity Recognition dataset [38]. If the goal is to improve the learning for both domains, then this is called *Multi-task learning*[39, 80, 61, 51]. If we focus on enhancing the target task, then it is referred to as *Model Transfer* [74]. A subcategory of Model Transfer that uses label space transformation between the source and the target domains [3, 28]

In this thesis, we focus on the problem of transfer learning for NLP tasks where the target task and the source task share the same domain but have different label spaces. Specifically, motivated by the practical applications in NLP, our investigation focuses on transfer learning under the scenarios where there are known structures between the source and target labels.

Zero-shot learning aims to recognize classes whose instances have not been seen during training. An increasing number of zero-shot learning methods are proposed to alleviate the need for annotating additional training examples as a new class is introduced. Zero-shot learning has been successfully applied to a wide range of Computer Vision tasks [32, 59, 17, 82], and it is also becoming an important focus for the NLP community [65, 23, 24, 95, 66]. Zero-shot

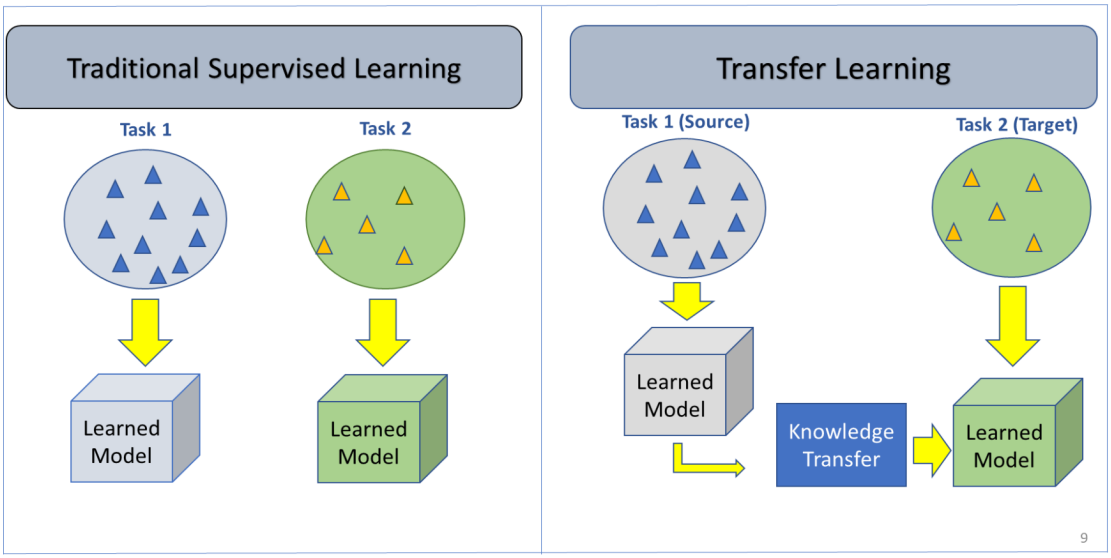

Figure 1.1: Traditional Supervised Learning vs.Transfer learning

learning can be considered a special case of transfer learning where $\mathcal{Y}_s \cap \mathcal{Y}_t = \Phi$. The source domain is assumed to contain manual annotations of seen types while on the target side, we have unseen types that do not have any training examples. Instead, we assumed the availability of an auxiliary source of information (e.g., descriptions, attribute vectors, annotation guidelines) that encodes the semantics of the seen and unseen classes. The auxiliary information source is then used to learn a shared concept semantic space of all types along with alignment between the source examples and the seen classes. Afterward, the learned knowledge can be transferred from seen to unseen types. The second work of this thesis follows this general scheme for the problem of zero-shot classification of fine-grained entity types and learns the representations for the types based on their Wikipedia descriptions.

Weakly supervised learning is a general term that covers a variety of techniques that are developed for the goal of learning effectively from noisy data collected under a weak supervision scheme. Weak supervision refers to the general scenario where the labels of the training data are acquired at a substantially lower cost, resulting in annotations that are of considerably lower quality or lower resolution.

Several methods have been introduced to obtain weakly labeled data including but not lim-

ited to 1) Distant supervision where the unlabeled data are heuristically mapped to an external knowledge base[43, 21, 90], 2) Crowdsourcing is another cost-saving way to collect training data by recruiting a large group of independent non-expert workers to provide labels for training data based on their own judgments [11, 97]. 3) Pre-trained models on different domains combined with heuristics are used to obtain labels for unsupervised data.

Another type of week supervision is obtained by leveraging higher-level inputs from Subject Matter Experts (SMEs). Instead of preparing detailed annotation guidelines and asking the experts to follow the guidelines and annotating every single example in the training data, SMEs can infuse their knowledge into the model in the form of hand-written heuristics that are translated into a programming script to be used in annotating data points automatically. Alternatively, SMEs can provide a limited number of high-level annotations, such as annotating samples with coarse types instead of going over all of the fine-grained types or provide annotation at a coarse resolution and use multi-instance learning to infer finer resolution labels [97]. For example, most image categorization datasets only have image-level labels while the task is to recognize and classify the objects in the image. Getting image-level annotation needs far fewer annotation efforts than acquiring the object-level ones. SMEs can be also a part of an active learning cycle where they are asked to iteratively annotate critical data points, such as the points laying on the decision boundary of a model trained on automatically generated data [97]. More recent techniques have been proposed to manage multiple sources of weak supervision that have different levels of accuracy [2].

The last work of this thesis presents a systematic empirical examination of the problem of fine-grained entity typing for which distant supervision is not only the primary source of training data, but also the primary source of technical challenge in learning [40, 45, 90, 58].

There are also other related lines of research aimed at increasing data efficiency in learning, including *Semi-supervised learning* [60, 52], and *Data augmentation* [77, 26]. Semi-supervised learning assumes the availability of a small labeled training set and a much larger unlabeled data set. The goal is to exploit the unlabeled data in addition to labeled data to improve learning performance, it differs from transfer learning in that both labeled and unlabeled data have the same marginal feature distribution. Data augmentation is a strategy that is used increase the diversity of the training data without collecting new data, instead, labeled distribution is extended to extend cover set of transformation $\{t_i(x)\}_{i=1}^{m}$ of the labeled instance x, and then set $P(y|x)$ to $P(y|t_i(x))\ \forall i \in 1, ..., m$. These topics are closely related to weak supervision, but they are not the focus of this thesis.

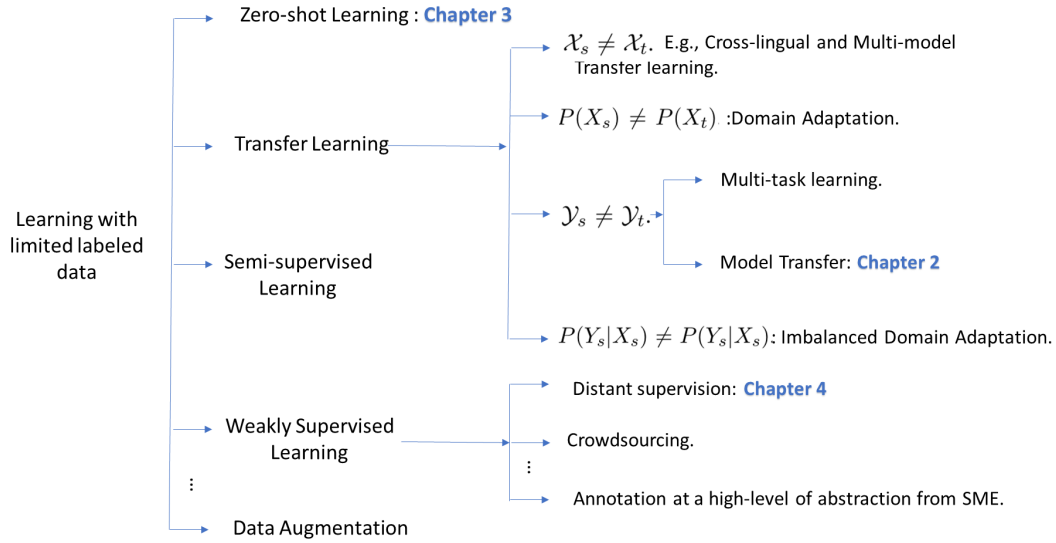

Figure 1.2: Contributions in this work.

1.2 Thesis outline

In Chapter 2, We study the transferability of neural networks between related NLP tasks that differ in their labeling schemes. We present a method that exploits the relationship between the target and source labels as an additional source of information to enhances the learning transfer. In our study, we consider two NLP tasks: Event Typing and Text Classification. For each task, we use two different datasets to serve as the source and the target tasks respectively. The datasets are selected such that they have different but related label spaces.

Our method utilizes a bipartite graph that represents the source and target label connections (which can be 1-1, 1-n, n-1 or n-m) to assign each source example to a target label set. Because the mapped source examples may have incorrect types within the assigned candidate label set, we formulate the learning as a weak supervision problem and introduce a weak supervision objective for training a neural network for the target domain on both the target and the mapped source examples. We compare it with alternative previously used weak supervision objectives [58, 1]. We test our method and compare it with traditional fine-tuning based transfer learning methods and multi-task learning under several label-relationship scenarios and data sizes. Our experimental results show that our method compares favorably to the standard deep transfer learning approach under several scenarios.

Chapter 3 considers the problem of zero-shot fine-grained entity typing (FGET). Fine-grained Entity typing (FGET) is the task of assigning a fine-grained type from a known type hierarchy to the entity mentions in the text. As the type taxonomy often evolves over time, it is desirable for an entity typing system to be able to recognize new types without additional training data, motivating zero-shot learning for FGET. In this thesis, we present a *zero-shot* entity typing approach that leverages the Wikipedia descriptions of the types to build a distributed semantic representation of the types. During training, our system learns to align the entity mentions and their corresponding type representations on the known types. At test time, any new type can be incorporated into the system given its Wikipedia descriptions. We evaluate our approach on FIGER, and BBN, two public benchmark FGET datasets. Because the existing testing set of FIGER covers only a small portion of the fine-grained types, we created new testing set by manually annotating a part of the noisy train data. Our experiments demonstrate the effectiveness of the proposed method in recognizing novel types that are not present in the training data.

Distant supervision in conjunction with a knowledge base (KB) is extensively used to automatically generate labeled training data for the task of FGET. However, the types obtained from the KB are usually noisy as they are assigned without taking the mention's local context into account. Several methods have been proposed to tackle FGET, most of them are evaluated using three publicly available benchmark datasets: FIGER, OntoNotes, and BBN. These methods vary in different aspects, one aspect is how they treat the typing noise. Typically, FGET systems either use pruning heuristics to clean the data from the noisy mentions [19], or use weak supervision to learn from the mentions with the noisy types [1, 58, 83], or even neglect the typing noise entirely and assume that all mentions are assigned clean type sets [87, 63]. However, there are some fundamental issues in the empirical evaluation of these methods. Critically, most existing evaluation only reports the overall performance on all types, which is dominated by the performance on coarse types and provides very little information regarding how well these methods work for the fine-grained types. This is further compounded by the fact that the testing sets for FIGER and OntoNotes have very poor coverage of the fine-grained types. For example, about 50% of test mentions for *OntoNotes* only have the coarse type "other". In chapter 4, we present a new empirical study that re-evaluates the most recently proposed FGET methods by introducing new testing sets with significantly improved coverage for fine-grained types and examining not only the overall performance but also per-level, type-specific performance. The ability of different methods to learn from noisy data effectively is one of the dimensions that we focus on in our analysis. We noticed the efficiency of the methods to learn from noisy data can be masked

by reporting the overall performance solely. Chapter5 concludes our work and provide a brief discussion of potential directions for future improvement.

Chapter 2: Deep Transfer learning for NLP tasks with disparate labels.

2.1 Abstract

Deep neural Networks hold strong promise for transfer learning when the source and target domains have disparate labels because the hidden layers can be easily transferred independently from the output layer connecting to the labels. This method, however, ignores the relationship between the target and source labels, which can provide valuable information to enhance the learning transfer. We investigate how to best exploit the label structure between domains by mapping the source examples to corresponding target labels. We examine several weak supervision objectives for training neural networks for the target domain on both the target and the mapped source examples. We apply our methods on two NLP tasks: Event Typing and Text Classification. We study the performance of the proposed method against the standard neural transfer techniques under several systematic label relationship scenarios.

2.2 Introduction

In NLP supervised learning requires a significant amount of training data to learn effectively. *Transfer learning* aims to leverage the knowledge from one or more *source* tasks or domains to improve the learning performance in a related *target* task or domain. It is important because it reduces the need to annotate a large amount of data each time a new task or domain emerges. Significant research has been devoted to *Domain Adaptation* [14, 4], which assumes that the source and target domains use the same label set but with different input distributions.

In many real-world applications, however, the label sets may evolve from one task to another — labels may be merged or partitioned into finer-grained concepts, and new labels may be introduced, and old labels may phase out. Transfer learning with disparate label sets thus becomes an important problem, which unfortunately cannot be solved by directly applying standard domain adaption techniques.

Few works have been proposed to address this problem. Kim et al. [28] propose a solution to this problem by deriving a mapping between the source and the target labels to reduce the

problem to the standard domain adaptation settings. They learn an embedding for each label in both domains induced by Canonical Correlation Analysis (CCA). Then, they use the label embeddings to create a mapping by finding the nearest neighbor of each label type. However, their method is just applicable if the relationship between the target labels and the source labels are 1-1 or n-1. In other words, if the source-to-target label relationship is n-1 or if the labels overlap, then this method will produce erroneous mapping. For Example, assume that the source has the label type ”life,” and the target has it’s the fine-grained label types ”be-born,” ”die,” ”marry,” and ”divorce.” Then, all examples under ”life” will be mapped to one of its four subtypes, the one that has the most similar label embedding. Bhatt et al. [3] assume that no labeled data are available in the target domain, and use multiple source domain with different label schemes to induce labels for the target.

With the emerging popularity of the deep neural networks for NLP applications, one might expect that having a disparate label set should not prevent the knowledge transfer across domains because the feature extraction layers, e.g., convolutional neural networks (ConvNet), are independent of the number and type of labels. One can simply take a deep network that is trained on the source domain and replace the final fully connected output layer with a target-specific output layer and train its weight on the target domain data.

Based on how the source and the target examples are scheduled in the learning process, neural transfer learning can be classified into parameter initialization and multi-task learning. In parameter initialization, a model is trained on the source, and then, the last output layer is removed and the model parameters (e.g. word embeddings, convolutions) used to initialize the target model. A target specific output layer is added to the target model. The transferred weights are either tuned (aka Fine Tuning) or frozen, i.e. used as a fixed feature extractor for the target domain. Multitask learning means the source and the target models are trained simultaneously. They shared some or all of the hidden layers while keeping the task-specific output layer.

A recent study [47] demonstrates that the transferability of DNN for NLP tasks depends highly on how the source and target tasks are related semantically. However, they only considered the simplified scenarios in which the source and target tasks that are either identical (e.g., two different sentiment analysis datasets), or semantically completely different but using the same neural architecture. Transferability of DNN across semantically related domains that use different but related labeling schemes is still an open question, which is the subject of this work.

In this work, we are interested in the transferability of DNN across semantically related domains where the labeling schemes are different but related. We show that the relationship

between the source and target labels, if known, can provide tremendously useful information to enhance the knowledge transfer. Specifically, given a bipartite graph representing the source and target label relations (which can be 1-1, 1-n, n-1 or n-m), we assign each source example to a target label set. Learning can then be performed on both source and target data with weak supervision objectives. We study how source-to-target label relationships affect the performance of the standard deep transfer learning methods in comparison with the proposed weak-supervision methods on two NLP tasks: event typing (ET) and text classification (TC).

2.3 Problem Statement

We are given data from two related domains, the source domain $S = \{x_s^i, y_s^i\}_{i=1}^N$ and the target domain $T = \{x_t^i, y_t^i\}_{i=1}^M$, where $y_s^i \in Y_s$ and $y_t^i \in Y_t$, N and M are the number of training examples in S and T respectively. We assume that the two domains have a shared input space but map them to different (yet related) label spaces, whose relationship is represented by a bipartite graph B. An edge of B between two labels a_s and b_t can be interpreted as "an input example belonging to the source class a_s can possibly belong to the target class b_t." B is designed to reflect human knowledge about the relations between the source and target labels. We assume that $M << N$ and the goal of transfer learning is to leverage the data-rich source domain S to improve the performance on the data-scarce target domain T.

2.4 Transfer Learning by Weak Supervision

The proposed method aims to benefit from the label structure between the source and target domains S and T represented by B. Specifically, the bipartite graph B defines a mapping from each source label y_s to a (possibly empty) set of target labels: $B_{st}(\cdot) : Y_s \rightarrow 2^{Y_t}$. A source domain instance $\{x_s^i, y_s^i\}$ can thus be mapped to the target domain labeled with an ambiguous set of labels $\mathcal{Y}_t^i = B_{st}(y_s^i)$ (see also Figure 2.1). If $\mathcal{Y}_t^i = \emptyset$, it is ignored, which helps to avoid the *Negative Transfer* that may come from S. Let $S_{map} = \{x_s^i, \mathcal{Y}_t^i\}_{i=1}^{N'}$ be the set of source examples mapped to non-empty target label sets. We can then add S_{map} to the T to created an augmented training set $T_{aug} = T \cup S_{map}$. The mapped examples often have $|\mathcal{Y}_t^i| > 1$, which provides a candidate label set ambiguous for learning.

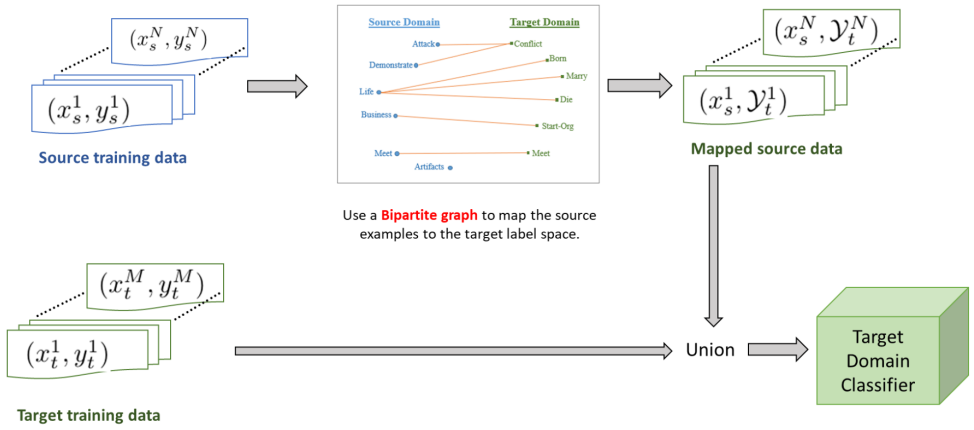

Figure 2.1: An illustration of the transfer learning by weak supervision.

Weak Cross-Entropy (WeakCE). Mapped examples often have $|\mathcal{Y}_t^i| > 1$. For such examples, $\mathcal{Y}_t^i$ can be viewed as providing a candidate label set, which is ambiguous for learning. To learn from T_{aug}, our method must account for the ambiguity for those mapped examples. To this end, we extend the classic Cross-Entropy loss and define a weak Cross-Entropy (WeakCE) loss for $x_i \in T_{aug}$ as follows:

$$loss_{WeakCE}^i = -\frac{1}{|\mathcal{Y}_t^i|} \log \sum_{y_j \in \mathcal{Y}_t^i} P(Y = y_j | x^i) \tag{2.1}$$

where $P(Y = y_j|x^i) = \text{Softmax}(f_{y_j}(x^i))$, and $f_{y_j}(x^i)$ is the score assigned to class y_j for instance x^i by the deep learning model. Minimizing this loss maximizes the probability that the predicted label for x_i is in $\mathcal{Y}_t^i$, leaving it to the model to decide which label in $\mathcal{Y}_t^i$ to assign high scores based on x_i. We also weigh each example by $\frac{1}{|\mathcal{Y}_t^i|}$ because the larger the size of the candidate label set, the less informative is the example.

