Enhanced Representations for Relations by Multi-task Learning

by

Lisheng Fu

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January 2020

Professor Ralph Grishman

Dedication

I want to thank my father for encouraging me to pursue a PhD. He got his part-time PhD in the age of 40s. It is quite inspiring for me to pursue knowledge.

I want to thank my mother for telling me I can go home whenever I meet difficulties in my life.

Acknowledgments

There are a lot of people who have helped me in my PhD life. First, I want to thank my advisor Ralph Grishman. He is the one who led me into the research community and taught me to come up with my own ideas. I have had quite a few lab mates. I want to thank Thien Huu Nguyen first who shared a lot of ideas and discussions with me. I want to thank our previous post-doc Yifan He who helped me to understand more about research in the early years of my PhD program. I want to thank our previous graduate Bonan Min who encouraged me to publish more papers and actively helped me for my papers.

I want to thank Professor Adam Meyers and Professor Satoshi Sekine for sharing high-level ideas. I want to thank my fellow PhD students (Kai Cao, Xiang Li, Maria Pershina) for having fun in our PhD lives. I want to thank previous graduates (Shasha Liao and Ang Sun) for guidance at the beginning of my PhD program. I want to thank previous graduates and now faculties (Wei Xu and Heng Ji). Even though I never directly work with them, their active work in the research community has been very inspiring to me. I have attended quite a few good courses at NYU. I want to thank Professor Kyunghyun Cho for his course Natural Language Understanding with Distributed Representations and Doctor Slav Petrov for his course Statistical Natural Language Processing. Both courses are very helpful for my PhD research.

During my PhD program, I also did a few internships to work on a variety of problems. I want to thank all my advisors (Bo Pang, Octavian Popescu, Alexandre Lacoste, Victoria Fossum, Pablo Barrio) who worked with me in a short period of time. I have enjoyed all my internships and learned a lot from them.

Last, I want to thank all my committee members (Professor Ralph Grishman, Professor Adam Meyers, Professor Sam Bowman, Professor Kyunghyun Cho, Professor Thien Huu Nguyen) again for their suggestions. Especially, I want to thank Professor Sam Bowman for his careful review of my thesis.

Abstract

A relation describes the relationship between a pair of entities. Relation Extraction is the process of extracting relations from free text and converting them to structured machine-readable knowledge. This process can facilitate building and extending knowledge bases, and therefore can benefit a variety of natural language processing applications such as Question Answering and Summarization.

Typical relation extraction projects start by defining a relation schema: a set of mutually-exclusive relation types. Based on these definitions, all instances of these relations in a text corpus are labeled by hand, producing a dataset which can be used to train a statistical model. Labeling relations in text is difficult and time-consuming. There only exist limited relation datasets developed in this way. New applications will give rise to new schemas, so the lack of high-quality labeled data is almost inevitable for Relation Extraction.

Despite limited labeled samples in relation datasets, neural net models have been shown to be more effective than traditional methods in learning feature representations with pre-trained word embeddings. In the context of representation learning, this thesis presents multi-task learning frameworks to learn enhanced representations for relations. It shows how to learn better feature representations in both unsupervised and supervised ways. First, the dissertation shows how to learn domain invariant representations using unlabeled entity pairs. Then it shows how to learn a unified encoder by combining multiple annotated datasets. Finally, it shows how to learn the relatedness between relation types across different relation schemas. These techniques improve the relation models without requiring more annotation from the target dataset. The multi-task learning frameworks could be an efficient toolkit for relation extraction in general.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

A relation represents the relationship between two entities (E.g., *Smith* went to a conference in *Brazil*). Relation types can be defined based on a certain Ontology or users' interests. Relations extracted as structured forms such as a knowledge graph can serve downstream applications. Relation types can be defined at different levels of granularity. They can be fine-grained and literal such as */business/company/place_founded* in the NYT10 dataset (Riedel et al., 2010). They can also be coarse and abstract such as *Organization-affiliation* in the ACE 2005 datasets,[1] which includes a few kinds of relationships such that a person's working for a company or a country's being a member of the United Nations. There can be a lot of ways to express relations even for the same type, and some of them domain-specific. For example, different job titles imply the same employment relation. Extracting relations from text is a difficult problem and has been studied

1. ACE 2003-2008 task descriptions are available through LDC at `https://www.ldc.upenn.edu/collaborations/past-projects/ace`

since MUC-7 (1998).[2]

To solve this relation extraction problem, people developed hand-annotated datasets (e.g., MUC, ACE, ERE) and distantly supervised datasets (e.g., NYT10). However, we still only have small high-quality data for learning. Same as many other natural language processing tasks, it is time-consuming to annotate relations in text. The distant supervised data is created by aligning entities from a knowledge base to text. It can create larger amount of data but noisy, which yields limited accuracy. In addition, the relation schema and the definition of relation types changed from year to year for some existing hand-annotated datasets such as ACE, which makes it hard to combine directly. Even if we have a large corpus with a coherent relation schema, people can always define new relation for their own purposes. The lack of high quality training data for relations seems to be inevitable in practice despite that there exist multiple resources for relations.