Partial-label Pairwise Ranking(PL-rank). Alternatively, we consider partial-label pairwise ranking loss function (similar to [58, 87]) that tries to rank the *best* label inside $\mathcal{Y}_t^i$ higher than all labels in $\bar{\mathcal{Y}_t^i}$. Partial-label ranking (PL-rank) loss function for x_i is defined as follows:

$$loss_{PL\text{-}rank}^i = \sum_{y_k \in \bar{\mathcal{Y}_t^i}} max(0, 1 - f_{y_j^*}(x^i) + f_{y_k}(x^i))$$

$$y_j^* = argmax_{y_j \in \mathcal{Y}_t^i} f_{y_j}(x^i)$$

Multi-label Cross Entropy (MLCE). We also consider a multi-label extension of cross-entropy that weighs all labels in $\mathcal{Y}_t^i$ equally assuming that all of them are correct:

$$loss^i_{MLCE} = -\sum_{y_j \in \mathcal{Y}_t^i} \frac{1}{|\mathcal{Y}_t^i|} \log P(Y = y_j | x^i)$$

Note that both WeakCE and MLCE reduce to the classic cross entropy loss for the target training examples with unambiguous label.

Training details. It is crucial to control the strength of the weak training signal from S_{map}, especially because S_{map} is often overwhelmingly larger than T. Blindly treating the source and the target examples the same during training can overwhelm the learning process with source examples, leading poor performance for target testing.

We use two training strategies to avoid the "source-take-over". First, we balance the source and target contributions to the overall loss with a hyberparameter λ:

$$Loss = \sum_{i \in T} loss^i + \lambda \times \sum_{j \in S_{map}} loss^j$$

Secondly, batches are formed with equal number of examples from S_{map} and T. Due to the significantly smaller size of T, this will lead to completing multiple epochs on T before finishing one epoch on S_{map}, ensuring that our model is always trained with a current focus on T.

2.5 Deep Transfer Learning Methods

In addition to *Target Only (TO)* which is training a model only using the target data, we examine the following deep transfer learning methods: *Fine Tuning (FT)* trains a model on the source domain [1] and use it to initialize the target model and fine tune all the parameters including the embeddings and the convolutions. We also consider *FreezC* which is a variant of FT that freezes the convolution layer. Multitask learning(MULT) generally applied by sharing the embeddings and the feature extraction layers between the source and the target domain, while keeping several

[1] We use a subset of S as the development set and train on the rest.

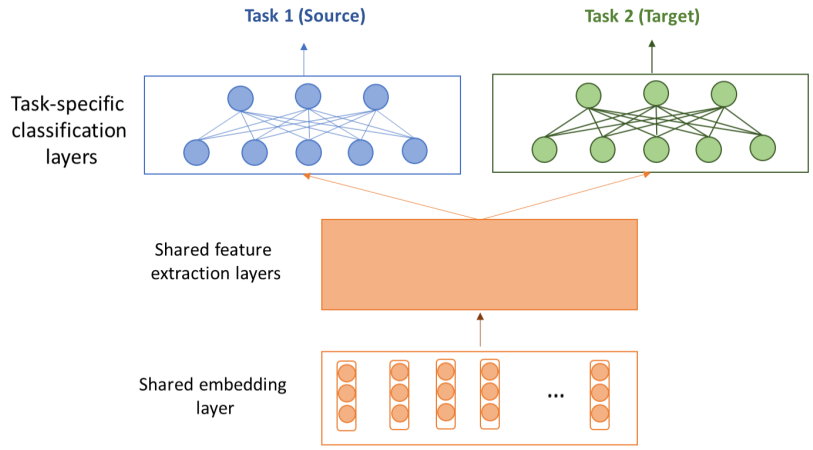

Figure 2.2: Parameter sharing for multi-tasking (MULT) in deep neural networks.

task-specific output layers as shown in figure 2.2, then the network is trained simultaneously for both domains. The training process is similar to the one we adopted for our method. The loss in MULT is be computed as:

$$Loss = \sum_{i \in T} loss^i + \lambda \times \sum_{j \in S} loss^j$$

2.6 Tasks and models

We consider two different NLP tasks: Event Typing (ET) and Text Classification (TC). Below we briefly describe the two tasks and the models used.

Event Typing. The input to ET is a span of text and the goal is to assign an event type label. Here we bypass the upstream extraction step and simply use the text span around the ground truth trigger words as the input text and train a model to predict its event type. We adopt the convolutional network architecture (CNN) by [50]. Specifically, each input example is a text span of 31 tokens centered around the event trigger (padding is performed if necessary). Each token is represented by a word embedding [55] and a position embedding. The representation of the text span is then passed to a convolutional layer, a max-pooling layer and a softmax layer at the end to perform the classification. In the convolution layer, three sets of feature maps (filters)

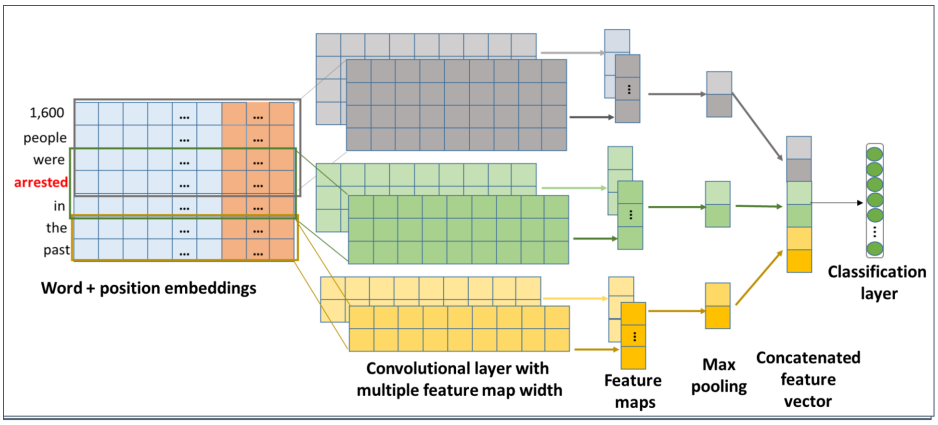

Figure 2.3: Convolutional Neural Network for Event Typing.

are used for the convolution operation. Each set corresponds to some window size. Figure 2.3 illustrates the adopted CNN Architecture.

Text Classification Given a span of text, the goal is to assign a topical label to the text. We use a similar architecture for this task but without position embedding, and the text inputs are complete sentences. The detailed architecture can be found in [27].

2.7 Experimental Setup

Datasets. We conducted experiments on two datasets for each task. For **Event Typing**, we use ACE2005 as the source domain and Rich ERE 2015 as the target domain. ACE 2005 contains eight event types and 33 sub-types. Rich ERE 2015 (RichERE) contains nine event types and 38 sub-types(see figure 2.4 for the type hierarchies). To examine the ability of our method to handle different types of source-to-target relationships, we created four synthetic scenarios where the source and target domains use coarse, fine and mixed event types respectively. All the bipartite graphs of the four scenarios presented in figure 2.4. Since transfer learning assumes data scarcity on the target side, we sample several *small* train/development sets from the original RichERE training set. Specifically, we randomly sampled five different train/development set with 300 examples for training. For each sample, we run all the models with five random initializations. All results are reported on the target (RichERE) testing set which contains 402 examples, averaged over the 25 runs. We repeat the process with training sizes 400, 500 and 600. Each time the

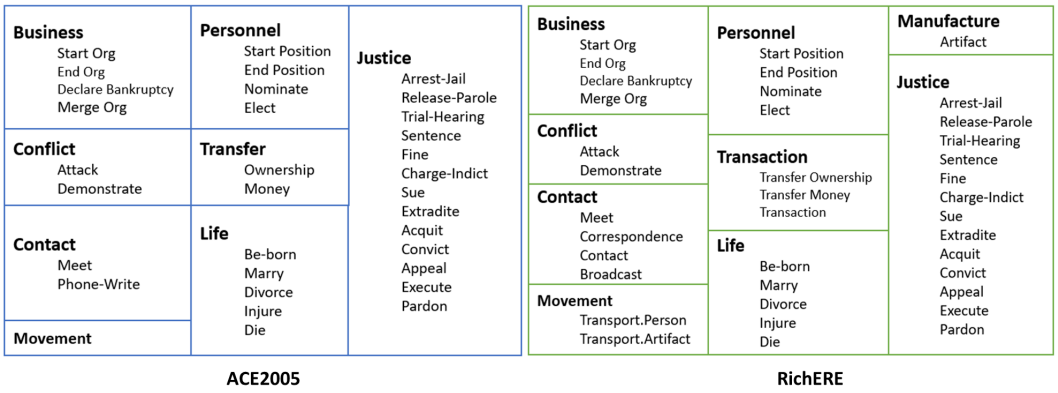

Figure 2.4: Type Hierarchies of ACE2005 and RichERE, the coarse types are in boldface.

development set size is half of the training size. ACE2005 training has 4420 examples.

For **Text Classification**, we use two publicly available datasets: News (source) and Google-snippets (target). Examples of both datasets are short snippets. News contains seven classes, while Google-snippet has eight. Labels in these datasets are not the same but highly related with no clear precise mapping. Since Google-snippets is fairly large, we randomly sample 10 training examples and 5 dev examples for each class. This is repeated five times to generate five different train/development sets. Each time we run all the models with five different initializations, leading to 25 random runs. The reported testing performances are averaged over the 25 runs. News training has 25023 examples. Figure 2.6 presents the bipartite graph that maps the types between News and Google-snippets.

Training and evaluation We evaluated the performance using the prediction accuracy, and macro-averaged recall, precision, and F_1. For every model, we tune all hyper-parameters using a fixed development set. For each experiment, we report the testing results of the model with the best Accuracy on the development set. We evaluated the performance using the prediction accuracy, and macro-averaged recall, precision, and F_1. For the CNN architecture, we use filters of width sizes 3, 4, 5, to generate feature maps. We utilize 100 feature maps for each window size. The window size for triggers is set to 31 in ET while it is the length of longest sentence in TC padded if necessary. We use the pretrained word embeddings word2vec with 300 dimensions [55] and we adopt position embedding of size 50. We employ Adam optimizer with $lr = .001$, dropout rate = 0.5, the mini-batch size = 64. We train each model for 12 epochs.

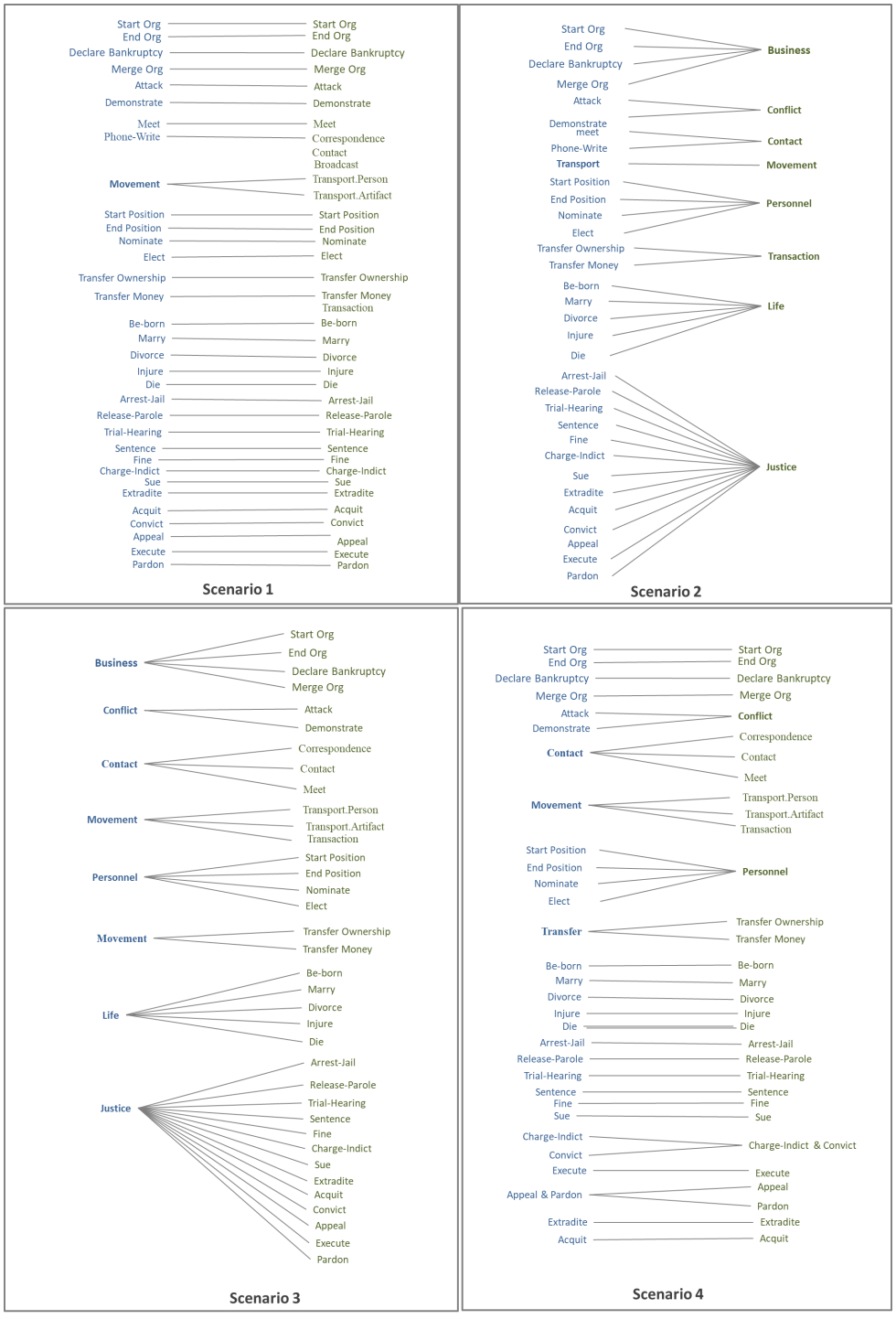

Figure 2.5: The bipartite graph that defines the mapping between labels in News dataset (source) and Google-snippets dataset (target) used in Text Classification.

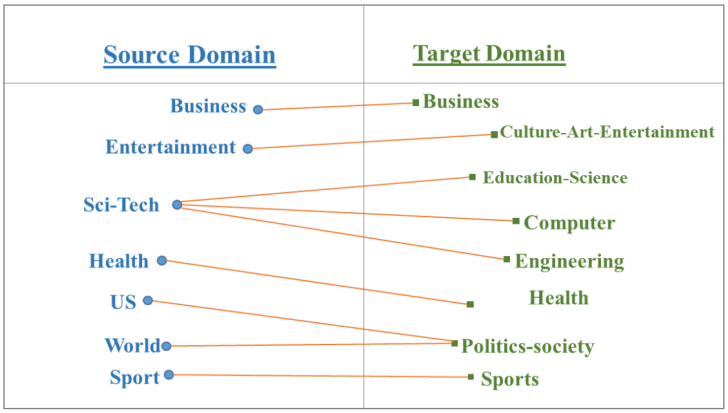

Figure 2.6: The bipartite graph that defines the mapping between labels in News dataset (source) and Google-snippets dataset (target) used in Text Classification.

2.8 Results and Discussion

Results on ET Table 2.1 presents the results on Entity Typing for four scenarios using a target training set of 300 examples. Specifically, in scenario 1, the bipartite graph B defines a fine-to-fine source-types to target-types mapping. This mapping is mostly 1-to-1 except for one type in the source ("movement") that is mapped into two subtypes in the target ("Transport.Person " and "Transport.Artifact"). In Scenario 2, B maps multiple fine-grained source types to a single coarse target type (n-to-1). Scenario 3 is the inverse of Scenario 2 (1-to-n). Finally, the mapping for scenario 4 is mixed, containing 1-to-1, 1-to-n and n-to-1 mappings.

From the table, we note that all of the transfer learning methods provide a significant improvement over Target Only (TO) in most cases.

We also observe that weak supervision with WeakCE consistently outperforms the other methods for scenarios 1 and 4 and get comparable performance to PL-rank and MLCE for scenarios 2 . Specifically, we see that WeakCE outperforms the best baseline methods by a margin of 1-3%, for these three scenarios, especially achieving a substantial gain in recall (4-6%). For scenario 1 and 2, MLCE performs comparably to WeakCE, but is substantially worse for scenario 4. This is because in scenario 4 S_{map} contains some ambiguous cases, the assumption made by MLCE is that all labels in the ambiguous candidate set are correct for the given source instance, which can potentially mislead the training.

In scenario 1, WeakCE outperforms MLCE. Although the relationship between the source

and the target labels are 1-1, there is one 1-n relations, e.g., the label *transport* in ACE2015 is mapped to two RichERE labels (*transport-person* and *transport-artifacts*). MLCE considers both labels as true labels and treats them equally while only one of them is the true label. In Scenario 2, the three objectives produce comparable performance since scenario 2 deals with n-1 label relationships, thus the candidate label set assigned to a mapped source-example has only one label, which means that both WeakCE and MLCE become identical to the standard cross entropy.

Another interesting observation is that WeakCE outperforms PL-rank in Scenario 3 and 4, we think this is because WeakCE is more capable of modeling the type correlation. E.g., In Scenario 3, a source example of the type *life* is mapped to the label candidate set {*be-born, die, marry, divorce* and *injure*}. If the correct label is *marry*, one intuitively can expect that *divorce* should get higher score than *be-born, die*, and *injure*. However, PL-rank tries to score the *best* label given the event mention and its context higher than incorrect labels ignoring the varying level of correlation between the labels in the candidate set, while WeakCE accounts for these correlations by leaving it for the training process to distribute the probability between the candidate labels.

Scenario 3 is the most challenging scenario for weak supervision since every mapped example is assigned an ambiguous candidate set, which can be overly large. For example, the label *justice* in the ACE2005 is mapped to 13 fine-grained types in RichERE (e.g., *appeal*, *execute* ..., etc.). More critically, as demonstrated by Cour et al. [9], how much can be learned from ambiguous training data is fundamentally affected by the *ambiguity degree* of the labels, which intuitively measures the maximum probability of an erroneous label co-occurring with the correct label in the candidate set. Because each source training example with label *justice* will be mapped to the same 13 fine-grained target types, the *ambiguity degree* will be high, limiting the potential benefit from weak-supervision. Indeed, we note that FT and MULT achieve better accuracy than WeakCE and MLCE for this scenario, although the weak supervision approaches are still advantageous in terms of the recall.

To close the gap between WeakCE and traditional transfer learning, we also consider WeakCE+, a variant of WeakCE that trains with the introduced objective (Equation 2) for the first four epochs and then continue to train only on unambiguous examples. From the results, we can see WeakCE+ improves over WeakCE by 1% for accuracy in scenario 3.

ET : ACE2005 → RichERE								
	Scenario 1 (fine to fine)				**Scenario 2 (fine to coarse)**			
		Macro-Avg				**Macro-Avg**		
Method	**Accuracy**	**P**	**R**	**F**	**Accuracy**	**P**	**R**	**F**
TO	47.41±3.04	43.93±4.03	36.06±3.74	39.56±3.59	70.71±2.27	62.59±4.85	56.05±2.74	59.08±3.39
FT	63.71±1.04	65.53±3.24	59.62±1.99	62.40±2.24	76.71±1.89	65.19±6.12	62.14±3.15	63.58±4.43
FreezC	64.87±1.31	66.84±3.39	62.97±2.06	64.83±2.56	78.66±1.46	68.93±4.34	65.76±3.01	67.27±3.40
MULT	63.67±2.20	63.01±3.77	59.22±2.84	63.01±3.77	77.58±2.21	68.54±4.31	64.58±3.72	68.54±4.31
MLCE	65.54±1.19	66.69±2.26	**67.48±1.29**	66.69±2.26	79.80±1.29	71.11±1.23	71.26±1.54	71.11±1.23
PL-rank	66.68±1.28	65.32±3.16	64.78±2.75	65.32±3.16	80.00±1.33	71.35±1.28	**71.90±1.68**	71.35±1.28
WeakCE	**66.76±1.36**	**66.72±3.10**	66.10±2.74	**66.72±3.10**	79.88±1.07	71.26±0.89	71.25±1.39	71.26±0.89
	Scenario 3 (coarse to fine)				**Scenario 4 (mixed to mixed)**			
TO	47.41±3.04	43.93±4.03	36.06±3.74	39.56±3.59	52.80±4.94	48.13±4.32	45.09±4.40	46.49±4.05
FT	**57.02±2.05**	**53.22±4.15**	47.48±3.93	50.13±3.63	63.76±1.97	61.69±3.33	59.61±2.40	60.59±2.36
FreezC	53.79±1.90	46.60±3.37	45.37±3.11	45.94±2.95	64.97±1.20	64.44±3.40	62.49±1.77	63.42±2.38
MULT	56.52±2.12	51.28±3.90	47.19±3.41	51.28±3.90	64.91±1.81	61.64±3.82	60.37±2.58	61.64±3.82
MLCE	54.98±2.18	50.64±3.52	51.87±3.37	50.64±3.52	64.91±1.58	64.68±2.78	64.75±2.01	64.68±2.78
PL-rank	49.94±2.42	46.09±3.82	41.09±2.58	46.09±3.82	66.55±1.20	65.31±3.34	64.41±2.26	65.31±3.34
WeakCE	55.55±2.83	52.95±4.67	48.16±3.82	**52.95±4.67**	**68.25±1.36**	**69.58±2.94**	**68.24±1.51**	**69.58±2.94**
WeakCE+	56.51±1.87	51.46±3.71	**48.46±3.12**	51.46±3.71	-	-	-	-

Table 2.1: Accuracy and macro-averaged recall, precision and F_1 of the methods TO, FT, and FreezC, MULT, MLCE, WeakCE, WeakCE+ on the task of ET. The size of the target training set is 300.

Varying the Size of Target Data of ET. Table 2.2 presents the *accuracy* of all tested method by varying the size of the target (RichERE) set. We can observe that freezing the word embeddings and the convolutional layers is better than fine-tuning them for small target training set size. If the network weights learned from the source are fine-tuned using a small target data, the network may overfit to this data and fail to generalize well to unseen data. This explains why the performance of FT become comparable to FreezC as the size of the target dataset increases. It can also observe from Table 2.2 that the improvement margin that can be obtained by transfer learning decreases as the size of the target data increases. In addition, we can see that the traditional deep transfer learning methods are more sensitive to the size of the target data than the mapping-based methods, as the later can maintain reasonable improvement over TO regardless that target set size in most of scenarios.

Results on TC. Results on TC are presented in Table 2.3. It can be observed that FT and its variant are surprisingly harmful to the target task even though the two domains are very related while MULT does not add any improvement. MLCE and WeakCE improve the accuracy of the target task by 2-3%. Interestingly, this is the only case where we see MLCE outperform WeakCE. To understand this, we studied the bipartite graph used to map the source label to target labels.

ET : ACE2005 → RichERE								
	Scenario 1 (fine to fine)				**Scenario 2 (fine to coarse)**			
	Training set size				**Training set size**			
Method	**300**	**400**	**500**	**600**	**300**	**400**	**500**	**600**
TO	47.41±3.04	53.86±2.63	59.78±2.50	61.29±1.98	70.71±2.27	75.84±1.43	75.16±2.97	76.14±1.48
FT	63.71±1.04	64.81±2.51	66.67±1.46	67.94±1.31	76.71±1.89	79.32±0.81	80.27±1.08	79.72±0.98
FreezC	64.87±1.31	65.22±1.95	66.89±1.33	67.73±1.49	78.66±1.46	80.36±1.46	80.57±1.08	81.30±1.17
MULT	63.67±2.20	64.89±2.36	66.76±1.60	67.86±2.14	77.58±2.21	80.38±1.09	80.22±1.17	79.95±0.76
MLCE	65.54±1.19	66.37±1.48	67.33±1.41	67.98±1.62	79.80±1.29	81.07±0.84	81.63±1.00	**82.18±1.20**
PL-rank	66.00±1.11	67.01±1.61	67.76±1.12	67.92±1.40	80.00±1.33	**81.23±0.94**	81.62±1.04	81.81±1.06
WeakCE	66.76±1.36	68.00±1.07	68.31±1.46	**69.58±1.08**	80.00±1.24	81.19±0.93	**81.89±0.86**	82.00±1.29
	Scenario 3 (coarse to fine)				**Scenario 4 (mixed to mixed)**			
TO	47.41±3.04	53.86±2.63	59.78±2.50	61.29±1.98	52.80±4.94	58.04±3.32	59.81±3.73	62.04±3.14
FT	**57.02±2.05**	59.94±2.27	**63.36±2.26**	64.06±1.73	63.76±1.97	66.13±1.44	67.10±0.95	68.35±1.55
FreezC	53.79±1.90	56.89±1.70	59.32±1.77	60.30±1.58	64.97±1.20	66.30±0.97	66.75±1.46	68.43±1.80
MULT	56.52±2.12	59.00±2.52	60.58±2.90	**64.17±1.32**	64.91±1.81	66.62±1.68	67.98±1.06	69.24±1.55
MLCE	54.98±2.18	59.30±2.34	61.86±2.20	62.69±1.64	64.91±1.58	66.87±1.05	67.37±1.28	68.42±1.91
PL-rank	49.94±2.42	54.22±2.38	57.89±2.66	59.64±1.77	66.55±1.20	67.46±1.60	69.29±1.32	69.60±1.77
WeakCE	55.55±2.83	58.77±2.28	62.02±2.76	62.91±1.35	**68.25±1.36**	**69.27±1.78**	**69.61±1.19**	**70.34±1.59**
WeakCE+	56.51±1.87	**60.15±2.43**	63.32±2.62	64.10±1.47	-	-	-	-

Table 2.2: Accuracy of TO, FT, and FreezC, MULT, MLCE, WeakCE, WeakCE+ on the task of Entity Typing for varying training set sizes.

TC : News → Google Spinets				
Method	Accuracy	**Macro-Avg**		
		P	**R**	F_1
TO	82.20±2.34	82.34±2.05	82.93±2.34	82.34±2.05
FT	79.48±2.72	80.11±2.22	79.98±2.80	80.11±2.22
FreezC	79.58±2.35	79.99±2.04	80.10±2.35	79.99±2.04
MULT	82.26±1.82	82.26±1.53	82.75±1.96	82.71±1.73
MLCE	**85.50±1.99**	**85.35±1.81**	**85.40±2.37**	**85.35±1.81**
PL-rank	82.17±2.59	83.94±1.80	81.26±3.56	83.94±1.80
WeakCE	84.69±2.10	85.13±1.79	84.20±2.81	85.13±1.79
WeakCE+	84.88±1.81	85.35±1.48	84.53±2.28	85.35±1.48

Table 2.3: Accuarcy and macro-averaged recall, precision and F_1 of the methods TO, FT, and FreezC, MULT, MLCE, WeakCE, WeakCE+ on the task of TC.

We observed that the source label *Sci-tech* is mapped into three target labels: *education-science*, *computers* and *engineering*. Taking a closer look at the specific examples, we found that many source examples in *Sci-tech* truly belong to two or more of the corresponding target classes. In other words, the underlying assumption made by MLCE is more accurate for this particular transfer scenario, hence leading to improved performance compared to WeakCE. PL-rank is less effective in modeling this relationship because it pushes the score of exactly one label up and suppresses the score of rest.

To conclude our discussion of the experimental results, we note that Weak supervision, especially WeakCE and WeakCE+ appear to offer the most consistent and substantial gain from transfer learning across a variety of tasks and transfer scenarios. However, it should be noted that when the bipartite graph introduces highly ambiguous training from the source, Fine Tuning remains a highly competitive method.

2.9 Conclusion

In this paper, we address the problem of transfer learning in neural network-based NLP tasks with variant but related label sets. We proposed a method to employ the knowledge about the relationship between the source and the target labels represented by a label bipartite graph to improve the transfer learning. The bipartite graph is used to map the training examples of the source task to the target task label space where the mapped example could be assigned an ambiguous target label set. Then, the training is performed on a union of the target data and the mapped source data using a weak supervision objective function. We compare our method to the standard neural-based transfer learning methods under several label relationships scenarios. We found that even for related tasks, the structure of the label relationships plays an important role in the efficacy of the tested methods. We also found that our method outperforms that standard neural transfer learning except for the scenario where the ambiguity degree of label set assigned to a source examples is high. As future work, we would like to explore ways to learn an initial mapping automatically and adjust this mapping during the learning.

Description-Based Zero-shot Fine-Grained Entity Typing.

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi and Prasad Tadepalli

Proceedings of NAACL-HLT 2019, pages 807–814

Chapter 3: Description-Based Zero-shot Fine-Grained Entity Typing.

3.1 abstract

Fine-grained Entity typing (FGET) is the task of assigning a fine-grained type from a hierarchy to entity mentions in the text. As the taxonomy of types evolves continuously, it is desirable for an entity typing system to be able to recognize novel types without additional training. This work proposes a *zero-shot* entity typing approach that utilizes the type description available from Wikipedia to build a distributed semantic representation of the types. During training, our system learns to align the entity mentions and their corresponding type representations on the known types. At test time, any new type can be incorporated into the system given its Wikipedia descriptions. We evaluate our approach on FIGER, a public benchmark entity tying dataset. Because the existing testing set of FIGER covers only a small portion of the fine-grained types, we created a new testing set by manually annotating a portion of the noisy train data. Our experiments demonstrate the effectiveness of the proposed method in recognizing novel types that are not present in the training data.

3.2 Introduction

Entity Typing assigns a semantic type (e.g., *person*, *location*, *organization*) to an entity mention in text based on the local context. It is useful for enhancing a variety of Natural Language Processing(NLP) tasks such as question answering [22, 13], relation extraction [40, 85], and entity linking [68, 62]. Traditional Named Entity Typing systems consider a small set of coarse types (e.g., *person*, *location*, *organization*) [70, 31, 6]. Recent studies address larger sets of fine-grained types organized in type hierarchies (e.g., *person/artist*, *person/author*) [37, 8, 83, 49]. Fine-Grained Entity Typing (FGET) is usually approached as a multi-label classification task where an entity mention can be assigned multiple types that usually constitute a path in the hierarchy [58].

In real-world scenarios, there is a need to deal with ever-growing type taxonomies. New types emerge, and existing types are refined into finer sub-categories. Traditional methods for

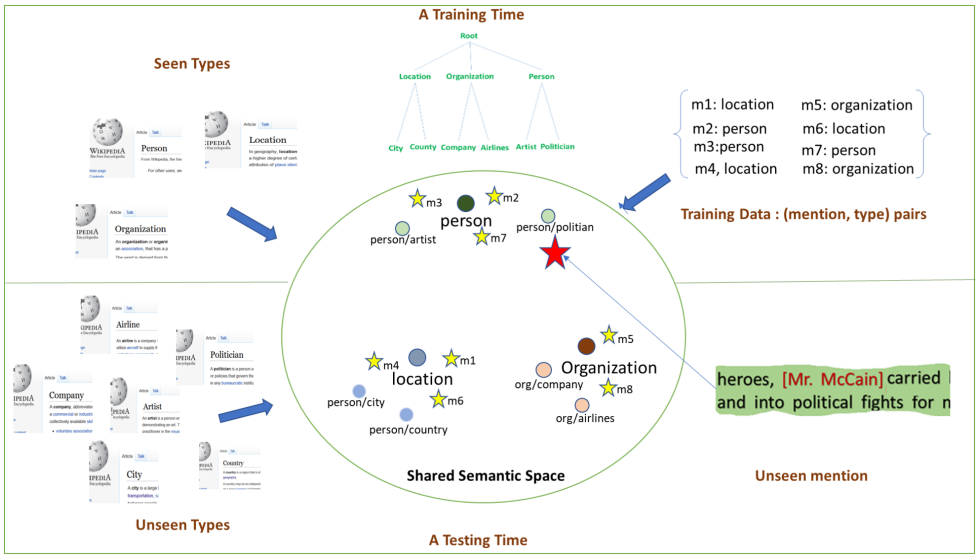

Figure 3.1: Framework Overview of DZET

entity typing assume that the training data contains all possible types, thus require new annotation effort for each new type that emerges. *Zero-shot learning (ZSL)*, a special kind of transfer learning, allows for new types to be incorporated at the prediction stage without the need for additional annotation and retraining. The main idea behind ZSL is to learn a shared semantic space for representing both the seen and unseen types, which allows the knowledge about how examples link to the seen types to be transferred to unseen types.

For fine-grained entity types, we observe that their associated Wikipedia pages often provide a rich description of the types. To capture this, we propose a *Description-based* Zero-shot Entity Typing (DZET) approach that utilizes the Wikipedia description of each type (e.g., see `https://en.wikipedia.org/wiki/Artist` for description of the type *person/artist*) to generate a representation of that type. We learn to project the entity-mention representations and the type representations into a shared semantic space, such that the mention is closer to the correct type(s) than the incorrect types. The mid-level type representation derived from the Wikipedia page along with the learned projection function allows the system to recognize new types requiring zero training examples. The framework of DZET is presented in Figer 3.1.

We investigate different approaches for constructing the type representation based on Wikipedia descriptions. Note that the descriptions can be quite long, often containing many different parts

that are useful for recognizing different entity mentions. This motivates us to generate a bag of representations for each type and apply average pooling to aggregate the results.

Framework Overview of AFET

We evaluate the performance of our methods on FIGER, a benchmark dataset for the FNET task, in which types are organized in 2-levels hierarchy. In this work, We focus on testing our method's capability in recognizing unseen fine-grained types (Level-2 types in this dataset). As the current testing set of FIGER contains examples from only a few level-2 types, we created a new test data that covers most of the level-2 types by manually annotating a portion of the noisy training data.

Below we summarize our main contributions.

- We proposed a description-based zero-shot fine-grained entity typing framework that uses Wikipedia descriptions to represent and detect novel types unseen in training.
- We created a new testing set for fine-grained entity typing that provides much better coverage of the level-2 (fine-grained) types compared to the original FIGER test data.
- We provided experimental evidence of the effectiveness of our approach in comparison with established baselines.

3.3 Related Work

Existing work on FGET focuses on performing context-sensitive typing [19, 8], learning from noisy training data [1, 58, 83], and exploiting the type hierarchies to improve the learning and inference [87, 49]. More recent studies support even finer granularity [7, 49]. However, all the methods above have the limitation that they assume all types are present during training.

Zero-Shot Learning has been extensively studied in Computer Vision (CV) [76] for tasks such as image classification [32, 94, 64], object localization [35, 36] and image retrieval [84, 92]. A common approach for zero-shot learning in CV is to represent each class (e.g., Zebra) by a set of semantic attributes such as its shape and color. The semantic attributes serve as the intermediate level that connects the visual features with the classes. The model is trained to learn an alignment between the semantic attributes and the visual features where a new class can be recognized using its semantic attributes without the need for any training examples. In contrast, this type of approach tends not to work well for NLP applications as the semantic

concepts/classes in NLP are often more complex and cannot be easily described by a set of predefined attributes. This explains why the few studies of ZSL for NLP use very different methods to create the transferable intermediate representations.

Zero-Shot Learning has been studied for a number of NLP tasks including event extraction [24, 34, 67], relation extraction[40], Conversational Language Understanding [34]. Specifically, Zero shot entity typing has also been explored, where most of the prior methods adopt the idea of learning a shared semantic space for representing the entities as well as the types, but differ in how they construct the type embeddings. In OTyper [89], each type is represented by averaging the embedding of the words constitutes the type label. On the other hand, ProtoLE [41] represents each type by a prototype that consists of manually selected entity mentions, where the type embedding is obtained by averaging the prototype mentions' word embeddings. While label embeddings adopted by OTyper not sufficient to compensate for the absence of training data, ProtoLE still requires a very careful manual mention collecting. In contrast, our work differs from OTyper and ProtoLE by constructing the type representations based on the Wikipedia descriptions of the types, which not only carry more information about the type but also can be easily adapted to other tasks such as event typing and text classification.

3.4 Proposed Approaches

3.4.1 The Typing Function

We will begin by introducing our typing function that is used to compute a score between a given mention and type pair, given their corresponding vector representations. We will discuss how to construct the representations in later sections. Formally, the input to this typing function consists of the representation of the mention, denoted by $x \in \mathbf{R}^d$; and the representation of a candidate type t, denoted by $y_t \in \mathbf{R}^{\acute{d}}$. It computes a bi-linear score for the (x, y_t) pair as follows:

$$f(x, y_t, W) = x^T W y_t$$

where $W \in \mathbf{R}^{d \times \acute{d}}$ is a compatibility matrix. Following [87, 41], we factorize W as a product of two low-rank matrices to reduce the number of parameters. That is $W = A^T B$, where $A \in \mathbf{R}^{h \times d}$ and $B \in \mathbf{R}^{h \times \acute{d}}$ (We use $h = 20$). The scoring function f can be rewritten as:

$$f(x, y_t, A, B) = \theta(x, A) \cdot \phi(y_t, B) = (Ax)^T B y_t$$

where $\theta(x, A) : x \rightarrow Ax$ and $\phi(y_t, B) : y_t \rightarrow By_t$ serve as the projection functions that map x and y_t into a shared semantic space.

3.4.2 Entity Mention Representation

To obtain the representation for entity mentions, we adopt the same neural approach proposed by Shimaoka et al. [63]. Given an entity mention with its context, we compute a vector v_m to present the mention m itself, and another vector v_c to represent its left and right contexts c^l and c^r. v_m is computed by simply averaging the embedding of the individual words in m.

To compute the context embedding v_c, we first encode c^l and c^r using a bidirectional-LSTM. Let $c^l_1, ..., c^l_s$ and $c^r_1, ..., c^r_s$ be the word embedding of the left and the right context respectively, where s is the window size (we use $s = 10$), the output layer of the bi-LSTM is denoted as: $\overrightarrow{h^l_1}, \overleftarrow{h^l_1}..., \overrightarrow{h^l_s}, \overleftarrow{h^l_s}$ and $\overrightarrow{h^r_1}, \overleftarrow{h^r_1}..., \overrightarrow{h^r_s}, \overleftarrow{h^r_s}$. We then compute a scalar attention for each context word using a 2-level feedforward neural network:

$$e^j_i = tanh(W_e \begin{bmatrix} \overrightarrow{h^j_i} \\ \overleftarrow{h^j_i} \end{bmatrix}); \tilde{a}^j_i = exp(W_a e^j_i)$$

Where $W_e \in \mathbf{R}^{d_h \times 2 \times d_a}$, $W_a \in \mathbf{R}^{1 \times d_a}$, d_h is the dimension of LSTM, d_a is the attention dimension, $j \in \{l, r\}$. Next, we normalize a^j_is such that they sum up to 1. i.e., $a^j_i = \frac{\tilde{a}^j_i}{\sum_{i=1}^{s}(\tilde{a}^l_i + \tilde{a}^r_i)}$. Finally the context representation is computed as:

$$v_c = \sum_{i=1}^{s}(a^l_i \begin{bmatrix} \overrightarrow{h^l_i} \\ \overleftarrow{h^l_i} \end{bmatrix} + a^r_i \begin{bmatrix} \overrightarrow{h^r_i} \\ \overleftarrow{h^r_i} \end{bmatrix})$$

. The final representation of the entity mention $x \in \mathbf{R}^d$ is a concatenation of v_m and v_c.

3.4.3 Type Representation

Let P_t be the Wikipedia page that is used to build a representation for type t. Some types do not have a Wikipedia page with a title the same as the type label. In such cases, we manually look for a Wikipedia page of a similar concept. For example, we represent the type *living-thing* by the Wikipedia page *organism*.