In this thesis, I explore methods to utilize these resources together. I propose multi-task learning frameworks to cope with the problem of lack of labeled relations. The multi-task models will benefit from data augmentation and regularization from multiple resources. In the following chapters, I will first review the history of Relation Extraction from traditional feature-based and kernel-based methods to more recent neural net based models. Then I will introduce multi-task learning frameworks in the context of neural models. I will start from the unsupervised setting where I try to learn domain-invariant representations using unlabeled entity pairs from different domains. The learned representations can

2. Proceedings for MUC 3-7 are available through the ACL website at `https://www.aclweb.org/anthology/venues/muc/`

be better generalized to new domains. In the supervised setting, I combine two datasets with similar relation schemas and train them together in a multi-task model. Thanks to more labeled training data, it can learn feature representations that work better for both relation tasks. Finally, I try to learn the relationship between relation types across different datasets. By learning the relatedness between the relation types, the model obtains more knowledge from the auxiliary types and learn better representations of the label embeddings. The experiments show larger improvement in the low-resource settings.

1.1 Relation Extraction

A relation describes a relationship about a pair of entities. E.g.,

- *An interviewer from The Patriot Ledger* (Employment)
- *George Bush traveled to France on Thursday for a summit* (Located)
- *The U.S. Congress* (Subsidiary)

In information extraction, relations typically represent permanent information or information of extended duration in contrast to events. Relation detection and classification task was first introduced in MUC-7 (1998). It covered only three types of relations involving organizations: location_of, employee_of, and product_of. The types of relations were extended and more extensively studied in ACE. The annotation guidelines were revised repeatedly every year. Most previous research published results for ACE 2003-2005 datasets. More recently, the task definition has been further refined in ERE (Song et al., 2015) . Because of the inconsistent relation schema and annotation guidelines, these datasets can not be combined directly. The ACE 2005 task had the following types and subtypes (Table 1.1).

Type	**Subtype**
ART (artifact)	User-Owner-Inventor-Manufacturer
GEN-AFF (Gen-affiliation)	Citizen-Resident-Religion-Ethnicity, Org-Location
ORG-AFF (Org-affiliation)	Employment, Founder, Ownership, Student-Alum, Sports-Affiliation, Investor-Shareholder, Membership
PART-WHOLE (part-whole)	Artifact, Geographical, Subsidiary
PER-SOC (person-social)	Business, Family, Lasting-Personal
PHYS (physical)	Located, Near

Table 1.1: ACE 2005 relation schema.

One property has been consistent in these guidelines. The pair of entities and the semantic relationship have to be explicitly expressed within a single sentence. In knowledge base construction, the relationships between entities can be inferred from multiple sentences, or through an intermediate entity. This would be considered as a separate step after Relation Extraction. This simplifies the evaluation of the Relation Extraction task. Given a test corpus annotated with entities and relations, we run the relation extractor on the entity pairs within a sentence and collect the system output relations. We count correct, missing and spurious relations and compute the precision, recall and F1 measure.

In the pipeline of Information Extraction, entities are usually required in order to predict relations and events. Most research used hand-annotated entities (perfect entities) as input for Relation Extraction. Using entities tagged by an entity tagger could lead to a significant decrease in performance in practice. Some previous work directly tackled this problem by using joint learning between Named Entity Recognition and Relation Extraction (Li and Ji, 2014; Miwa and Bansal, 2016), while others dropped some constraints of entities(Nguyen and Grishman, 2014; Plank and Moschitti, 2013) (E.g., assuming knowing the entity boundaries but not the types). In this thesis, I will use perfect entities as input for simplicity.

There are also other types of Relation Extraction tasks. SemEval-2010 (Hendrickx et al., 2009) introduced a relation classification task for nominals. For example,

- *The cup contained <u>tea</u> from dried <u>ginseng</u>.*

This sentence contains a Entity-Origin relation which describes an entity as coming or being derived from an origin. This dataset is substantially different from other relation datasets. The entities in this dataset are common nouns and do not contain named entities. The number of negatives is artificially reduced, which makes it a pure classification task rather than detection and categorization. However, it became one of the popular datasets in the era of neural network and deep learning. Various neural models have been proposed and benchmarked on this dataset.

Another type of relation task is Distantly Supervised Relation Extraction (Hoffmann et al., 2011; Mintz et al., 2009; Riedel et al., 2010; Surdeanu et al., 2012). The task provides a knowledge base with a lot of entities and relations instead of an annotated corpus. The idea is to align the entities from the knowledge base to the sentences that contain those entities. Then we create a set of positive instances for which the relations exist in the knowledge base and a set of negative instances that do not exist in the knowledge base. The supervision of the data is done by the knowledge base instead of human annotation. The benefit of this approach is that it can create a large amount of training data, but the data can be quite noisy with both false positives and false negatives. A typical dataset of this task is the NYT10 dataset (Riedel et al., 2010), which aligns Freebase Bollacker et al., 2008 to the NYT corpus. It contains 52 relation types. Some examples are the following:

- */location/neighborhood/neighborhood_of*
- */people/person/place_of_birth*
- */business/company/place_founded*
- */people/person/profession*
- */people/person/nationality*
- */business/company/major_shareholders*

The evaluation would be slightly different, as the ground truth is the knowledge base instead of annotated corpus. Hand annotation of the test set may be required for more accurate assessment.