To get a type representation, We started by the simplest possible method which is averaging

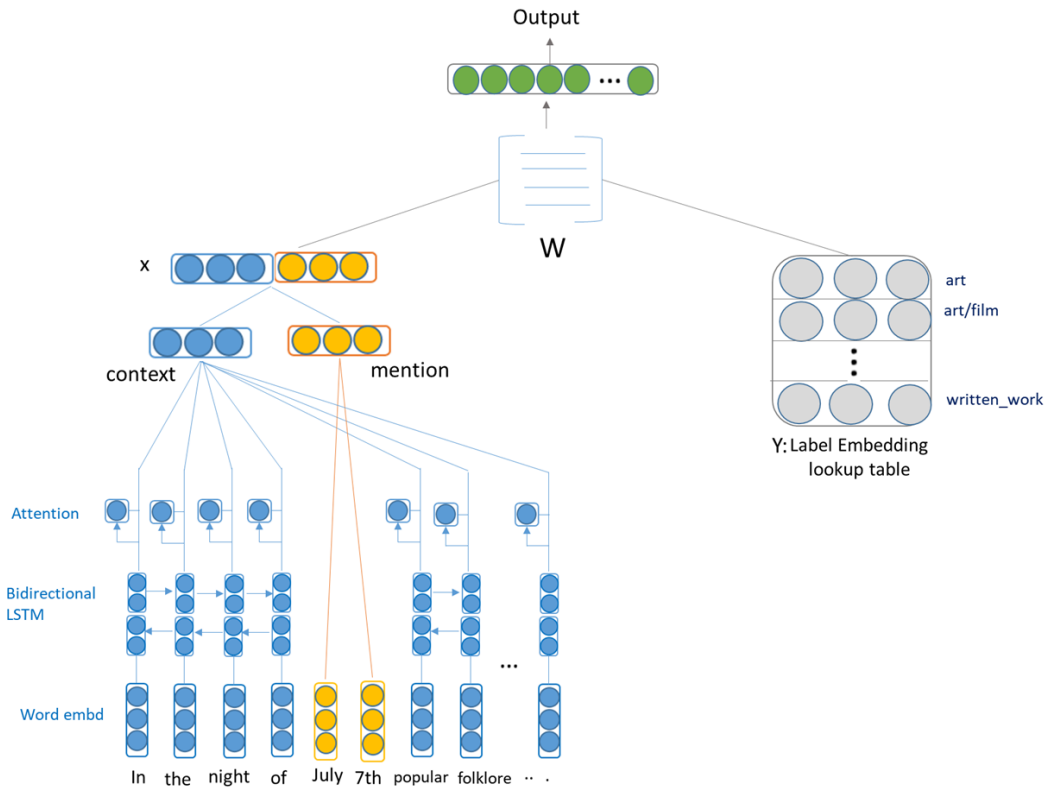

Figure 3.2: Overview of the general neural architecture used for DZET + Multi-rep, DZET + Avg encoder, DZET + Weighted Avg encoder, and the basellines(Label-embd, ProtoLE).All of these methods use the same architecture but differ in how they construct the label embeddings.

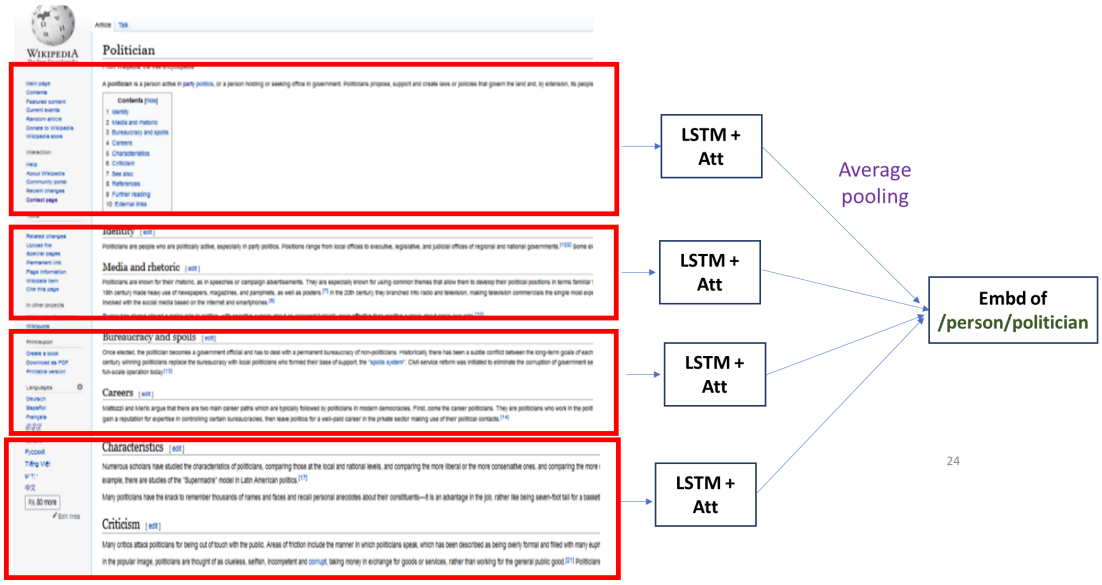

Figure 3.3: Overview of DZET +Multi-rep. Bi-directional LSTM and attention are used to get an embedding y_{ti} of every representation r_{ti} in the bag of representations of the type t. Then, the multiple representations are averaged to obtain a single representation of the type t.

the embedding of words in the Wikipedia page (we call this **Avg encoder**). Since some words in the Wikipedia page carry more of the type semantic than the other words we also consider a (tf-idf)-weighted version of the Avg encoder. Figer 2.2 illustartes the Neural architecture used in this work.

Learning multiple representations. Wikipedia descriptions are often long and contain multiple parts, where different parts may capture different aspects of the type and relate to different mentions. Moreover, sequence models such as LSTM cannot be applied to such long sequences. This motivates us to consider the approach of constructing a bag of multiple representations for each type based on its Wikipedia description. To obtain a bag of representations for type t, we first use a fixed-length window to incrementally break P_t into multiple parts, one paragraph at a time. If a paragraph fits in the current Window, it is added. Otherwise, a new window is initiated. Each window of text r_{ti} is then used to generate one representation. To construct an embedding for r_{ti}, we adopt the same Bi-directional LSTM and attention mechanism that is used to embed the mentioned context. Figure 3.2 illustrates learning multiple representations.

To compute the score for type t given its multiple representations, we compute the score with each individual representation and average them to produce the final score. This is equivalent to applying average pooling to the multiple representations to obtain a single representation due to the bi-linear typing function. During the training, it is important to apply backpropagation after aggregating the results, this helps greatly in alleviating the effect of differences in the size of the representation bags of different types.

3.4.4 Training and Inference

Given the training data, we jointly train the representation and the scoring function by minimizing a ranking score. Let $\mathcal{Y}^{(i)}$ and $\overline{\mathcal{Y}}^{(i)}$ denote the set of correct and incorrect types assigned to the example $x^{(i)}$ respectively, we learn to score types in $\mathcal{Y}^{(i)}$ higher than types in $\overline{\mathcal{Y}}^{(i)}$ with a multi-label max-margin ranking objective as follows:

$$\sum_{y \in \mathcal{Y}} \sum_{\acute{y} \in \overline{\mathcal{Y}}} max(0, 1 - f(x, y, A, B) + f(x, \acute{y}, A, B))$$

At testing, both seen and unseen types are mapped to their learned representations, which are then scored for a given input. Given the scores, we conduct a top-down search following the type hierarchy Ψ. Starting from the root we recursively find the type with the highest score among the children. Since we focus on the fine-grained types, we stop the search when a leaf type is reached and predict that the mention is positive for all types along the path leading the to leaf type.

3.5 Experimental Settings

Datasets. Our experiments use FIGER and BBN, two publicly available fine-grained entity typing benchmark datasets in which types are organized into a 2-level hierarchy. The training data of FIGEr consists of sentences sampled from Wikipedia articles, while BBN training comes from Wall Street journal articles. Both training sets are and automatically annotated via distant supervision [37], and the testing sets are of manually annotated.

Setting. To evaluate our capability to recognize fine-grained types in zero-shot setting, we assume all second-level types are unseen during training, i.e., we remove all level-2 types from

	Original dataset			New dataset		
	train	dev	test	train	dev	test
# of mentions	2000k	10k	563	1999k	10k	917
# of types	111	111	47	46	46	66
# of level-2 types	65	65	26	0	0	40

Table 3.1: Statistics of FIGER dataset.

the train and dev data but keep them in the test data. We observe that the FIGER testing set covers only a small number of second-level types. This renders it insufficient for testing under the evaluation setting we adopt. Moreover, the training data is noisy since it is automatically annotated by distant supervision. As a result, we cannot just use part of it for testing.

To overcome this limitation, we manually annotated new testing set from the noisy training data of FIGER. We first divide the training set into clean and noisy as suggested in [58]. Clean examples are those whose types fall on a single path (not necessarily ending with a leaf) in Ψ. For instance, the mention with labels *person*, *person/author*, and *person/doctor* is considered as noisy example because the labels form two paths.

We then manually verify the correctness of up to 20 examples from the clean training data for every level-2 type. These examples are removed from training and added to the testing set. We ignore the types with no clean examples. The statistics of the new and original datasets are reported in Table 4.1.

Baselines. We consider two baselines that employ the same neural architecture but use different type representations. The **Label-embd** baseline use the average of the embedding of the words in the type label as the type representation. **ProtoLE** baseline uses the prototypes-based label embedding learned by Ma et al.[41], where each type is represented by the set of the most representative entity mentions. The type embedding is the average of the word embeddings of all mentions in the corresponding prototype.

Evaluation metrics. Following prior works in FGET, we report *Accuracy (Strict-*$F1$*)*, *loose Macro-averaged* $F1$ ($F1_{ma}$) and *loose Micro-averaged* $F1$ ($F1_{mi}$) [37].

Training and hyperparameters. For every model we trained, we tune all of the hyper-parameters using a development set. We use the version of FIGER provided by [63] which already with-

Approach	Overall			Level-1		Level-2	
	Acc	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mir}$	$F1_{ma}$	$F1_{mir}$
Label-embd	0.2846	0.5510	0.5603	**0.8165**	**0.8163**	0.2854	0.2954
ProtoLE	0.2541	0.4982	0.5093	0.7424	0.7422	0.2541	0.2657
DZET + Avg encoder	0.3141	0.5522	0.5614	0.7903	0.7902	0.3141	0.3247
DZET + Weighted Avg encoder	0.3261	0.5500	0.5607	0.7740	0.7738	0.3261	0.3390
DZET + Multi-rep	**0.3806**	**0.5953**	**0.6045**	0.8100	0.8098	**0.3806**	**0.3926**

Table 3.2: Level-1 , Level-2 and overall performance of Label embd, ProtoLE, and DZET variants on FIGER dataset.

Approach	Overall			Level-1		Level-2	
	Acc	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mir}$	$F1_{ma}$	$F1_{mir}$
Label-embd	0.2458	0.5031	0.5180	0.7603	0.7603	0.2458	0.2608
ProtoLE	0.2186	0.5031	0.5201	**0.7877**	**0.7877**	0.2186	0.2338
DZET + Avg encoder	0.3435	0.5409	0.5583	0.7382	0.7382	0.3435	0.3664
DZET + Weighted Avg encoder	0.4030	0.5697	0.5899	0.7364	0.7364	0.4030	0.4325
DZET + Multi-rep	**0.4460**	**0.5993**	**0.6161**	0.7525	0.7525	**0.4460**	**0.4718**

Table 3.3: Level-1 , Level-2 and overall performance of Label embd, ProtoLE, and DZET variants on BBN dataset.

hold a portion of the training set as a development set. We randomly development set from BBN training set. For each experiment, we report that testing results of the model that has the best *accuracy* on the development set. We adopt *glove* 300-dimensional word embedding [55] throughout this work except for *prototype* baselines; we use word2vec [42] as it is used to compute the prototypes embedding in the original works [41]. The hyper-parameters used in the feature representation component are the same as in [63]. we set both of the hidden-size of the LSTM was set and the hidden-layer size of the attention module to 100. We use Adam optimizer [29] with the learning rate .001. The model is trained for five epochs. We use Window of size 200 to build a bag of representations for each type from its Wikipedia description.

3.6 Results and Discussions.

Tables 4.6 and 4.5 present the results on FIGER and BBN, evaluated on all types (Overall), the seen types (Level-1) and the unseen types (Level-2) respectively. From the results, we can see that our description-based methods have a particularly strong advantage over baselines on level-2 types. This is consistent with our expectation because Wikipedia descriptions tend to be

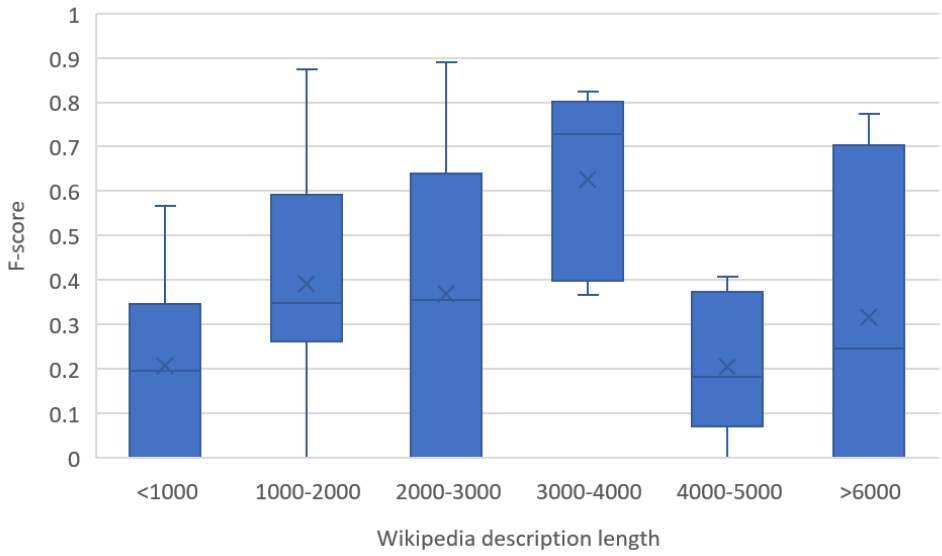

Figure 3.4: The relationship between the length of the Wikipedia description (word count) of level-2 types and the F-score obtained by *DZET+Multi-rep* method on FIGER dataset.

highly informative for fine-grained types, but less so for coarser types.

Among the average encoders, we found that weighting the word embedding by the word tf-idf produces better results than treating the words equivalently. As expected, using LSTM based multi-representation adds a noticeable benefit to our system as it produces the best performance among all tested methods, achieving the best performance for level-2 types on both datasets and outperforming others by a large margin while maintaining highly competitive performance for level-1 types on FIGER. For BBN, ProtoLE and Label-embd achieve better level-1 performance than Multi-rep, This could be due to the fact that the descriptions of level1-types tend to be more abstract and less informative than the description of level-2 types, which suggest that combining Multi-rep with ProtoLE or Label-embd could lead to further improvement.

The effect of description quality. Figure 3.4 analyzes the relationship between the length of the Wikipedia description as one criterion of the description quality and the performance of *Multi-rep* method on FIGER dataset. In particular, we group the types based on the length of their Wikipedia descriptions and provide the five-number summary box plot of the F-scores

for each group. It can be readily observed that the performance is low when the description of the type's Wikipedia page is too short ($<$ 1000 words) or too long ($>$ 4000 words). Short descriptions are less informative and carry less shared semantics with the type's mentions. On the other hand, overly long descriptions could also be confusing as it might share a significant number of common words with the descriptions of other types. A closer look into the results unveils some exceptions. For example, the F-score on the type '/education/educational-degree' is 0.7742 even it has a long description (6845 words). The description of this type is indeed very informative and includes a comprehensive list of the educational degrees awarded all around the world.

The length of the description is not the only factor that affects the performance of DZET methods. One factor is the performance on the Level-1 types. Since the inference is performed by following the type hierarchy, if an incorrect type is inferred at level-1, there is no hope to get the correct level-2 type. Another factor is the amount of overlapping between the descriptions of the related types. For instance, *Multi-rep* produces zero F-score on the types'/event/protest' and '/location/province' because they share a lot of common words with the types '/event/attack' and '/location/county' respectively, which negatively affects the ability of *Multi-rep* to distinguish between the related types. Both '/event/protest' and '/location/province' have a description length between 2000 and 3000 words.

To mitigate the effect of the contents overlapping between the highly related type, We plan to apply mention-sensitive attention mechanisms for future work to aggregate the scores in Multi-rep instead of max-pooling.

3.7 Conclusions

In this paper, we propose a novel zero-shot entity typing approach that uses Wikipedia descriptions to construct type embeddings. Our architecture relies on the type embeddings to make predictions for unseen types. We explore several ways to build labels embedding of the type from its corresponding Wikipedia description starting from using simple averaging weighted averaging encoders to representing each type by a bag of representations. We demonstrate experimentally that the bag of representation approach outperforms the simple averaging encoders and the baselines by a substantial margin, especially for the fine-grained types. We Also analyze the effect of the length of the type's Wikipedia description on the ability of the system to recog-

nize that type. We found that the description with moderate length is the most appropriate. A too short description is not sufficiently informative while too long description overlaps greatly with the descriptions of the semantically related types, making it harder to distinguish them during the training. As future work, we would try to focus on extracting the most useful information from the description that is specific to the type and not shared with other types. We also would like to explore combining several methods for the best performance on the types from different levels.

Chapter 4: Fine-grained Entity Typing: An Evaluation Focusing on Fine-grained Types.

4.1 Abstract

Fine-Grained Entity Typing (FGET) is the task of tagging entity mentions in the text with fine-grained type, organized in type hierarchies (e.g., person/author). Several methods have been proposed to tackle this problem, most of which are evaluated using three publicly available benchmark datasets: FIGER, OntoNotes, and BBN. However, we identify several fundamental issues in the empirical evaluations of these methods. Critically, most existing evaluations only report the overall performance aggregated on all types in the hierarchy, which is typically dominated by the performance on coarse types and provides very little information regarding how well these methods work for the fine-grained types. This is further compounded by the fact that the testing sets for two of the three benchmarks have very poor coverage of the fine-grained types. For example, about 50% of test mentions for the benchmark dataset *OntoNotes* only have a single coarse type "other". In this work, we present a new empirical study that re-evaluates the most recently proposed FGET methods by 1) introducing new testing sets with significantly improved coverage for fine-grained types; and 2) examining not only the overall performance but also detailed per-level, type-specific performance. Our analysis of the tested methods reveals new insights about the tested methods, invalidates some of the previous claims due to flawed evaluations and suggests new directions for future improvement for FGET performances.

4.2 Introduction

Tagging entity mentions in the text with their entity types based on the local context (e.g., *person*, *location*) is an important task in Natural Language Processing (NLP). It plays an important role in a large variety of NLP tasks such as question answering [22, 13], relation extraction [40, 85], and entity linking [68, 57]. While traditional Named Entity Typing systems (aka Named Entity Recognition) only consider a small set of coarse types (e.g. person, location, and organization) [70, 31, 6], Fine-Grained Entity Typing (FGET) addresses a significantly more challenging task

involving much larger sets of fine-grained types organized in type hierarchies (e.g., *person/artist*, *person/author*, *person/actor*) [37, 8, 83, 49].