1.2 Traditional Methods: Feature-based and Kernel-based

To build a relation extractor, people started from hand-crafted patterns from word sequences to syntactic patterns. E.g.,

- $\underline{\textit{person}}$ *lives in* $\underline{\textit{location}}$

- $\underline{\textit{person}} \xleftarrow{subject} \textit{resides} \xrightarrow{prep\_in} \underline{\textit{location}}$

These patterns are easy to explain and can achieve high precision, but give limited coverage. It is hard to scale the set of patterns when the data grows larger. People quickly shifted to machine learning approaches when more annotated corpora became available.

Relation Extraction is then converted to a classification problem. The candidate set consists of all the entity pairs appearing in the same sentence. The pairs of entities that are annotated with relation types are the positives, while the rest are negatives. If there are n relation types, it will be an $(n + 1)$-way classification. Feature-based systems (Jing and Zhai, 2007; Kambhatla, 2004; Zhou et al., 2005) define a rich feature set for each entity pair and feed the data to a classifier (E.g., SVM (Cortes and Vapnik, 1995) or Maximum Entropy(Ratnaparkhi, 1999)). Zhou et al. (2005) used 60 types of features for the ACE 2004 dataset. E.g.,

- Bag of words of the heads of the entities

- The order of the two entities

- the number of words between the two entities

- The word that the entity depends on along the dependency path

- Entity types of the entities

Because the weights of features are learned from limited training data, it is not clear which features are more important than the others in general. Given a new dataset, we may have to do feature engineering to find the best feature set again. Jing and Zhai (2007) did a systematic analysis and showed that word sequence, constituency parse and dependency features can be comparably effective. The combination of these features can be slightly better.

While the process of enumerating features can be arbitrary, kernel methods (Bunescu and Mooney, 2005a,b; Zelenko et al., 2003; Zhao and Grishman, 2005; Zhou et al., 2007) seek to find a metric to directly measure the similarity between two relation candidates. Word sequence kernels and syntactic tree kernels are the two common categories of kernels for Relation Extraction. An effective kernel finds specific feature space to measure relations. E.g., The shortest dependency path kernel (Bunescu and Mooney, 2005b) only takes the span between the two entities in the dependency tree. The feature space is constrained to the Cartesian product of word, POS, entity type, and dependency direction along the dependency path (Figure 1.1). The kernel calculates the number of common paths between the entity pairs.

Overall, the kernel-based methods obtain comparable results to the feature-based counterparts, while some papers claim the kernel-based models are slightly better. The composition of different kernels is likely to achieve better results (Plank and Moschitti, 2013; Zhang et al., 2006) and can also incorporate feature-like information (Zhao and Grishman, 2005).

$$\begin{bmatrix}\text{protesters}\\ \text{NNS}\\ \text{Noun}\\ \textsc{Person}\end{bmatrix} \times [\rightarrow] \times \begin{bmatrix}\text{seized}\\ \text{VBD}\\ \text{Verb}\end{bmatrix} \times [\leftarrow] \times \begin{bmatrix}\text{stations}\\ \text{NNS}\\ \text{Noun}\\ \textsc{Facility}\end{bmatrix}$$

Figure 1.1: Shortest dependency path kernel from (Bunescu and Mooney, 2005a)

1.3 Neural Models

Neural Relation Extraction was introduced by Zeng et al. (2014), largely inspired by the success of neural models for text classification and sentiment analysis. Initially, the model only added the position embedding to the word embedding to indicate the positions of the two arguments (entities) in the sentence. Later, the entity type embedding was added (Nguyen and Grishman, 2015a) to adapt to the datasets where entity type information can be crucial features. A common neural Relation Extraction model contains an input layer, an encoder and a decoder.

The input layer contains the concatenation of the word embedding, position embedding and entity type embedding:

Word embedding: A certain type of pre-trained word embedding is often used such as Word2vec (Mikolov et al., 2013). The size of the embedding table is $|V| \cdot d_w$, where $|V|$ is the vocabulary size, and d_w is the embedding dimension.

Position embedding: We can use one vector to represent the position relative to one argument (entity). Thus, we have one embedding table (list of vectors) to represent all positions relative to one argument in a sentence. For each token, we look up its two position embeddings from the two position embedding tables (randomly initialized) with its relative distances to the two arguments, respectively.

The final embedding is the concatenation of the two. The size of one embedding table is $(2 \cdot l_s - 1) \cdot d_p$, where l_s is the sentence length, and d_p is the embedding dimension.

Entity type embedding: Similarly to word embedding, we can convert the entity type of a token to its embedding from the entity-type embedding table. Tokens outside the two entity spans will be randomly initialized to the same non-entity vector. Tokens within the two arguments will be converted to the vector of the argument's entity type. The size of the embedding table is $(|E|+1) \cdot d_e$, where $|E|$ is the number of entity types, and d_e is the embedding dimension.

Then the input layer is connected to the encoder which is often a CNN (LeCun et al., 1998; Nguyen and Grishman, 2015a; Zeng et al., 2014), or a Bidirectional RNN (Zhou et al., 2016), or their variations (Miwa and Bansal, 2016). The output of the encoder is often considered as feature representations for relations. This will be followed by a softmax classifier with or without one hidden layer (Figure 1.2).

This model, however, relies on the pre-trained word embeddings since the relation datasets are all too small to train on their own. The usage of pre-trained word embeddings for Relation Extraction had been studied earlier by (Nguyen and Grishman, 2014). While it helps in both feature-based and kernel-based methods as real-valued features or part of the metrics, it is more natural to use neural nets on top of the word embeddings. The choice of whether to fine-tune the word embeddings seems to depend on the dataset and the choice of the optimizer. More recently, Soares et al. (2019) showed that models initialized with the representations from pre-trained language models (BERT)(Devlin et al., 2018) can be even

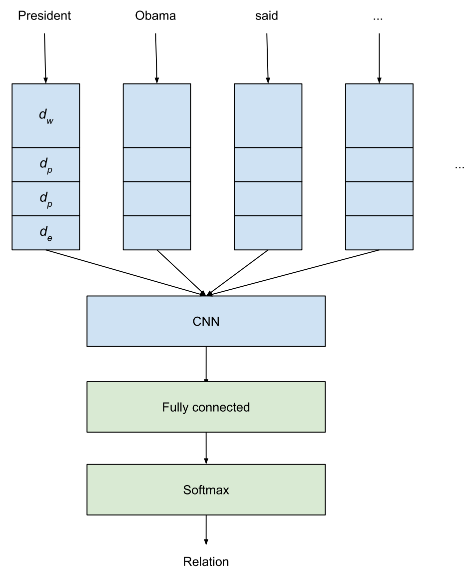

Figure 1.2: Neural Relation Extraction model.

better after fine-tuning on several benchmarks.

Most research on neural relation extraction has previously focused on variations of encoders (Dos Santos et al., 2015; Liu et al., 2015; Nguyen and Grishman, 2016; Socher et al., 2012; Xu et al., 2015) using the SemEval dataset (Hendrickx et al., 2009) as the benchmark, and recently more on developing different attention mechanisms (Cho et al., 2014) for the multi-instance representations on the distant supervision dataset (NYT10) (Du et al., 2018; Han et al., 2018b; Lin et al., 2016; Liu et al., 2018; Zeng et al., 2015). The general idea is basically to learn better feature representations for classification in supervised learning. While most of this work focuses on supervised learning on a single dataset, this thesis explores the possibility of utilizing multiple datasets at the same time.

We use multi-task learning to learn the original relation task along with the auxiliary task and related relation tasks. The multi-task paradigm has been used in many natural language processing tasks. In relation extraction, previous work has focused on the task relationship between named entity recognition and relation extraction. The proposed methods try to learn entities and relations at the same time to improve end-to-end relation extraction (Li and Ji, 2014; Miwa and Bansal, 2016). In this dissertation, we use annotated entities as input to relation extraction and focus on the relation task only. Another common multi-task scenario is the cross-lingual transfer learning by sharing parameters between models trained on different languages. Min et al. (2017) obtained improvement on bilingual relation extraction under the low-resource setting. In this thesis, we only evaluate our models on the English tasks and focus on the relationship between different datasets.

Chapter 2

Learning Domain-invariant Representations[1,2]

2.1 Introduction

The same type of relations might be expressed differently across diverse documents, topics and genres. We often observed that a relation extractor's performance degrades when applied to a domain other than the domain it is trained on. A simple method for domain adaptation (Blitzer et al., 2006; Daume, 2007; Jing and Zhai, 2007) is to construct a labeled dataset for the target domain, and then adjust a trained model with it. This is inefficient for relations - annotation is laborious to obtain, not to mention that relation mentions are sparse in the text. Take ACE 2004 as an example, *Personal/Social* relations appear only once

1. This chapter contains the work that has been published at IJCNLP 2017 (Fu et al., 2017).
2. I did the majority of the research and implementation for the chapter. The co-authors helped provide data, give high-level advice, and refine the paper.

on average per document. Such a method will not scale to the open-ended set of possible domains.

Among the features (Zhou et al., 2005) used for relation extraction, shortest dependency path can be applied cross-domain while argument-specific features (e.g., entity types, lexical forms) are likely to be more domain-specific. We hypothesize that it is possible to learn both domain-invariant and domain-specific representations with neural networks, and use the domain-invariant representation for new domains.

In this chapter, we propose to use a Domain Adversarial Neural Network (DANN) (Ajakan et al., 2014; Ganin and Lempitsky, 2015) to learn a domain-invariant representation for relations. Our contributions are twofold:

- We propose a novel domain adaptation approach for relation extraction that learns cross-domain features by itself and that requires no label in targets.
- Experiments on the ACE domains show that our approach improves on the state-of-the-art across all domains.

In the following sections, we will first briefly summarize related work, then describe the model (Section 2.3). We will present experimental results (Section 2.4) and conclusion at the end.