FIGER, OntoNotes, and BBN are three publicly available benchmark corpora used extensively in current FGET research. The training sets in these corpora are annotated automatically using distant supervision from the Knowledge Base (KB), while the testing sets are manually annotated to ensure accurate evaluation. Distant supervision for FGET is performed as follows. First, it identifies the borders of the entity mentions in the text. The entity mentions are then linked to their corresponding entities in the knowledge base. All the types that are associated with the linked knowledge base entities are then assigned to their corresponding mentions. One potential issue with distant supervision is that it introduces out-of-context typing noise since mention is assigned *all* of the types associated with its linked KB entity without taking its local context into account. We illustrate the typing noise in Figure 4.1. The mentions of ***Allen*** in all sentences (S1-S4) are linked to the entity ***Woody Allen***, which has several types including {*person*, *person/author*,*person/actor*, *person/director*}. Distant supervision thus assigns all these types to all the mentions, which creates challenges in learning because some types are not supported by the contextual evidence. For example, it is clear the local context of S1 only supports the type *person*, S2 supports the type *person/actor*, S3 supports the type *person/director*, and S4 supports the type *person/author*.

Numerous FGET approaches have been proposed in the literature to perform context-sensitive typing [19, 8, 63], learn from noisy distant supervision training data [1, 58, 83], and exploit the hierarchical structure of the types to improve learning and inference [87, 49]. However, the empirical evaluation protocols followed by these methods have a critical issue: most existing work only reports the overall performance averaged across all of the types. As a result, the reported performance could be dominated by the performance on the coarse types and fails to provide satisfying answers regarding the ability of these methods to recognize the fine-grained types and the robustness of these methods against the typing noise. Also, the literature frequently compares models trained using different training data. A recent work [63] demonstrated that the training data has a drastic impact on the performance, and the choice of the training data can lead to substantial differences in the reported results.

Moreover, we noticed that the testing sets of FIGER and OntoNotes don't cover most of the fine-grained types. For example, about 61% of the test mentions of *OntoNotes* have only coarse types (e.g., person, location, other, etc.). Of these coarse-type-only mentions, 82% of them belong to a single type "other". In other words, simply predicting *"other"* for all test mentions

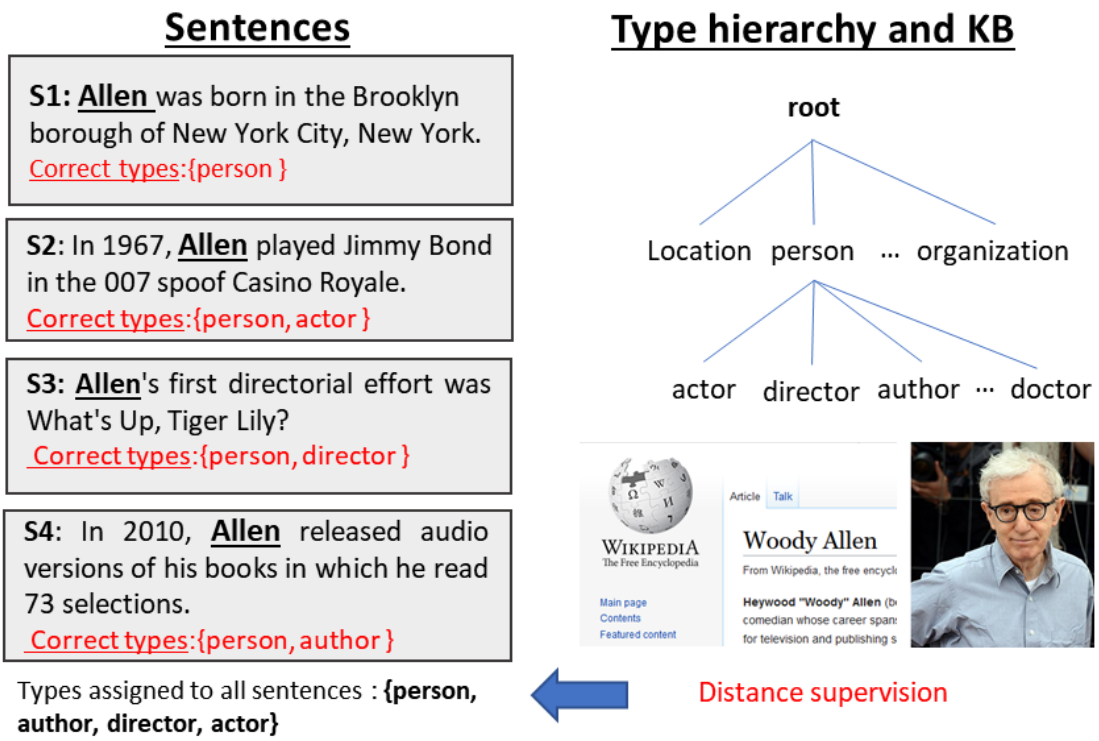

Figure 4.1: Typing noise introduced by distance supervision: the mention ***Alen*** in all sentences are labeled with the types { *person*, *person/author*,*person/actor*, *person/director*}, while for each mention only some of them are correct given the local textual context.

will lead to an accuracy of almost 50%, which is not that far off from the current state-of-the-art on this dataset. Similar issues can be found for FIGER's testing set: it a tiny set that has only 563 examples in comparison with the training set of more than 2 million examples. Also, more than 60% of the fine-grained type are not covered at all. Even for those fine-grained types that are covered, we found that 72% of them have five or few test examples. This highlights the need for more representative testing sets with adequate coverage of the fine-grained types.

In this work, we introduce new testing sets for FIGER and OntoNotes with significantly improved coverage for fine-grained types. We manually annotate part of the noisy training set for each benchmark to augment the corresponding testing set. We present a new empirical study that re-evaluates the most recently proposed FGET methods on both current and new testing sets. We train all models using the same train/dev/testing sets to ensure complete fairness in all compar-

isons. In addition to the overall performance, we report per-level, type-specific performance. Finally, we analyze the tested methods in view of the new results, which reveals new insights and suggests new directions for further improvement.

4.3 Related Work

This section reviews most of the FGET approaches that have been proposed to shed light on the deficiency of the de-facto evaluation protocols and benchmarks in assessing their performance.

FGET was first investigated by Ling and Weld [37]. They used distant supervision to collect FIGER, a large, noisy FGET training set that consists of 112 types organized into a hierarchy. They also manually annotated a significantly smaller testing set to evaluate their approach. Their system uses linear perceptron to perform multi-label classification. Gillick et al. [19] proposed a context-dependent FGET method that assigns types to the entity mentions based on the local context. They used distant supervision to annotate OntoNotes [78] with 89 types and provided manually annotated testing data. Subsequent FGET studies focus on performing context-sensitive typing [19, 8], learning from noisy training data [1, 58], and exploiting the type hierarchies to improve learning and inference for FGET [87, 83]. More recent studies support even finer granularity [7, 49].

Context-sensitive typing. Earlier works [8, 87, 19, 56] used hand-crafted features (e.g., headword, word shape, brown cluster, etc.) to build a representation of the mention given its context. However, these features are extracted using various NLP tools, therefore, the inevitable errors of these tools propagate to the resulting FGET models. Shimaoka et al. [63] proposed an attentive neural network that uses bi-directional LSTM to encode the mention's context in addition to the attention mechanism to focus on the most relevant parts of the context. The mention itself is encoded by an average encoder. Abhishek et al. [1] added a character-level LSTM to capture the entity mention morphological and orthographic properties. Xu et al. [83] used similar input encoding mechanisms as [63] except that they added LSTM over the words in the mention in addition to the average encoder to obtain the mention embedding. Murty et al. [49] employed CNN with position embeddings to obtain a representation of the mention and the context.

Learning from noisy training data. Many previous studies [37, 88, 15, 63] ignored the label noise introduced by distant supervision and assumed that *all* candidate types assigned to the

mentions are *true* types. As revealed by Table 4.2, there is a considerable percentage of noisy mentions in the training data of the benchmark tasks. Noisy labels may be misleading to the training and negatively affect the system's performance. Gillick et al. [19] are the first to address the typing noise issue by proposing pruning heuristics. However, this can significantly reduce the size of the training data primarily for types whose mentions are predominantly noisy (see Table 4.2). As we will show later in the experiments that even though pruning has been demonstrated to have limited impact on or even improve the overall performance, it may lead to substantial performance degradations on the fine-grained types and these types whose majority of the mentions have noisy typeset. We show also these interesting findings are completely masked by only reporting the overall performance.

Yogatama et al. [87] presented a label embedding method that projects the type labels and user-defined features into a shared semantic space to encourage the information sharing between related labels and alleviate the effect of the typing noise. Ma et al. [41] proposed a zero-shot label embedding framework that incorporates hierarchical and prototypical information of the labels to learn embeddings that are robust to the label noise and facilitate recognizing both seen and unseen types. Zhang et al. [91] presented a path-based attention neural model to learn from the noisy data by leveraging the hierarchical structure of the types. Nevertheless, these methods still treat all of the noisy candidate types as correct types.

Ren at al. [58] proposed an FGET system (called AFET) that separately models the clean and the noisy mentions, leading to a significant performance gain on benchmark datasets. AFET incorporates the type hierarchy and the type correlation information to induce two loss functions to learn from mentions with clean and noisy candidate types, respectively. The learning is performed collectively by minimizing the two objectives. Following the same direction, Abhishek et al. [1] presented a neural network model that uses two different loss functions to model the clean and the noisy mentions separately. Their loss functions are non-parametric zero-centered variants of the Hinge loss. More recently, Xu et al. [83] proposed to treat FGET as a single-label multi-class classification problem where each class is a node in the type hierarchy. They proposed a neural network and used a variant of the cross-entropy objective to handle the mentions with noisy labels.

However, all of these approaches use an uninformative testing set with very poor coverage of fine-grained types. Besides, they settle for reporting the overall performance and ignore how different methods may behave for types from different levels and with varying ratios of noise. As we will demonstrate later in the experiments, the overall performance can hide interesting

details of the performance and hinder a deep understanding of the weaknesses and the strengths of different methods.

4.4 Problem Description

Fine Grain Entity Typing aims to automatically tag an entity mention in a natural language sentence with its entity type(s). The training data $\mathcal{D}_{train}$ for FGET is automatically collected via distant supervision using a knowledge Base with a set of types T that are organized in a hierarchy Ψ. Specifically, $\mathcal{D}_{train}$ consists of a set of sentences, each sentence consists of the entity mention $m^{(i)}$, the context $c^{(i)} = [c^{l(i)}, c^{r(i)}]$, which includes both the left context $c^{l(i)}$ and the right context $c^{r(i)}$ respectively, and the assigned candidate type set $\mathcal{Y}^{(i)} \subseteq T$. In other words $\mathcal{D}_{train} = \{([m^{(i)}, c^{(i)}], \mathcal{Y}^{(i)})\}_{i=1}^{N}$. Frequently we use $\bar{\mathcal{Y}}^{(i)}$ to denote the complement of $\mathcal{Y}^{(i)}$, which contains the set of types that are not true for a mention $m^{(i)}$, i.e., $T = \mathcal{Y}^{(i)} \cup \bar{\mathcal{Y}}^{(i)}$. Given a test mention m along with its context c, the goal is to learn a classifier that predicts a one-hot label vector $\hat{y} \in \{0, 1\}^L$, where L denotes the number of types Ψ. Here $\hat{y}_t = 1$ if the mention m is of type t, and 0 otherwise. It is typically assumed that the types of a mention should form a single path in Ψ. Hence $\hat{y}$ is commonly restricted such that the positive types form a single type-path (not necessarily ending with a leaf type).

4.5 New Evaluation Setting

4.5.1 Datasets

Our experiments use three standard and publicly available benchmark datasets: FIGER, OntoNotes, and BBN. For FIGER and OntoNotes, we use the versions that are provided by Shimaoka [63] as they are used in the most recently proposed works. For BBN, we use the version provided by Ren et al. [58]. Table 4.1 shows the statistics of these datasets. The details are as follows:

- **FIGER.** The training data of FIGER consists of sentences sampled from Wikipedia pages and automatically annotated with a total of 111 types organized into a 2-level hierarchy derived from Freebase. The testing set consists of 77 news reports manually annotated as described by Ling and Weld [37].

- **OntoNotes.** This benchmark dataset consists of sentences from the news-wire documents

	FIGER			OntoNotes			BBN	
	train	test	new-test	train	test	new-test	train	test
# of mentions	2001k	563	1232	218K	8963	9370	86K	13766
# of types	111	47	73	87	67	79	47	46
# of level-1 types	46	15	25	4	4	4	16	16
# of level-2 types	65	26	48	43	34	40	31	30
# of level-3 types	-	-	-	40	29	35	-	-
% of mentions with level-1 types only	28.6	65.02	27.35	9.3	61.9	59.2	45.5	29.13
% of mentions with levels 1&2 types	71.4	34.98	72.65	56.2	33.6	34.5	54.5	70.87
% of mentions with levels 1&2&3 types	-	-	-	34.5	4.5	6.3	-	-
% of levels 2&3 types occur 5 times or less	0	72.0	1.1	0	36.5	5.3	0	6.8

Table 4.1: Statistics of FIGER, OntoNotes and BBN datasets including training sets, current testing set, and new testing sets.

Dataset	**FIGER**	**OntoNotes**	**BBN**
% Noisy mentions	35.3%	27.0%	24.11%
% Types with $\geq$ 50% noisy mentions	63.9%	42.5%	37.5%
# Types with 100% noisy mentions	5	4	4

Table 4.2: Statistics about noisy lables in FIGER, OntoNotes and BBN training sets.

presented in OntoNotes dataset [78]. The training set is automatically annotated using DBpedia Spotlight to link the entity mentions to Freebase, then mapped to a refined 3-level hierarchy. The testing set is manually annotated as described in [19].

- **BBN.** This dataset is a set of Wall Street journal articles. The DBpedia Spotlight entity linking system is used to annotate the training data, and the testing set contains mentions manually annotated using 47 types constituting a 2-level hierarchy.

The need for additional testing data. By observing Table 4.1, We can see that the testing sets in *FIGER* and *OntoNotes* are inadequate for testing FGET systems for fine-grained entity typing. For example, the testing set of FIGER is very small, containing only 563 mentions, and covers only 36.9% of all types. In addition, only 35.5% of the testing mention has a level-2 type, while 71.4% of the mentions in the training set belong to a second-level type. Even worse, 87.6% of level-2 types in FIGER's testing set have only 5 or fewer examples. Likewise, 51% of the fine-grained types are not presented in OntoNotes' testing set, and the size of 36.5% of the types that are present do not exceed five examples. In order to achieve an effective evaluation of the capability of fine-grained typing, we need to augment the testing data for both benchmarks to enhance their coverage of fine-grained types.

Testing data augmentation. Because the training sets of the benchmarks are noisy as they are annotated by distant supervision, it is not valid to simply take part of the training data to augment the testing set. To address this issue, we manually annotated a new testing set for FIGER and OntoNotes respectively from their corresponding noisy training set as follows. We first divide the mentions in the training set into the clean and noisy mentions as suggested by Ren et al. [58], where clean mentions are those whose types fall on a single path in Ψ (not necessarily ending with a leaf node). For instance, a mention with labels *person*, *person/author*, and *person/doctor* is considered noisy because the types form two pats in Ψ. We then collect a set of new testing data by manually verifying the typing correctness of a number of mentions from the clean mentions for types that have sufficient clean mentions. For the types that have limited or insufficient clean data, we then randomly sample some mentions from the noisy part and manually verify them by deleting types that are inconsistent with the mention's context. Following this procedure, we collected at least 20 testing examples for every level-2 types for FIGER and between 5 to 20 examples for level-l2 and level-3 types for OntoNotes. These

mentions are then removed from the training sets and added to the testing sets.

Additional Preprocessing. We apply some preprocessing steps to the datasets to fix some inherited issues. In addition to removing the redundant examples from Shimaoka's [63] version of OntoNotes, we perform the following:

- **Merging labels.** We found that FIGER has two identical types: *living_thing* and *livingthing*. We observed that mentions under both are about living species, thus we combined them. We found a similar issue in OntoNotes with the type *geography*. It is split into two types *geography* and *geograpy* because of a typo. Likewise, we combine them. This explains the slight differences in the number of types in these datasets with previous work. We also found three sub-types in the BBN testing set that do not exist in the training set, we back off the examples under these types to their parent types.
- **Fixing annotations in FIGER's testing set.** We found that some mentions in FIGER's original testing set only have level-2 types (e.g., mentions under /education/department should be under /education as well). We added the level-1 types to all of these examples.

4.5.2 Evaluated FGET Methods

In this work, we empirically evaluate four FGET methods: Attentive [63], AFET [58], AAA [1], and NFETC [83]. We choose them because they are the most recently proposed FGET methods with the highest reported results. Besides, the code of these systems is publicly available, enabling us to re-evaluate them using our new evaluation setting. Note that fundamentally FGET is a hierarchical multi-label classification problem with labeling noise due to distant supervision. Critically, these methods differ in how they approach the multi-label problem, how they incorporate the hierarchical information, and how they address the labeling noise. Below we introduce each method.

4.5.2.1 Attentive

Shimaoka et al. [63] proposed *Attentive*, which transforms the multi-label classification task for FGET into a set of binary classification tasks (often referred to as the binary relevance method [93]). Specifically, *Attentive* trains a neural network model that uses bi-directional

LSTMs to encode the left and the right context of an entity mention and uses an attention mechanism to focus on the most relevant expressions in the context. Notably, this method ignores the out-of-context typing noise and assumes that all labels obtained via distant supervision are correct. The details of Attentive are as follows:

Input Representation. To obtain a representation of the entity mention given the context, all words in m , c^l and c^r are represented by their word embeddings. The mention vector v_m is simply computed by averaging the individual words in the entity mention. The context embedding is generated by a trained bi-directional LSTM with attention mechanism to weight the LSTM outputs toward computing the final context representation. Formally, let $c_1^l, ..., c_s^l$ and $c_1^r, ..., c_s^r$ be the word embeddings of the left and the right context respectively, where s is the window size ($s = 10$ is used). c^l and c^r are encoded using bi-directional LSTMs. The output layer of the bidirectional LSTM is denoted as: $\overrightarrow{h_1^l}, \overleftarrow{h_1^l}..., \overrightarrow{h_s^l}, \overleftarrow{h_s^r}$ and $\overrightarrow{h_1^r}, \overleftarrow{h_1^r}..., \overrightarrow{h_s^r}, \overleftarrow{h_s^r}$. Then a scalar attention is computed for each context word using a 2-level feed-forward neural network:

$$e_i^j = tanh(W_e \begin{bmatrix} \overrightarrow{h_i^j} \\ \overleftarrow{h_i^j} \end{bmatrix}) \tag{4.1}$$

$$\tilde{a}_i^j = exp(W_a e_i^j) \tag{4.2}$$

Where $W_e \in \mathbf{R}^{d_h \times 2 \times d_a}$, $W_a \in \mathbf{R}^{1 \times d_a}$, d_h is the dimension of LSTM, d_a is the attention dimension, $j \in \{l, r\}$. Next, we normalize a_i^j's across both the left and right contexts such that they sum up to 1, i.e.,

$$a_i^j = \frac{\tilde{a}_i^j}{\sum_{i=1}^{s} \tilde{a}_i^l + \tilde{a}_i^r}$$

. Finally the context representation is computed as

$$v_c = \sum_{i=1}^{s} \left(a_i^l \begin{bmatrix} \overrightarrow{h_i^l} \\ \overleftarrow{h_i^l} \end{bmatrix} + a_i^r \begin{bmatrix} \overrightarrow{h_i^r} \\ \overleftarrow{h_i^r} \end{bmatrix} \right)$$

. The final representation of the entity mention $x \in \mathbf{R}^d$ is a concatenation of v_m and v_c (i.e. $[v_m, v_c]$)) .

Training and Inference Having computed the entity mention representations, the probability of the type y is computed as follows:

$$\hat{p}(y = 1|m, c) = \frac{1}{1 + exp\Big(- W_y \begin{bmatrix} v_m \\ v_c \end{bmatrix} \Big)} \tag{4.3}$$

Where $W_y \in \mathbf{R}^{L \times (d_a + d_h)}$ is the weight matrix. The binary cross-entropy objective is used for training. At inference time, the assumption that at least one type is assigned to each mention is enforced by first assigning the type with the largest probability. Then, additional types are assigned to the mention based on the condition that their corresponding probabilities must be greater than a threshold (0.5) that is determined by tuning on the development set.