2.2 Related Work

There has been a lot of research on domain adaptation in natural language processing (Ajakan et al., 2014; Blitzer et al., 2006; Daume, 2007; Ganin and

Lempitsky, 2015; Glorot et al., 2011; Jing and Zhai, 2007). Most of the existing domain adaptation methods are based on discrete feature representations and linear classifiers. There is also recent work on domain adaptation for relation extraction including feature-based systems (Nguyen et al., 2014; Nguyen and Grishman, 2014) and kernel-based system (Plank and Moschitti, 2013). Nguyen and Grishman (2014) and Nguyen et al. (2014) both require a few labels in the target domain. Our proposed method can perform domain adaptation without target labels.

Some other methods also do not have such requirement. Plank and Moschitti (2013) designed the semantic syntactic tree kernel (SSTK) to learn cross-domain patterns. Nguyen et al. (2015b) constructed a case study comparing feature-based methods and kernel-based models. They presented some effective features and kernels (e.g. word embedding).We share the same intuition of finding those cross-domain features, but our work differs from such previous work in that they manually designed those features and kernels while we automatically learn cross-domain features from unlabeled target-domain examples with neural networks. To our best knowledge, this is the first work on neural networks for domain adaptation of relation extraction.

2.3 Model

We formulate the relation extraction task as a classification problem over all entity pairs (relation candidates) in a sentence. The overall structure of the model is shown in Figure 2.1. The model will first convert a relation candidate into a

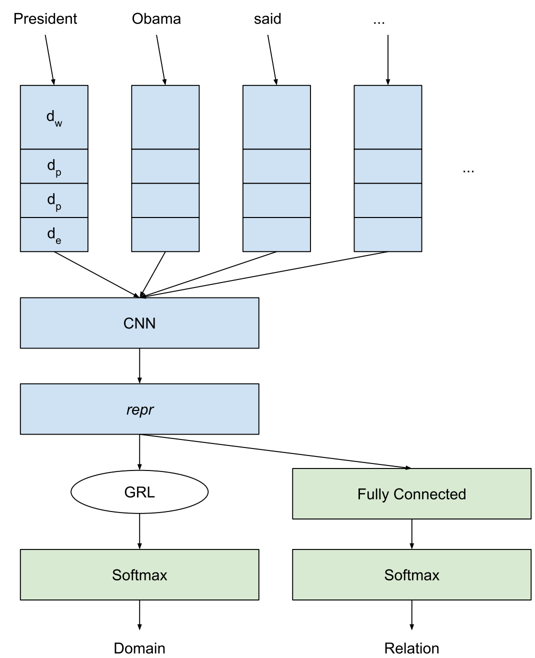

Figure 2.1: Model architecture

fixed-length matrix, then uses a single-layer Convolutional Neural Network (CNN) with dropout to learn its hidden representation *repr*. On top of *repr*, it uses two decoders: a fully-connected layer with dropout for predicting the relation type (Zeng et al., 2014) (Section 2.3.1), and another decoder with domain adversarial neural network(Ganin and Lempitsky, 2015) to predict its domain. The additional domain-adversarial decoder is used to enforce the domain-invariance of the feature layer (Section 2.3.2).

2.3.1 CNN-based Encoder-Decoder Model for Relations

Each sentence is truncated or padded to a fixed length (l_s) of tokens. Each token of the text is then represented as the concatenation of several types of embeddings. We follow the previous work as introduced in Chapter 1.3 to use d_w-dimension word embedding, two d_p-dimension position embeddings and d_e-dimension entity type embedding. To compare to the state-of-the-art results, we also add the chunk embedding and the on_dep_path embedding from (Nguyen and Grishman, 2016):

Chunk embedding: Similar to entity type, we have chunk embedding according to each token's chunk type. The size of embedding table is $(|C|+1) * d_c$, where $|C|$ is the number of chunk types, and d_c is the embedding dimension.

On dep path embedding: For each token, we have a vector to indicate whether the token is on the dependency path between the two entities. We have two vectors in total. The vector size is d_d.

The input layer is a matrix with size $(d_w + 2 \cdot d_p + d_e + d_c + d_d) \cdot l_s$. A standard convolution layer with variable window sizes (feature maps) is applied on this, following by max-pooling and dropout. Each filter with the same window size has the same filter size. The output is the feature representation layer ($repr$) of size $d_f \cdot |W|$, where d_f is the filter size, and $|W|$ is the size of the set of window sizes. We add fully connected layers to this feature representation with softmax to predict the relation type. The model is similar to that in (Nguyen and Grishman, 2016).

2.3.2 Domain Adversarial Neural Network

How does domain adaptation work without any labeled examples for the target domain? Following Ganin and Lempitsky (2015) and Ajakan et al. (2014), we use DANN to learn a representation that is more general across domains and eliminating source-only distinctive features that are easily learned with labeled source data.

It learns domain invariant features by jointly optimizing the underlying feature layer from the main learning task and the domain label predictor. In this case, the main learning task is the relation type prediction in Section 2.3.1. The domain label predictor is a binary classifier that discriminates whether the example is from source or target. The domain classifier consists of the gradient reversal layer (GRL) and a few fully connected layers. The GRL is defined as an identity function with reversed gradient. For input layer x:

$$GRL(x) = x, \frac{d}{dx}GRL(x) = -I,$$

where I is the identity matrix.

We use a binary cross-entropy loss for the domain classifier:

$$L_{domain} = \sum_{i=0}^{N_s+N_t}\{d_i log(\hat{d_i}) + (1-d_i)log(1-\hat{d_i})\},$$

where $d_i \epsilon \{0, 1\}$ is the domain label $\{source, target\}$, and N_s, N_t stand for the number of examples in source and target.

The loss of the whole model is the linear combination of the task loss and the domain loss:

$$L = L_{relation} + \lambda \cdot L_{domain},$$

where λ is the adaptation weight, and $L_{relation}$ is the loss of the relation classifier.

During the training, half of the examples comes from the source and half from the target in a single batch. Only examples from the source have relation labels, while both source and target examples have domain labels. As a result, the source part is used to calculate the relation loss $L_{relation}$. The whole batch is used to calculate the domain loss L_{domain}.

We choose the feature representation layer (*repr*) from the relation model (Section 2.3.1) as the input to GRL. During the training, while the parameters of the relation and domain decoders are both optimized to *minimize* their errors, the parameters of *repr* are optimized to *minimize* the loss of the relation decoder and to *maximize* (due to GRL) the loss of the domain classifier. The latter encourages domain-invariant features to emerge for domain adaptation.

In feature-based models, lexicon-level features are often domain-specific such as a person's name. e.g. word-level features that contain Obama and US can be indicators for an employment relation. It is true in many news articles, but not in general. Instead of manually deciding whether to use the feature or not, we can use DANN to read the target domain text to make the decision depending on the domain.

2.4 Experiements

2.4.1 Dataset

We use the ACE 2005 dataset to evaluate domain adaptation by dividing its articles from its six genres into respective domains: broadcast conversation (*bc*), broadcast news (*bn*), telephone conversation (*cts*), newswire (*nw*), usenet (*un*) and weblogs (*w*l). Previous work (Gormley et al., 2015; Nguyen and Grishman, 2016) uses newswire (*bn* & *nw*) as the training set, half of *bc* as the development set, the other half of *bc*, *cts* and *wl* as the test sets. We use the same data split. Our model requires unlabeled target domain instances. To meet this requirement and avoid train-on-test, we also split *cts* and *wl* when adapting to them. For all three test domains, we use half of the dataset as the development set, and the other half as the test set (Table 2.1). †The **bc** split is the same as several previous work. We use the same training set and the same preprocessing as the previous work. This results in 43,497 entity pairs for training. We also use the same label set which is expanded by creating two relation types for each asymmetric relation.

Split	**bc†**	**wl**	**cts**
train	nw & bn	nw & bn	nw & bn
dev	half of bc	half of wl	half of cts
test	half of bc	half of wl	half of cts

Table 2.1: Data split for the experiments.

2.4.2 Configuration and Hyperparameters

We use word embedding pre-trained on newswire with 300 dimensions from word2vec (Mikolov et al., 2013). We fix the word embeddings during the training because tuning did not show improvement. We follow Nguyen and Grishman (2016) to set the hyperparameters for CNN: the embedding sizes (Section 2.3.1) $d_e, d_p, d_d, d_c, d_d, = 50$, the max sentence length $l_s = 50$, the set of filter window sizes $W = 2, 3, 4, 5$, the number of filters for each window size $d_f = 150$, and the dropout rate to be 0.5. We use one fully connected layer with 300 dimensions for the relation decoder before the softmax layer. We only use a softmax layer for domain decoder. The learning rate is 0.001. We halve the learning rate every two epochs. We use Adam as the optimization method. The adaptation weight is tuned to be 0.1 using the dev set. For all scores, we run experiments 10 times and take the average.

2.4.3 Evaluation

Method	**bc**	**wl**	**cts**	**avg**
(Gormley et al., 2015)	61.90	50.36	52.93	55.06
(Nguyen and Grishman, 2016)	63.26	**53.91**	**55.63**	57.60
CNN	64.44	53.34	55.06	57.61
CNN + DANN	**65.16**	53.59	55.43	**58.06**

Table 2.2: Adaptation to the *bc* domain.

Our baseline CNN model achieved comparable performance to the state-of-the-art relation extraction methods (Table 2.2). This Table is using the data split following previous methods (not Table 2.1). F1 scores are reported on test sets with the same splits as previous work. Compared to (Gormley et al., 2015; Nguyen and Grishman, 2016), our baseline model already obtained higher score on *bc*. They also reported higher scores by ensemble with other models (feature-based or multiple neural net models) which is unfair to compare for a single model. Essentially, our model can also serve as one of the base models in the ensemble.