Hierarchical Label Encoding. Since Binary Relevance treats the multiple types independently and neglects the hierarchical structure of the types, Attentive also employs a hierarchical encoding scheme that allows for parameter sharing between the types and their ancestors in Ψ. Note that the rows of weight matrix W_y can be viewed as the embedding of the types, whose dot product with the mention and context representation is used to score each type. To capture the hierarchical structure of the types, Attentive assumes that W_y is decomposed as the product of a learned matrix V_y and the structure matrix S_Ψ defined based on Ψ:

$$W_y^T = V_y S_\Psi$$

where $S_\Psi \in \mathbf{R}^{L \times L}$ is a binary matrix that encodes the "is-a" relationships between the types in Ψ. Concretely, S_Ψ is defined as follows:

$$S_\Psi(i, j) = \begin{cases} 1, & \text{if } i = j \text{ or } j \in \Gamma_i. \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

where Γ_i denotes the set of ancestors of the type y_i.

4.5.2.2 AFET

AFET was proposed by Ren et al. [58] and was the first FGET system that separately models clean and noisy mentions, leading to a significant reported performance gain, especially on OntoNotes. A key novelty of AFET is that it incorporates the type hierarchy Ψ and type-to-type correlation information to induce an adaptive-ranking loss function that poses a smaller penalty on negative types that are semantically more relevant to the positive types.

Input Representation. AFET relies on hand-crafted features to capture the syntactic and semantic properties of the mention and its context. Examples of these features include the headword of the mention, Part Of Speech (POS) tags, word shape, and the brown cluster.

Training set partition. Let M to be the set of all mentions in the training set, AFET separates the mentions in M into the sets of clean mentions M_c and noisy mentions M_n. If the types in $\mathcal{Y}^{(i)}$ associated with the mention $m^{(i)}$ constitute a single path in Ψ (not necessarily ending with a leaf node), then it is considered clean and added to M_c, otherwise, it is deemed noisy and added to M_n.

Training and Inference. AFET adapts a ranking-based learning objective for the task of FGET [16]. Their system learns a projection function to map the mention and its types (path) into a shared semantic space. The learning from the clean mentions is performed by Weighted Approximate-Rank Pairwise loss (WARP) [79] enhanced by adding adaptive margins between the types. The goal of the adaptive margins is to rank the positive types ahead of the negative type while differentially imposing varying penalties on the negative types according to how related they are to the positive types. For example, if the positive label is "athlete", the negative type "coach" should receive a smaller penalty (i.e., has a smaller margin) than the negative type of "businessman" since the former is more related to the type "athlete" than the latter. More precisely, AFET defines the loss for clean mention $m^{(i)} \in M_c$ as follows:

$$l_c(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) = \sum_{y_k \in \mathcal{Y}^{(i)}} L(r_{y_k}) \sum_{y_{\bar{k}} \in \bar{\mathcal{Y}}^{(i)}} l^{(i)}_{k,\bar{k}} \tag{4.5}$$

$$l^{(i)}_{k,\bar{k}} = max(0, \gamma_{k,\bar{k}} - f_k(m^{(i)}) + f_{\bar{k}}(m^{(i)})$$

$$r_{y_k} = \sum_{y_{\bar{k}}} \mathbb{1}(\gamma_{k,\bar{k}} + f_{\bar{k}}(m^{(i)} > f_k(m^{(i)}))$$

where $\gamma_{k,\bar{k}}$ is the adaptive margin between y_k and $y_{\bar{k}}$ computed as $\gamma_{k,\bar{k}} = 1 + \frac{1}{w_{k,\bar{k}}+\alpha}$. Here α is a smoothing paramter, and $w_{k,\bar{k}}$ is the normalized number of shared entities between y_k and $y_{\bar{k}}$ in a KB. This implies that the smaller the margin $\gamma_{k,\bar{k}}$ is, the less that type $y_{\bar{k}}$ gets penalized. $L(z) = \sum_{i=1}^{z} \frac{1}{i}$ transforms ranks into weights. The intuition of using $L(r_{y_k})$ is that if the positive type is ranked lower, the violation should be penalized more. The loss $l_n(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)})$ over M_n is computed as follows:

$$l_n(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) = L(r_{y_{k^*}}) \sum_{y_{\bar{k}} \in \bar{\mathcal{Y}}^{(i)}} l^{(i)}_{k^*,\bar{k}} \tag{4.6}$$

$$l^{(i)}_{k^*,\bar{k}} = max(0, \gamma_{k^*,\bar{k}} - f_{k^*}(m^{(i)}) + f_{\bar{k}}(m^{(i)})$$

$$r_{y^*_k} = \sum_{y_{\bar{k}}} \mathbb{1}(\gamma_{k^*,\bar{k}} + f_{\bar{k}}(m^{(i)} > f_{k^*}(m^{(i)}))$$

where $y^*_k = \text{argmax}_{y_k \in \mathcal{Y}^{(i)}} f_k(m^{(i)})$.

Finally, the overall loss is computed as follows:

$$J(\theta) = \sum_{m^{(i)} \in M_c} l_c(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) + \sum_{m^{(i)} \in M_n} l_n(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) \tag{4.7}$$

The **type inference** is conducted via a top-down greedy search following the type hierarchy Ψ. Starting from the root, the type with the highest similarity to the mention among the children is returned. This continues recursively until a leaf type is reached or the similarity falls below a predefined threshold.

4.5.2.3 AAA

AAA [1] is another embedding-based system that separately models the mentions in M_c and M_n. It aims to overcome two limitations of AFET. First, it employs a neural network to learn a representation of the mentions instead of relying on hand-crafted features. Second, it uses a ranking-based objective [18] that is calibrated around zero to eliminate the need for a similarity threshold during the type inference.

Input Representation. Similar to Attentive, AAA uses bi-directional LSTMs to compute a context representation. However, AAA does not use attention mechanism to get a weighted average of the LSTM hidden states. Instead, the representation is simply the concatenation of the final outputs of the forward and backward LSTMs. Another difference is that it takes the whole context into account, not just a fix-sized window around the mention. It also includes the mention itself in the computation of the left and the right context embeddings. Furthermore, AAA uses a character-based LSTM to obtain an embedding of the mention itself that encodes information about the entity mention's morphology and orthography, instead of using an average encoder over the word embeddings.

Training and Inference AAA uses a non-parametric version of hinge loss that maintains a margin centered at zero between the positive and the negative types to learn from M_c as follows:

$$l_c(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) = \sum_{y_k \in \mathcal{Y}^{(i)}} max(0, 1 - f_k(m^{(i)}) + \sum_{y_{\bar{k}} \in \bar{\mathcal{Y}}^{(i)}} max(0, 1 + f_{\bar{k}}(m^{(i)}) \tag{4.8}$$

On the other hand, AAA learns from M_n by considering only the type in $\mathcal{Y}^{(i)}$ with the best score $f_{k^*}(m^{(i)})$. l_n is defined as follows and the total loss is computed exactly as in equation 4.7:

$$l_n(m^{(i)}, \mathcal{Y}^{(i)}, \bar{\mathcal{Y}}^{(i)}) = max(0, 1 - f_{k^*}(m^{(i)}) + \sum_{y_{\bar{k}} \in \bar{\mathcal{Y}}^{(i)}} max(0, 1 + f_{\bar{k}}(m^{(i)}) \tag{4.9}$$

.

Similar to AFET, the **type inference** AAA is also performed by a top-down search in the given Ψ, Starting from the tree root, the best type with the highest score among node's children is recursively selected based. This process continues until a leaf node is encountered, or the best score falls below zero.

4.5.2.4 NFETC

As mentioned before, binary relevance methods transform multi-label classification to a set of binary classification problems. Another approach for multi-label classification is the Label powerset (LP) [73] method that reduces MLC into a single-label multi-class classification problem. In particular, it treats each unique type set that has appeared in the training data as a single unique class. This approach is adopted by Xu et al.[83] to tackle FGET as a multi-class classification

problem based on the assumption that each mention can only have one *type-path* depending on the context, and each type-path can be represented uniquely by its terminal type, which may or may not be a leaf node in Ψ. Their Neural Fine-Grained Entity Type Classification (NFETC) model uses a variant of cross entropy loss function to handle out-of-context labels automatically during the training phase. Additionally, a hierarchical loss normalization is introduced allowing the model to utilize the type hierarchy and to adjust the penalties for correlated types.

Input Representation. NFETC uses a similar input representation as in *Attentive* with two slight differences. First, when computing the context representation, the embedding of each context word e_i^j in equation 4.1 is computed by applying $tanh$ directly on the hidden states of the bi-directional LSTMs without using the weight matrix W_e. Formally, let $c_1^l, ..., c_s^l$ and $c_1^r, ..., c_s^r$ be the word embeddings of the left and the right context respectively, where s is the window size ($s = 10$ is used). c^l and c^r are encoded using bi-directional LSTMs. The output layer of the bidirectional LSTM is denoted as: $\overrightarrow{h_1^l}, \overleftarrow{h_1^l}..., \overrightarrow{h_s^l}, \overleftarrow{h_s^r}$ and $\overrightarrow{h_1^r}, \overleftarrow{h_1^r}..., \overrightarrow{h_s^r}, \overleftarrow{h_s^r}$. Then a scalar attention is computed as follows:

$$e_i^j = tanh(\begin{bmatrix} \overrightarrow{h_i^j} \\ \overleftarrow{h_i^j} \end{bmatrix}) \tag{4.10}$$

Secondly, in addition to the mention average encoder, the mention is extended by adding one word before and one word after, then apply word-level LSTM on the extended mention to capture more semantic information from the mentions. The last output h_{t+1} then serves as the LSTM representation v_e of the mention. It is worth mentioning that we tried to add this component to *Attentive* observed that it did not add any noticeable difference to the performance.

Training and Inference. We first update the definition of $\mathcal{Y}^{(i)}$ to be the set of *type paths* from Ψ that are associated with the entity mention m_i, each represented by its terminal type. Given a mention m and the computed entity mention and context representations, the probability of type y (aka the path ending at y) for mention m is computed as follows:

$$\hat{p}(y|m, c) = Softmax\Big(W_y \begin{bmatrix} v_m \\ v_c \\ v_e \end{bmatrix} + b\Big) \tag{4.11}$$

Similar to AFET and AAA, NFETC separately model the M_c and M_n during the training. For all the mention in M_c , $\mathcal{Y}^{(i)}$ has only one type-path which is $y^{(i)}$, while it has more then one type path for every mention in M_c. To learn from M_c, NFETC employs the traditional cross-entropy function as follows:

$$J(\theta) = \frac{1}{N}\sum_{i}^{N} \log \hat{p}(y^{(i)}|m^{(i)}, c^{(i)}) + \lambda||\theta||^2 \tag{4.12}$$

where θ denotes all parameters of model, λ is the regularization parameter. In contrast, NFETC learns from M_n using a variant of the cross-entropy loss:

$$J(\theta) = \frac{1}{N}\sum_{i}^{N} \log \hat{p}(y^{*^{(i)}}|m^{(i)}, c^{(i)}) + \lambda||\theta||^2$$

$$\text{where} y^{*^{(i)}} = \underset{y \in \mathcal{Y}^{(i)}}{\operatorname{argmax}} \hat{p}(y|m^{(i)}, c^{(i)})$$

This follows the assumption that the type (path) in $\mathcal{Y}^{(i)}$ that gets the highest probability given the local context during the learning is the correct type.

Hierarchical Loss Normalization. Since it is unreasonable to treat all types equally and ignore the hierarchical relationship between the types, NFETC introduced the concept of hierarchical loss normalization to adjust the probability in 4.11 as follows:

$$\hat{p^*}(\hat{y}|m, c) = \hat{p}(\hat{y}|m, c) + \beta \sum_{t \in \Gamma(\hat{y})} \hat{p}(t|m, c) \tag{4.13}$$

where $\Gamma(\hat{y})$ is the set of ancestor types along the type-path of $\hat{y}$. This can be viewed as performing smoothing of the probability where the probability of a type is enhanced by adding a portion of its ancestors' probabilities. This also helps in introducing less penalty on the ancestors of the correct type. β is a hyperparameter to control the amount of smoothing. This update is followed by a re-normalization to form a valid probability distribution.

4.5.3 Evaluation Metrics

Following prior works in the literature, we evaluate all methods using *Accuracy (Strict-*$F1$*)*, *loose Macro-averaged* $F1$ ($F1_{l.ma}$) and *loose Micro-averaged* $F1$ ($F1_{l.mi}$) [37]. Let T_i and $\hat{T}_i$

be the true types and the predicted types of the mention m_i respectively, N is the total number of mentions.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I(T_i = \hat{T}_i)$$

$$Recall_{l.ma} = \frac{1}{N}\sum_{i=1}^{N} \frac{|T_i \cap \hat{T}_i|}{|T_i|}$$

$$Precision_{l.ma} = \frac{1}{N}\sum_{i=1}^{N} \frac{|T_i \cap \hat{T}_i|}{|\hat{T}_i|}$$

$$Recall_{l.mi} = \frac{\sum_{i=1}^{N}|T_i \cap \hat{T}_i|}{\sum_{m=1}^{N}|T_i|}$$

$$Precision_{l.mi} = \frac{\sum_{i=1}^{N}|T_i \cap \hat{T}_i|}{\sum_{m=1}^{N}|\hat{T}_i|}$$

$F1_{l.ma}$ and $F1_{l.mi}$ are computed as the harmonic mean of the corresponding recall and precision. Accuracy measures the exact matches between the prediction $\hat{T}_i$ and ground truth T. The correctness of partial matching is reflected in $F1_{l.ma}$ and $F1_{l.mi}$. Specifically, types missed by the system will penalize recall, and incorrect predictions will penalize precision. The difference between $F1_{l.ma}$ and $F1_{l.mi}$ is that the later aggregates the statistics overall mentions before computing precision and recall, and thus favors methods that work well on mentions with larger type sets.

A closer inspection of these *de facto* metrics reveals that they are limited in evaluating the effectiveness in recognizing fine-grained types as they aggregate all types and the results can be dominated by coarse-types. To address this, we propose additional metrics that are type-centric and focus on evaluating how well different methods can recognize different types. Specifically, we consider the macro-averaged $F1$ ($F1_{ma}$) and *per-level* $F1_{ma}$ and $F1_{mi}$ by computing the precision/recall for the individual types and aggregate them per level. Let $\mathcal{Y}$ denote all the types in Ψ, $M(y)$ denote the set of all mentions that belong to type y, and $\hat{M}(y)$ denote the set of all mentions that are predicted to belong to type y. Below we give the definitions of overall macro-averaged and micro-averaged precision and recall.

$$Precision_{ma} = \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}}\frac{|M(y)\cap\hat{M}(y)|}{|\hat{M}(y)|}$$

$$Recall_{ma} = \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}}\frac{|M(y)\cap\hat{M}(y)|}{|M(y)|}$$

$$Precision_{mi} = \frac{1}{|\mathcal{Y}|}\frac{\sum_{y\in\mathcal{Y}}|M(y)\cap\hat{M}(y)|}{\sum_{y\in\mathcal{Y}}|\hat{M}(y)|}$$

$$Recall_{mi} = \frac{1}{|\mathcal{Y}|}\frac{\sum_{y\in\mathcal{Y}}|M(y)\cap\hat{M}(y)|}{\sum_{y\in\mathcal{Y}}|M(y)|}$$

By restricting $\mathcal{Y}$ to including only types of a specific level, we can easily compute per-level $F1_{ma}$ and $F1_{mi}$ measures.

Note that $F1_{ma}$ and $F1_{mi}$ reveal different insights about a method's performance. Specifically, $F1_{ma}$ computes per-type precision and recall before averaging over all types. If a method performs well on a few highly populated types but poorly on some rare types, it will score well with $F1_{mi}$ but poorly with $F1_{ma}$. By examining both $F1_{ma}$ and $F1_{mi}$ over all the types as well as level-specific types, we expect more information can be revealed about the strengths and weaknesses of different methods.

4.5.4 Training and Hyper-parameter Tuning

For all methods, we follow the same training and parameter tuning protocol. Specifically, we used the same training set for all methods and randomly sampled 10% of the testing set as the development set. Then we run each of the tested methods using the corresponding publicly available code five times with different random initializations using the same dev/test split and report the mean and the standard deviation of the five runs. For each method, we used the hyper-parameters that are reported for each dataset in the corresponding article. If a method has not been tested and reported previously on a dataset, we perform hyper-parameter tuning using the development set (e.g., we carefully tuned NFETC for BBN).

Method Variations. We tried several variants of the evaluated methods as their source code allows to understand the impact of different components on the performance.

- For Attentive, we consider two variants, with (*Attentive-Hier*) and without (*Attentive*) the hierarchical label encoding respectively.
- For AAA, we consider the original algorithm *AAA* and an additional variant named *AAA-AllC*, which treats all mentions as clean mentions, i.e., with not special handling of the noisy mentions.
- For NFETC, we consider the variant *NFETC-hier(f)*, which filters the noisy mentions before learning. In contrast, we will refer to the original NFETC as *NFETC-hier(r)*, which uses the raw data and separately models the clean and the noisy mentions with different loss functions.

4.6 Results and Discussions

In this section, we present the results of our evaluation and discuss the specific insights that are revealed by the results.

4.6.1 The Effect of Pruning The Noisy Mentions

We start by re-examining the effect of removing the noisy mentions from the training and development data on the performance of the FGET systems. This pruning heuristic was originally proposed and examined by Gillick et al. [19]. Their experimental results showed that applying this heuristic can help greatly in improving the overall performance. More recently, Xu et al. [83] reported the performance of NFETC with and without applying this heuristic. Their results indicate that separately modeling the clean and the noisy mention adds little or no improvement over using the pruning heuristic. These prior results appear to support that this simple pruning heuristic can be as effective as (even outperform) approaches with specially designed objectives for learning from the noisy data.

To put this hypothesis to test under the new evaluation setting, we run *Attentive* on BBN and FIGER with and without applying the pruning heuristic. With pruning, there are some types that are removed completely from the training set as 100% their mentions are noisy. Such types are also removed from the testing set. The resulting filtered testing set is then used to evaluate the systems trained on both the filtered and the raw data for a fair comparison. We selected Attentive because it does not have any special handling of the noisy mentions. That is, it assumes that all

Dataset	Overall					Level-1		Level-2	
	Acc	$F1_{l.ma}$	$F1_{l.mi}$	$F1_{ma}$	$F1_{ma}^{>50\%\text{-noisy}}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$
Filtered data									
BBN	53.8±0.7	72.3±0.4	72.0±0.5	36.9±1.5	35.3±1.3	49.2±1.0	75.7±0.5	29.6±1.5	66.0±1.1
FIGER (org test)	58.4±0.5	75.7±0.6	71.5±0.5	34.5±1.2	34.3±1.6	42.6±1.5	78.1±0.5	28.8±2.5	49.1±0.5
Raw data									
BBN	49.3±0.5	73.4±0.5	72.1±0.5	38.1±1.6	38.9±1.6	45.8±0.3	74.0±0.5	33.6±1.1	68.9±0.5
FIGER (org test)	58.1±1.2	76.8±0.8	73.3±0.7	45.1±0.7	43.1±0.9	51.4±1.0	77.9±0.6	39.8±0.5	58.8±1.2

Table 4.3: A comparison between the performance of Attentive by using the row training data of *BBN* and *FIGER* vs the training data after being filtered from noisy mentions. Performance is evaluated in term of overall Accuracy, $F1_{l.ma}$ and $F1_{l.mi}$, per-level $F1_{ma}$ and $F1_{mi}$, and $F1_{ma}^{>50\%\text{-noisy}}$ which is the $F1$ macro-averaged over the types that more the 50% of their mentions have noisy type sets.

types assigned to the mention are correct. The results are reported in Table 4.3 — the results on the filtered (raw) data are shown on the top (bottom). The first three columns of the results report the de-facto metrics, and the remaining columns report the new metrics that we consider. Focusing on the previously considered metrics, we can see that the pruning heuristic improves the accuracy of both datasets (a significant 4.5% for BBN). Considering the noisy mentions leads to a higher $F1_{l.ma}$ and $F1_{l.mi}$, but the difference is small. This to a large extent is consistent with the findings of [83, 19]. However, by looking at $F1_{ma}$, $F1_{ma}^{>50\%\text{-noisy}}$ (considering only types that have more than 50% of the mentions being noisy) and level-2 $F1_{ma}$ and $F1_{mi}$, we can see that pruning noisy mentions drastically decrease the performance (up to 11%) of the system, especially on the fine-grained types and the types that are dominated by noisy mentions.

To understand how pruning of the noisy mentions affects the performance, we report the detailed recall, precision, and $F1$ for the types that are dominated by noisy mentions in Table 4.4. We removed the types that have zero recall, precision, and $F1$ in both settings from the table. We found that removing the noisy mentions significantly affects the recall for most of these (noisy) types, which is consistent with our expectation because after filtering they tend to have very few training mentions remaining. The precision is also affected for some of these type. There are a few types that gained some improvement by removing the noisy mentions. We notice that most of them have a large number of training mentions, thus eliminating the noisy mentions helped in improving the quilty their training data while still maintaining a sufficiently large number of

Type	Filtered			Raw		
	Precision	*Recall*	$F1$	Precision	*Recall*	$F1$
/time˙	100.00	95.83	97.87	100.00	91.67	95.65
/internet	100.00	50.00	66.67	100.00	50.00	66.67
/internet/website	0	0	0	100.00	50.00	66.67
/law	100.00	50.00	66.67	100.00	50.00	66.67
/person/coach	0	0	0	100.00	50.00	66.67
/building/sports˙facility	0	0	0	100.00	60.00	75.00
/medicine	100.00	50.00	66.67	100.00	66.67	80.00
/education	100.00	20.00	33.33	100.00	33.33	50.00
/transportation	0	0	0	50.00	25.00	33.33
/transportation/road	0	0	0	66.67	50.00	57.14
/news˙agency	0	0	0	50.00	25.00	33.33
/government˙agency	100.00	12.50	22.22	50.00	25.00	33.33
/military	0	0	0	66.67	100.00	80.00
/art	100.00	14.29	25.00	0	0	0
/living˙thing	0	0	0	100.00	33.33	50.00
/written˙work	33.33	28.57	30.77	20.00	20.00	20.00
/people	50.00	60.00	54.55	57.14	80.00	66.67
/people/ethnicity	50.00	60.00	54.55	66.67	80.00	72.73
/building	100.00	07.14	13.33	100.00	35.71	52.63
/location/province	100.00	28.57	44.44	100.00	25.00	40.00
/title	33.33	50.00	40.00	50.00	50.00	50.00
/person/athlete	35.29	60.00	44.44	43.75	70.00	53.85
/organization/sports_team	67.74	77.78	72.41	75.00	75.00	75.00
/event	20.00	28.57	23.53	20.00	25.00	22.22
/organization/company	75.00	46.15	57.14	68.42	48.15	56.52
/location/country	85.71	60.00	70.59	90.00	75.00	81.82
/location/city	72.41	60.00	65.62	85.71	64.86	73.85
/organization	78.46	71.83	75.00	75.35	74.83	75.09
/person	92.54	97.69	95.05	85.31	97.21	90.87
/location	80.56	62.37	70.30	72.73	68.82	70.72

Table 4.4: Precision, recall, and F1 for types that more than 50% of their mentions are noisy using Attentive approach on *filtered* and raw FIGER training data. We omitted the types that get zero precision, recall, and F1 for both data versions. The types that get better F1 by using the raw un-filtered data are in boldface font while the types that get better F1 when the filtered data is used are in italic font.

Approach	Overall			Level-1		Level-2	
	Test Acc	$F1_{l.ma}$	$F1_{l.mi}$	$F1_{ma}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$
Attentive	47.8±1.3	73.1±0.8	71.9±0.9	35.9±2.0	47.2±1.4	74.2±0.5	29.6±2.2	67.9±2.0
Attentive-Hier	48.0±0.7	73.4±0.5	72.3±0.6	36.7±2.0	43.3±1.3	73.8±0.7	32.8±1.4	69.7±0.6
AFET	64.9±0.6	70.7±0.5	71.1±0.7	35.6±1.2	45.0±0.9	74.5±0.5	28.7±1.5	47.8±2.0
AAA-AllC	58.7±0.5	70.9±0.8	72.6±0.9	34.9±1.1	40.8±1.1	75.9±0.9	29.4±1.3	67.1±0.8
AAA	62.4±1.0	70.8±1.0	72.5±0.8	34.4±6.6	40.1±3.5	74.6±1.2	29.3±6.6	69.1±0.5
NFETC-hier(f)	66.4±0.4	72.6±0.3	72.7±0.3	37.7±0.9	48.5±0.4	76.5±0.3	31.7±1.2	67.1±0.3
NFETC-hier(r)	**70.9±0.2**	**75.6±0.2**	**76.4±0.2**	**42.9±1.0**	**50.7±0.7**	**78.2±0.2**	**38.5±1.2**	**73.7±0.2**

Table 4.5: Level-1 , Level-2 and overall performance on BBN.

mentions for effective learning.

4.6.2 Insights From The New Evaluation

In this section, we present the performance of the evaluated methods using the new evaluation metrics on the original and the new testing sets. Table 4.5 reports the overall and per-level performance on BBN using the original testing set. Tables 4.6 and 4.7 report the overall and per-level performance on FIGER and OntoNotes respectively using the original and new testing sets. Tables 4.8 present the results on OntoNotes testing sets after removing the type "other". In the following, we discuss some insights derived from the new experimental results.

Performance on fine-grained types can be masked by the overall performance. First, by looking at the results presented in the tables 4.5, 4.6, 4.7 and 4.8, we can see that the poor performances on fine-grained types are not called out by the commonly used overall performance metrics and by the testing set with poor fine-grained type coverage. Several methods (e.g., AFET on both BBN and FIGER) performed poorly on level-2 and level-3 types, yet the overall performance appears competitive measured by accuracy, $F1_{l.mi}$, and $F1_{l.ma}$. For instance, all methods achieve (61.1% - 71.1%) overall $F1_{l.mi}$ scores on OntoNotes using the original testing set as can be noted by looking at Table 4.7, on the other hand, the $F1_{ma}$ scores achieved for level-2 and level-3 are (14.5% to 42.8%) and (0.8% to 9.7%) respectively. Similar performance gaps can be observed by looking at the results on the new testing set.

The high accuracy values and loose F1 scores don't necessarily imply that the system can adequately recognize different types. It is a common scenario to have skewed class distribu-

Original testing set								
Approach	Overall				Level-1		Level-2	
	Acc	$F1_{l.ma}$	$F1_{l.mi}$	$F1_{ma}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$
Attentive	60.8±1.7	77.8±1.1	74.3±1.5	46.7±4.3	54.6±3.3	78.7±1.2	40.1±3.8	59.6±3.3
Attentive-Hei	60.5±1.3	77.9±1.0	74.3±1.1	47.9±3.1	54.9±2.9	78.8±0.9	41.9±2.8	60.0±2.3
AFET	-	-	-	-	-	-	-	-
AAA-AllC	66.9±0.5	**81.0±0.5**	78.1±0.6	42.2±1.2	50.6±1.3	82.8±0.6	35.1±1.7	62.1±1.9
AAA	65.4±1.0	79.7±0.9	77.3±0.7	**47.8±1.3**	52.7±0.9	81.5±0.9	43.6±1.5	64.5±0.8
NFETC-hier(f)	66.4±1.0	78.7±1.2	75.5±1.2	47.0±2.0	**55.9±1.8**	79.7±1.2	**42.9±2.8**	62.9±1.3
NFETC-hier(r)	**68.8±1.2**	**81.0±0.9**	**78.2±0.8**	46.7±1.7	55.3±2.5	**81.9±0.8**	42.8±1.6	**67.0±0.7**
New testing set								
Attentive	50.8±0.8	77.0±0.2	75.6±0.2	66.5±0.6	65.7±1.9	80.1±0.4	66.7±0.7	68.3±0.5
Attentiv-Heir	51.8±0.5	77.4±0.3	76.1±0.3	67.2±0.9	64.8±0.8	80.1±0.2	68.4±0.9	69.8±0.6
AFET	-	-	-	-	-	-	-	-
AAA-AllC	69.4±0.3	**85.3±0.4**	**85.3±0.4**	66.8±0.5	65.1±1.8	**89.6±0.5**	67.5±0.3	78.2±0.4
AAA	**71.1±0.9**	84.6±0.3	84.9±0.2	**71.9±1.2**	**67.6±2.1**	88.4±0.3	**74.0±0.9**	**79.4±0.6**
NFETC-hier(f)	63.1±0.8	82.6±0.5	82.1±0.5	62.2±0.8	66.0±1.9	88.4±0.4	60.3±1.1	71.8±0.7
NFETC-hier(r)	61.9±0.7	82.2±0.4	81.5±0.3	58.8±1.1	65.5±0.5	88.6±0.5	55.4±2.0	69.6±0.8

Table 4.6: Level-1 , Level-2 and overall performance on FIGER for the original and the new testing sets.

Original testing set										
Approach	Overall			Level-1		Level-2		Level-3		
	Acc	$F1_{l.ma}$	$F1_{l.mi}$	$F1_{ma}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$
Attentive	50.6±0.4	67.2±0.5	60.5±0.7	8.6±1.8	59.5±2.5	71.9±0.6	7.8±1.4	13.5±3.9	0.8±1.5	2.6±5.2
Attentiv-Heir	50.7±0.3	67.2±1.2	61.0±1.8	13.0±2.6	58.8±6.2	71.5±1.7	12.0±2.2	22.8±5.2	6.4±1.8	13.2±2.4
AFET	57.1±0.3	74.8±0.3	68.1±0.3	8.9±0.4	69.4±0.4	79.9±0.2	8.2±0.2	21.6±1.1	0.4±0.8	2.2±4.4
AAA-AllC	54.4±1.2	72.6±0.8	66.7±0.6	11.7±1.7	67.8±0.9	78.0±0.7	12.9±1.7	29.2±1.5	1.1±1.1	0.8±1.0s
AAA	53.3±0.5	71.4±0.8	65.9±0.7	14.9±2.1	66.7±0.8	76.7±0.9	15.6±2.1	31.8±1.7	5.0±2.2	15.5±2.9
NFETC-hier(f)	59.7±0.2	76.0±0.3	69.7±0.3	**21.1±1.0**	71.7±0.6	80.6±0.4	**23.2±0.7**	40.6±0.4	**9.7±0.9**	16.6±3.4
NFETC-hier(r)	**60.3±0.4**	**76.2±0.3**	**70.1±0.4**	20.5±0.7	**72.1±0.4**	**80.6±0.3**	23.1±1.3	**42.8±0.4**	7.6±0.9	**18.0±0.7**
New testing set										
Attentive	51.0±0.6	70.0±0.6	63.7±0.8	20.1±5.1	65.4±1.1	75.4±0.7	23.3±5.4	25.4±4.0	9.9±3.4	12.5±5.0
Attentiv-Heir	49.7±0.4	68.3±0.7	62.5±0.8	27.3±3.4	63.7±2.0	73.3±0.9	28.7±2.3	28.9±1.0	20.5±3.5	22.0±2.9
AFET	54.9±0.2	74.6±0.1	67.7±0.2	10.1±0.3	71.4±0.3	80.8±0.2	10.5±0.3	21.5±0.6	0.9±0.7	2.9±3.1
AAA-AllC	54.1±0.5	73.0±0.5	67.3±0.6	18.6±4.2	70.2±0.4	78.9±0.3	22.1±5.1	33.4±2.2	7.9±2.8	8.0±3.7
AAA	54.4±0.3	72.5±0.4	66.8±0.3	17.4±1.8	69.1±0.4	78.2±0.4	20.2±2.2	33.4±1.2	7.6±1.8	13.4±1.7
NFETC-hier(f)	**59.4±0.2**	**75.9±0.2**	69.9±0.2	29.6±0.5	**73.1±0.2**	**80.9±0.2**	38.0±0.6	44.7±0.3	14.5±1.2	20.8±1.6
NFETC-hier(r)	59.2±0.1	75.7±0.2	**69.9±0.1**	**32.7±1.3**	73.1±0.3	80.7±0.2	**41.4±1.3**	**44.8±0.2**	**17.4±1.5**	**23.0±1.6**

Table 4.7: Level-1 , Level-2, Level-3 and overall performance on OntoNotes for the original and the new testing sets.

Approach	Overall			Level-1			Level-2		Level-3	
	Acc	$F1_{l.ma}$	$F1_{l.mi}$	$F1_{ma}$	$F1_{ma}$	$F1_{mi}$	$F1_{ma}$	$F1_{mi}$		
Original testing set										
Attentive	20.2±0.3	51.4±0.5	49.7±0.6	14.4±0.1	63.0±0.8	61.5±0.6	13.4±0.3	28.3±1.4	6.9±0.2	14.9±0.4
Attentiv-Heir	19.5±0.4	50.8±0.7	49.3±0.9	15.1±0.5	62.5±1.0	60.5±0.7	14.9±0.7	29.5±1.2	6.8±0.3	14.5±0.8
AFET	17.3±0.4	50.4±0.9	49.5±1.0	9.0±0.2	67.8±1.3	63.4±1.0	9.1±0.1	22.0±1.0	0.0±0.0	0.0±0.0
AAA-AllC	24.9±0.3	56.3±0.3	55.1±0.3	14.8±0.9	68.3±0.3	67.4±0.3	17.7±1.2	34.7±0.5	2.6±0.7	1.8±0.8
AAA	26.8±1.0	56.1±0.1	55.2±0.2	18.7±0.9	67.0±0.5	66.1±0.5	20.5±1.6	38.3±1.3	8.3±0.6	20.2±1.1
NFETC-hier(f)	31.3±0.8	58.3±0.6	57.4±0.5	**23.1±1.1**	69.6±0.6	67.3±0.5	**27.0±0.9**	42.7±0.6	10.4±0.6	18.8±0.6
NFETC-hier(r)	**32.7±0.2**	**59.9±0.2**	**59.1±0.3**	21.6±1.0	**70.5±0.3**	**68.9±0.3**	24.5±1.3	**45.9±0.3**	**10.5±0.4**	**20.7±0.9**
New testing set										
Attentive	20.5±0.1	53.1±0.3	51.3±0.3	28.3±1.1	65.2±0.7	63.6±0.5	32.6±0.7	30.5±0.9	17.8±1.5	20.5±0.7
Attentive-Heir	19.8±0.4	52.9±0.3	51.4±0.4	31.8±1.5	64.6±0.4	63.1±0.3	32.7±1.0	31.4±0.9	25.9±1.8	26.7±1.5
AFET	16.7±0.2	51.4±0.2	50.2±0.3	9.5±0.4	68.9±0.3	65.8±0.2	10.1±0.4	21.6±0.8	0.6±0.6	1.7±2.8
AAA-AllC	25.6±0.5	57.9±0.5	56.6±0.4	23.2±0.6	70.5±0.5	69.8±0.5	27.1±0.6	36.2±0.6	12.6±1.3	12.4±1.3
AAA	27.1±0.7	57.6±0.3	56.6±0.3	26.4±2.5	69.3±0.4	71.6±0.5	30.1±1.7	40.0±0.3	16.4±2.7	21.3±3.0
NFETC-hier(f)	33.3±0.4	61.0±0.4	59.7±0.3	**30.8±0.2**	70.5±0.5	71.6±0.5	**40.0±0.3**	45.8±0.4	15.1±0.7	17.7±2.9
NFETC-hier(r)	**33.9±0.4**	**62.2±0.5**	**60.9±0.4**	30.8±0.5	**72.6±0.6**	**71.8±0.5**	39.5±0.9	**47.3±0.1**	**15.7±1.2**	**24.0±1.1**

Table 4.8: Level-1 , Level-2, Level-3 and overall performance on OntoNotes for the original and the new testing sets after removing the type "other".

tion in the testing set. If the system performs well on the heavily populated types, it achieves high overall performance. BBN is a good example where the two types "/gpe/city" and "/organization/corporation" represent 46.08% of the testing set. Most of the methods perform well on these two types. E.g., the $F1$ scores achieved by NFETC-hier(r) on "/gpe/city" and "/organization/corporation" are 81.37% and 80.74% respectively. The nuanced differences between methods cannot be revealed by just reporting overall accuracy, $F1_{l.ma}$, and $F1_{l.mi}$. On the other hand, when looking at $F1_{.ma}$ and level-2 $F1_{.ma}$ in Table 4.5 we can see that, in average, most of the methods don't recognize more ¼ of the level-2 types.

Current evaluations can't adequately assess the contribution of different component of the systems. Comparing AAA-AllC vs. AAA as shown in Tables 4.6, 4.7, we notice that when evaluated by the overall performance, AAA appears to have little or no advantage over AAA-AllC, which treats the noisy training data as clean, especially on the original testing sets. In fact, AAA performed worse than AAA-AllC on FIGER and OntoNotes original testing sets. This seems to suggest that the noisy training data does not really hurt the performance and we can simply treat them as clean. However, when comparing the performance on level-2 and level-3 types using the original testing sets, and by looking at the results of the new testing sets as

shown by Tables 4.6, 4.7, we see noticeable gain by AAA compared to AAA-AllC, suggesting that careful treatment of the noisy training data will likely have a stronger impact on fine-grained types.

Another example is when we compare Attentive vs. Attentive-Heir. By looking at the overall performance, we can see that using the hierarchical label encoding adds a very slight improvement to the accuracy of Attentive on FIGER (using new testing set) as presented in Table 4.6, or even hurt the performance as for BBN and OntoNotes (shown by Tables 4.5, 4.7). On the other hand, Attentive-Heir significantly outperforms Attentive by observing the overall, level-2 and level3 $F1_{ma}$ in most of the cases.

Test data composition has a huge impact on the overall performances. We see much flipping of the ordering of methods when looking at the overall accuracy and $F1_{l.ma}$ and $F1_{l.mi}$ when the testing set used for evaluation is altered. For example, by comparing the overall and per-level performance of OntoNotes before and after removing the type "other" as shown in Tables 4.7 and 4.8, we can observe that removing the type "other" from OntoNotes original and new testing sets significantly changes the relative performance of AFET in comparison to other methods.

Another example can be seen by examining the overall and per-level performance of AAA vs. NFETC-hier(r) on the original testing set vs. the new testing set of FIGER presented in Table 4.6. Using the original testing sets, NFETC-hier(r) outperforms AAA consistently over all datasets using all metrics. However, AAA gains a substantial improvement in level-2 $F1_{ma}$ (17.9%) and the accuracy (9.1%) over NFETC-hier(r) on the new testing set of FIGER which basically has the evenest coverage of the fine-grained types among all testing sets used in this study. This suggests that when it comes to the effectiveness of FGET systems on the fine-grained types, AAA is a highly competitive method.

The new metrics are less sensitive to the dominant types. Table 4.7 reports the performance on OntoNotes for both the original and the augmented testing sets. Table 4.8 present the performance on both the original and the augmented testing sets of Ontonotes after removing the type "other". By comparing the results in these tables, we can see that the overall performance dropped significantly(about 30% performance degradation) for both original and augmented testing set when the type "other" is removed. On the other hand, we can see that the newly introduced evaluation metrics are less sensitive to that dominant types (e.g., "other").