We trained DANN to read the development set of *bc* to adapt to this domain. Although the gain seems to be small, the improvement is statistically significant ($p\text{-}value = 0.00289$ between CNN and CNN+DANN on *bc* domain). We ran an instance-based sign test on the combination of the output of 10 experiments. We have 10 observations of each instance in the original dataset. We treat them as independent examples when calculating the significance. While DANN improves *bc* significantly, it does not help the other two domains when adapting to this domain. We also want to find out how it works on other domains. In the original split used by previous work, *wl* and *cts* do not have dev and test split. We, therefore, created the data split by ourselves (Table 2.1) and compare the results to our own baseline model . We observe similar improvement on *wl*, but not on *cts* (Table 2.3). This Table is using the data split in Table 2.1. By doing some feature engineering on the embedding layer, we found that the Chunk embedding and On dep path embedding improves the *cts* a lot. The model obtains 52.96 (without) and 57.02 (with) these embedding. With DANN, it obtains 53.74 (+0.78) and 57.19 (+0.17). The effective hand-designed cross-domain features from the embedding layer could make the room for improvement smaller.

Given a group of documents, our setting is to let the DANN read more unlabeled documents from the same domain and train the relation model along with it. Then, we obtain a better model for this domain. This also means that we will have to train different models for different domains. Ideally, we would like to have a model that can work on all domains at the same time. To test this, we try to adapt to the three domains in the dataset at the same time. Under this setting, DANN reads unlabeled data from all three domains along with the supervised relation model. As the result (Table 2.3), the model tends to learn something in between. It performs better on *bc* and *wl*, but worse on *cts*. It is not very surprising since DANN will force the representation layer to be domain-invariant. To really lift the performance of all the domains with a single model, the model needs to capture some domain-specific representation as well. This would be hard to achieve without labels from the target domains, but still an interesting direction to investigate. Under the current situation, it would be better to train separate models that are adapted to each domain.

Method	**bc**	**wl**	**cts**	**avg**
CNN	64.33	54.58	57.02	58.64
+ DANN (all)	64.94	55.17	56.08	58.73
+ DANN (each)	**65.16**	**55.55**	**57.19**	**59.30**

Table 2.3: F1 scores on adaptation to all three domains at the same time and adaptation to each domain individually.

2.5 Conclusion

Our model successfully obtains improvement on all three test domains of relations at ACE 2005. It uses a domain adversarial neural network to learn cross-domain features. It does not require hand-crafted features for domain adaptation. It can be a useful tool for relation extraction since labeled data is always hard to acquire. This work has been extended by (Shi et al., 2018). They add autoencoders to the model to further improve the domain adaptation capability.

Chapter 3

Learning a Unified Encoder[1,2]

3.1 Introduction

Several relation schemas and annotated corpora have been developed such as the Automatic Content Extraction (ACE), and the Entities, Relations and Events (ERE) annotation (Song et al., 2015). These schemas share some similarity, but differ in details (Aguilar et al., 2014). A relation type may exist in one schema but not in another. An example might be annotated as different types in different datasets. For example, Part-whole.Geographical relations in ACE05 are annotated as Physcial.Located relations in ERE. Most of these corpora are relatively small. Models trained on a single corpus may be biased or overfitted towards the corpus.

Despite the difference in relation schemas, we hypothesize that we can learn a

1. This chapter contains the work that has been published at the 4th Workshop on Noisy User-generated Text (W-NUT) at EMNLP 2018 (Fu et al., 2018).

2. I did the majority of the research and implementation for the chapter. The co-authors helped provide data, give high-level advice, and refine the paper.

more general representation with a unified encoder. Such a representation could have better out-of-domain or low-resource performance. We develop a multi-task model to learn a representation of relations in a shared relation encoder. We use separate decoders to allow different relation schemas. The shared encoder accesses more data, learning less overfitted representation. We then regularize the representation with adversarial training in order to further enforce the sharing between different datasets. In our experiments, we take ACE05 [3] and ERE [4] datasets as a case study. Experimental results show that the model achieves higher performance on both datasets.

3.2 Related Work

Relation extraction is typically reduced to a classification problem. A supervised machine learning model is designed and trained on a single dataset to predict the relation type of pairs of entities. Traditional methods rely on linguistic or semantic features (Jing and Zhai, 2007; Zhou et al., 2005), or kernels based on syntax or sequences (Bunescu and Mooney, 2005a,b; Plank and Moschitti, 2013) to represent sentences of relations. More recently, deep neural nets start to show promising results. Most rely on convolutional neural nets (Fu et al., 2017; Nguyen and Grishman, 2015a, 2016; Zeng et al., 2014, 2015) or recurrent neural nets (Miwa and Bansal, 2016; Zhang et al., 2015; Zhou et al., 2016). Our supervised base model will be similar to (Zhou et al., 2016). Our initial experiments did not use syntac-

3. urlhttps://catalog.ldc.upenn.edu/LDC2006T06

4. We use 6 LDC releases combined: LDC2015E29, LDC2015E68, LDC2015E78, LDC2015R26, LDC2016E31, LDC2016E73

tic features (Fu et al., 2017; Nguyen and Grishman, 2016) that require additional parsers.

In order to further improve the representation learning for relation extraction, Min et al. (2017) tried to transfer knowledge through bilingual representation. They used their multi-task model to train on the bilingual ACE05 datasets and obtained improvement when there is less training available (10%-50% of the whole training set). Our experiments will show our multi-task model can make significant improvement on the full training set.