Misidentified mention (#occur)	Label (#occur)	Training example	Potential KB entry
uh (580)	/location/geography /body_of_water (839)	**[Uh]**, that was the gist of Mei Shirong 's remarks at the end of the report	Uh is the Slovak name of Uzh River in Ukraine and Slovakia
yes (245)	/organization/music (1390)	**[Yes]**, eh, just now Director Xing mentioned a very important issue	Yes (band)
who (161)	/organization/music (839)	**[Who]** else was in attendance at the time ?	The Who, an English rock band
it (2042)	/other/internet (4356)	**[It]** is both a European and Asian nation	the Internet top-level domain for Italy
As (591)	/other/scientific (4025)	**[As]** a matter of fact, Professor Liu has already been affected to some extent	Arsenic (As), a chemical element

Table 4.9: Examples of identification noise in OntoNotes' training set including misidentified mention and #occurrences, the assigned type label and #occurrences of that label, an example from the training set, and a potential KB entry to which it was linked during generating the training data by distant supervision

Example	Assigned type	Correct type
The production of internal - combustion engines, metal-cutting machines, large, medium and small-sized tractors, vans, **[motor trucks]**, etc.	/other	/other/product/car
The capability of existing fields to deliver oil is dropping,and oil exploration activity is also down dramatically , as many producers shifttheir emphasisto **[natural gas]**, said Ronald Watkins.	/other	/other/product
[Harvard Law School Professor Laurence Tribe] says, there is a " generation - skipping " flavor to current dissents	/person	/person/title
[Officials in the Iraqi Department of Defense] said that	/person	/person/title

Table 4.10: Examples of mentions in testing set that are assigned to higer level types while they can be assigned to more fine-grained types.

OntoNotes has a major issue to be used for training and evaluating FGET systems. The performance drops after removing the type "other" from the testing set of OntoNotes encouraged us to take a closer look at both train and testing set of OntoNotes. In addition to the typing noise, we observe that the training data of OntoNotes has a substantial *identification noise* that comes from incorrect identification of the mentions. We found that there is a considerable number of completely incorrect identifications which we think are more harmful to the learning process more than the typing noise. Table 4.9 lists examples in the mention identification noise along with potential reasons of why they are linked to the knowledge base. Another issue can be observed by looking at the testing set. We found that a lot of mentions that are labeled under coarse types can be labeled under fine-grained types. By referring to the paper that describes how OntoNotes testing set is originally annotated [19], it is mentioned that the annotators back off all the confusing mentions to their coarse types. This means that even if a correct fine-grained type is predicted for these test mentions, it is considered as an erroneous prediction and affects both accuracy and F1 scores, Table 4.10 lists some examples from the original testing set of OntoNotes.

4.7 Conclusion

Tagging entity mentions in the text with fine-grained types is a beneficial step for a variety of downstream NLP applications. Several FGET approaches have been proposed in the literature. Most prior works are evaluated using FIGER, OntoNotes, and BBN benchmark datasets. However, the current empirical evaluation only reports the overall performance on all types and fail to assess the ability of these systems to recognize fine-grained types. Moreover, the testing sets for FIGER and OntoNotes have very poor coverage of the fine-grained types, which means that these systems are mostly evaluated by their performance on the coarse types. In this work, we present a new empirical study that re-evaluates the most recently proposed FGET methods by augmenting the current testing set to have almost a full coverage of the fine-grained types and examining not only the overall performance but also per-level, type-specific performance. The new experimental results reveal very interesting observations about the tested methods. For example, we found that the performance on fine-grained types is not always consistent with overall performance; systems with high overall scores are not necessarily the best for recognizing the fine-grained types, or for sufficiently covering different types. Moreover, current evaluations do not adequately assess the contribution of different components of the systems. Furthermore, the

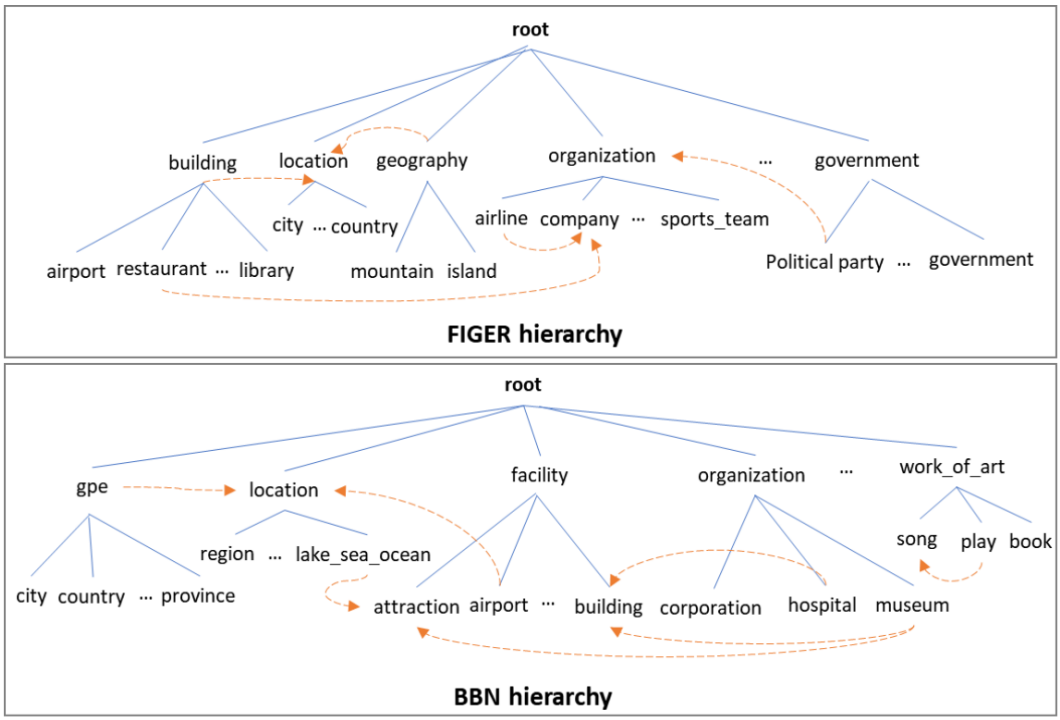

Figure 4.2: Examples of auxiliary connections (shown by the dashed arrows) between types in the type hierarchies of FIGRE and BBN datasets.

type distribution in the test data has a significant impact on the overall performances, which can be observed by the flipping orders for the tested method when the new test data is introduced. We also observed that our new metrics are more rubost to the dominat types. Finally, we note several issues with OntoNotes datasets that are not previously revealed.One of these issues is the incorrect identification of mentions. We leave dealing with the identification noise to future work.

We also observed that the definition of clean mentions followed by all the methods that separately model the clean and the noisy mention is not always correct. As mentioned before, Ren et al. [58] define the clean mentions as the mentions with types (assigned by distant supervision) form a single path in the type hierarchy; otherwise, it is noisy. However, we found that the types in type set of mention could truly belong to several paths in the tree. For example, by looking at a part of the type hierarchy of *FIGER*, in Figure 4.2, we can see the *political party* is a sub-type of *government*, However, it is known that political parties are organizations, and this is consistent

with the fact that the *government/political party* always co-occur with the type *organization* in the training set. Similar examples can be found in *BBN*. Moreover, we can see that about 7% of the mention in FIGER test set (which is manually annotated) have types that don't constitute paths in the tree, these mentions are missed by most of the proposed FGET methods. As future work, we would extend the type hierarchy to include these correct assignments by adding *auxiliary connections* between the nodes in the type hierarchy and develop an FGET system that uses these connections to improve training and type inference.

Chapter 5: Conclusion

Despite the noticeable gain coming from applying deep learning models to NLP applications, it difficult to be translated into real-world settings. In real NLP applications, we always face the challenge of not having enough high-quality labeled data available because of the high annotation cost, and the need for reannotating the hand-labeled training data partially or completely as the modeling goals change. As a result, techniques that address how to learn effectively from limited or cheaply collected labeled data are receiving growing attention in the NLP community. This research is a manuscript of scientific articles through which we propose methods to address the data scarcity problem in supervised learning for NLP applications.

Our first work (chapter 2) proposes *Transfer Learning* approaches to transfer the knowledge between domains with variant but related label space. The goal is to model the relationships between the labels of the source and the target domains represented by a bipartite graph and use it to enhance the transfer learning. We utilize the bipartite graph to map the source examples to corresponding target labels. Because the source mapped examples could be assigned multiple target labels where not all of them are necessarily true, we examine several weak supervision objectives for training neural networks for the target domain on both the target and the mapped source examples. We compare them to the standard deep transfer learning techniques such as fine-tuning and multi-tasking. We apply our methods on two NLP tasks: Event Typing and Text Classification and created several synthetic label-relationship scenarios to rigorously study the strengths and the weaknesses of our methods in comparison with the baselines.

We demonstrate that all the transfer learning methods provide a significant improvement over training a model on the target data only. We experimentally proved that in most cases, the label structure can provide considerably useful information to improve the knowledge transfer between the domains. Our weak supervision methods outperform the standard deep transfer learning methods by a substantial margin when the relationships between the source and the target labels are 1-to-1, n-to-1 or a mix of relationships. For the 1-to-n, the performance of our approach depends highly on the *ambiguity degree* of the mapped example label set. With a moderate label ambiguity, our method is still beneficial over the other methods. On the other hand, learning from an example with highly ambiguous candidate label set is challenging espe-

cially when compounded with having few examples under these labels in the target as it is hard to differentiate these labels during training. Additionally, we found that source-to-target label relationships affect the performance of the standard deep transfer learning methods: they work better if the source-to-target relationships are 1-to-1 or n-to-1. Furthermore, the improvement margin that can be obtained by transfer learning in general decreases as the size of the target data increases. However, we found that traditional deep transfer learning techniques are more sensitive to the size of the target data than the mapping-based methods, as the later can maintain a reasonable improvement with varying target set sizes.

In Chapter 2, we propose a *Zero-Shot Fine-grained Entity Typing* approach that utilizes the description of the types available from *Wikipedia* to recognize new types, requiring zero-training examples. At training time, our methods learn a compatibility function to project the mentions and the descriptions of the types presented in the training data into a shared semantic space. At testing time, the learned compatibility function paired with the description of the new types can be used to recognize that type without the need for new training examples. We compare our method with two previously proposed methods on two benchmark datasets and found that our approach outperforms these methods substantially and consistently on the fine-grained types.

We examine several methods to construct a type representation from the type's Wikipedia description and found that representing the type by a bag of representations extracted from the corresponding description combined with sequence modeling gives marginally better results than the simple averaging techniques. Finally, we analyze the effect of the description length on our method's performance for individual types and found that it works better if the type description is not too short or not too long. The short description can be inadequate in capturing sufficient semantics of the types while the long description can overlap with other types' descriptions making it hard to distinguish between highly related types. Extracting high-quality representations from the noisy Wikipedia descriptions of the types draw future directions for further improvements.

In Chapter 3, we present a new empirical study that re-evaluates the most recently proposed FGET methods to overcome two primary limitations of the current evaluations. First, the testing sets of some of the benchmarks have poor coverage of the fine-grained types. Second, they only report the overall performance on all types ignoring how the performance of these methods may vary on the types from different levels. To overcome these limitations, we augment the current testing sets to have substantially better coverage of the fine-grained types. We also examine not only the overall performance but also per-level, type-specific performance. The new experimental results reveal very interesting observations about the tested methods. We found that the

performance on fine-grained types is not always consistent with overall performance; systems with high overall scores are not necessarily the best for recognizing the fine-grained types, or sufficient covering different types. Moreover, current evaluations don't adequately assess the contribution of different component of the systems. For examples, it turns out that separately modeling the clean and the noisy mentions significantly improve the performance of the fine-grained types, this conclusion is masked by the previously reported overall scores. Furthermore, the testing data composition has an enormous impact on the overall performances, as the ranking of some of the tested method change by switching to the new testing sets. We also observed several issues with OntoNotes datasets that are not previously revealed such as the incorrect identification of the mentions. As future work, we would try to focus on cleaning OntoNotes from Identification noise. Also, we would try to enhance the type hierarchy by adding auxiliary connections between types from different paths and develop FGET system that utilizes these connections to enhance the training and the type inference.

Bibliography

[1] Abhishek Abhishek, Ashish Anand, and Amit Awekar. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 797–807, 2017.

[2] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. 2019.

[3] Himanshu Sharad Bhatt, Manjira Sinha, and Shourya Roy. Cross-domain text classification with multiple domains and disparate label sets. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1641–1650, 2016.

[4] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[5] Joseph Chee Chang, Saleema Amershi, and Ece Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2334–2346. ACM, 2017.

[6] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[7] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. Ultra-fine entity typing. In *In Annual Meeting of the Association for Computational Linguistics*, 2018.

[8] Luciano del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. Finet: Context-aware fine-grained named entity typing. Assoc. for Computational Linguistics, 2015.

[9] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12(May):1501–1536, 2011.

[10] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[11] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

[12] Amit Das and Mark Hasegawa-Johnson. Cross-lingual transfer learning during supervised training in low resource scenarios. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[13] Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. Question answering on knowledge bases and text using universal schema and memory networks. *arXiv preprint arXiv:1704.08384*, 2017.

[14] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[15] Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. A hybrid neural model for type classification of entity mentions.

[16] André Elisseeff and Jason Weston. A kernel method for multi-labeled classification. In *Advances in neural information processing systems*, pages 681–687, 2002.

[17] Zhenyong Fu, Tao Xiang, Elyor Kodirov, and Shaogang Gong. Zero-shot object recognition by semantic manifold distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2635–2644, 2015.

[18] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

[19] Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*, 2014.

[20] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[21] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

[22] Sangdo Han, Soonchoul Kwon, Hwanjo Yu, and Gary Geunbae Lee. Answer ranking based on named entity types for question answering. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, page 71. ACM, 2017.

[23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *arXiv preprint arXiv:1902.00751*, 2019.

[24] Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R Voss. Zero-shot transfer learning for event extractions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1, 2018.

[25] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271, 2007.

[26] Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202, 2017.

[27] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[28] Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 473–482, 2015.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2452–2460, 2015.

[31] Vijay Krishnan and Christopher D Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1121–1128. Association for Computational Linguistics, 2006.

[32] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2014.

[33] Matthew Lease. On quality control and machine learning in crowdsourcing. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[34] Sungjin Lee and Rahul Jha. Zero-shot adaptive transfer for conversational language understanding. *arXiv preprint arXiv:1808.10059*, 2018.

[35] Zhenyang Li, Efstratios Gavves, Thomas Mensink, and Cees GM Snoek. Attributes make sense on segmented objects. In *European Conference on Computer Vision*, pages 350–365. Springer, 2014.

[36] Zhenyang Li, Ran Tao, Efstratios Gavves, Cees GM Snoek, and Arnold WM Smeulders. Tracking by natural language specification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7350–7358. IEEE, 2017.

[37] Xiao Ling and Daniel S Weld. Fine-grained entity recognition. In *AAAI*, volume 12, pages 94–100, 2012.

[38] Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015.

[40] Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2107–2116, 2014.

[41] Yukun Ma, Erik Cambria, and Sa Gao. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180, 2016.

[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[43] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 777–782, 2013.

[44] Tzu Ming Harry Hsu, Wei Yu Chen, Cheng-An Hou, Yao-Hung Hubert Tsai, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Unsupervised domain adaptation with imbalanced cross-

domain data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4121–4129, 2015.

[45] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 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, 2009.

[46] Seungwhan Moon and Jaime Carbonell. Proactive transfer learning for heterogeneous feature and label spaces. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 706–721. Springer, 2016.

[47] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in NLP applications? *arXiv preprint arXiv:1603.06111*, 2016.

[48] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[49] Shikhar Murty, Patrick Verga, Luke Vilnis, Irena Radovanovic, and Andrew McCallum. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 97–109, 2018.

[50] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 365–371, 2015.

[51] Rasha Obeidat, Xiaoli Z Fern, and Prasad Tadepalli. Label embedding approach for transfer learning. In *ICBO/BioCreative*, 2016.

[52] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

[53] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[54] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[55] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[56] Maxim Rabinovich and Dan Klein. Fine-grained entity typing with high-multiplicity assignments. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 330–334. Association for Computational Linguistics, 2017.

[57] Jonathan Raphael Raiman and Olivier Michel Raiman. Deeptype: multilingual entity linking by neural type system evolution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[58] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378, 2016.

[59] Marcus Rohrbach, Michael Stark, and Bernt Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR 2011*, pages 1641–1648. IEEE, 2011.

[60] Rakesh Kumar Sanodiya, Jimson Mathew, Sriparna Saha, and Michelle Davies Thalakottur. A new transfer learning algorithm in semi-supervised setting. *IEEE Access*, 7:42956–42967, 2019.

[61] Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6965–6969. IEEE, 2013.

[62] Hamed Shahbazi, Xiaoli Z Fern, Reza Ghaeini, Chao Ma, Rasha Obeidat, and Prasad Tadepalli. Joint neural entity disambiguation with output space search. pages 2170–2180, 2018.

[63] Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. Neural architectures for fine-grained entity type classification. In *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2017.

[64] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[65] Yangqiu Song, Shyam Upadhyay, Haoruo Peng, Stephen Mayhew, and Dan Roth. Toward any-language zero-shot topic classification of textual documents. *Artificial Intelligence*, 274:133–150, 2019.

[66] Shashank Srivastava, Igor Labutov, and Tom Mitchell. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316, 2018.

[67] Shashank Srivastava, Igor Labutov, and Tom Mitchell. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 306–316, 2018.

[68] Rosa Stern, Benoît Sagot, and Frédéric Béchet. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60. Association for Computational Linguistics, 2012.

[69] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

[70] 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*, pages 142–147. Association for Computational Linguistics, 2003.

[71] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2255–2265, 2017.

[72] Yao-Hung Hubert Tsai, Cheng-An Hou, Wei-Yu Chen, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Domain-constraint transfer coding for imbalanced unsupervised domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[73] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[74] Dong Wang and Thomas Fang Zheng. Transfer learning for speech and language processing. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1225–1237. IEEE, 2015.

[75] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1129–1134. IEEE, 2017.

[76] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):13, 2019.

[77] William Yang Wang and Diyi Yang. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, 2015.

[78] Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. OntoNotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer*, page 59, 2011.

[79] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[80] Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4460–4464. IEEE, 2015.

[81] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[82] Guo-Sen Xie, Li Liu, Xiaobo Jin, Fan Zhu, Zheng Zhang, Jie Qin, Yazhou Yao, and Ling Shao. Attentive region embedding network for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9384–9393, 2019.

[83] Peng Xu and Denilson Barbosa. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 16–25, 2018.

[84] Yahui Xu, Yang Yang, Fumin Shen, Xing Xu, Yuxuan Zhou, and Heng Tao Shen. Attribute hashing for zero-shot image retrieval. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 133–138. IEEE, 2017.

[85] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. Noise mitigation for neural entity typing and relation extraction. *arXiv preprint arXiv:1612.07495*, 2016.

[86] Emre Yılmaz, Henk van den Heuvel, and David van Leeuwen. Code-switching detection using multilingual dnns. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 610–616. IEEE, 2016.

[87] Dani Yogatama, Daniel 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 (Volume 2: Short Papers)*, volume 2, pages 291–296, 2015.

[88] Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. Hyena: Hierarchical type classification for entity names. *Proceedings of COLING 2012: Posters*, pages 1361–1370, 2012.

[89] Zheng Yuan and Doug Downey. Otyper: A neural architecture for open named entity typing. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[90] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762, 2015.

[91] Denghui Zhang, Manling Li, Pengshan Cai, Yantao Jia, and Yuanzhuo Wang. Path-based attention neural model for fine-grained entity typing. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[92] Haofeng Zhang, Yang Long, and Ling Shao. Zero-shot hashing with orthogonal projection for image retrieval. *Pattern Recognition Letters*, 2018.

[93] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[94] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 4166–4174, 2015.

[95] Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2065–2076, 2018.

[96] Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Shen-Shyang Ho. Transfer learning for cross-language text categorization through active correspondences construction. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[97] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.

[98] Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.