In terms of the regularization to the representation, Duong et al. (2015) used l2 regularization between the parameters of the same part of two models in multi-task learning. Their method is a kind of soft-parameter sharing, which does not involve sharing any part of the model directly. In the previous chapter, we applied domain adversarial networks (Ganin and Lempitsky, 2015) to relation extraction and obtained improvement on out-of-domain evaluation. In this chapter, we attempt to use it as a regularization tool in a different context and find some improvement.

3.3 Supervised Neural Relation Extraction Model

The supervised neural model on a single dataset was introduced by Zeng et al. (2014) and followed by many others (Fu et al., 2017; Miwa and Bansal, 2016; Nguyen and Grishman, 2015a, 2016; Zhou et al., 2016). We use a similar model as our base model. It takes word tokens, position of arguments and their entity types as input. Some work (Fu et al., 2017; Nguyen and Grishman, 2016) used extra syntax features as input. However, the parsers that produce syntax features could have errors and vary depending on the domain of text. The syntax features learned could also be too specific for a single dataset. Thus, we focus on learning representation from scratch, but also compare the models with extra features later in the experiments. The encoder is a bidirectional RNN with attention and the decoder is one hidden fully connected layer followed by a softmax output layer.

In the input layer, we convert word tokens into word embeddings with pre-trained word2vec (Mikolov et al., 2013). For each token, we convert the distance to the two arguments of the example to two position embeddings. We also convert the entity types of the arguments to entity embeddings. The setup of word embedding and position embedding was introduced by Zeng et al. (2014). The entity embedding (Fu et al., 2017; Nguyen and Grishman, 2016) is included for arguments that are entities rather than common nouns. At the end, each token is converted to an embedding w_i as the concatenation of these three types of embeddings, where $i \in [0, T)$, T is the length of the sentence.

A wide range of encoders have been proposed for relation extraction. Most of them fall into categories of CNN (Zeng et al., 2014), RNN (Zhou et al., 2016) and TreeRNN (Miwa and Bansal, 2016). In this work, we follow Zhou et al. (2016) to use Bidirectional RNN with attention (BiRNN), which works well on both of the datasets we are going to evaluate on. BiRNN reads embeddings of the words from both directions in the sentence. It summarizes the contextual information at each state. The attention mechanism aggregates all the states of the sentence by paying more attention to informative words. Given input w_i from the input layer, the encoder is defined as the following:

$$\overrightarrow{h_i} = \overrightarrow{GRU}(w_i, \overrightarrow{h_{i-1}}), \tag{3.1}$$

$$\overleftarrow{h_i} = \overleftarrow{GRU}(w_i, \overleftarrow{h_{i+1}}), \tag{3.2}$$

$$h_i = concatenate(\overrightarrow{h_i}, \overleftarrow{h_i}) \tag{3.3}$$

$$v_i = tanh(W_v h_i + b_v), \tag{3.4}$$

$$\alpha_i = \frac{exp(v_i^\top v_w)}{\sum_t exp(v_t^\top v_w))}, \tag{3.5}$$

$$\phi(x) = \sum_i \alpha_i h_i. \tag{3.6}$$

We use GRU (Cho et al., 2014) as the RNN cell. W_v and b_v are the weights for the projection v_i. v_w is the word context vector, which works as a query for selecting important words. The importance of the word is computed as the similarity between v_i and v_w. The importance weight is then normalized through a softmax function. Then we obtain the high level summarization $\phi(x)$ for the relation example.

The decoder uses this high level representation as features for relation classification. It usually contains one hidden layer (Fu et al., 2017; Nguyen and Grishman, 2016; Zeng et al., 2014) and a softmax output layer. We use the same structure which can be formalized as the following:

$$h = ReLU(W_h \phi(x) + b_h), \tag{3.7}$$

$$p = softmax(W_o h + b_o), \tag{3.8}$$

where W_h and b_h are the weights for the hidden layer, W_o and b_o are the weights for the output layer. We use cross-entropy as the training loss.

3.4 Learning Unified Representation

While the data for one relation task may be small, noisy and biased, we can learn a better representation combining multiple relation tasks. We attempt to use multi-task learning to learn a unified representation across different relation tasks. The method is simple and straightforward. We use the same encoder to learn the unified feature representation for both relation tasks, and then we train classifiers for each task on top of this representation. We then apply regularization on this representation by adversarial training.

3.4.1 Multi-task Learning

Given example x_1 from relation schema 1 and x_2 from relation schema 2, we use the same encoder to obtain representation $\phi(x_1)$ and $\phi(x_2)$ respectively. Then we build separate decoders for them using the same structure (3.7) (3.8). To train them at the same time, we put examples from both tasks in the same batch. The ratio of the examples are controlled so that the model reads both datasets once every epoch. We use linear interpolation to combine the loss from them.

$$L = (1 - \lambda)L_1 + \lambda L_2, \tag{3.9}$$

where λ is used to control the attention to each task. The model may learn the two tasks at different speed. During optimization, one task can be seen as the

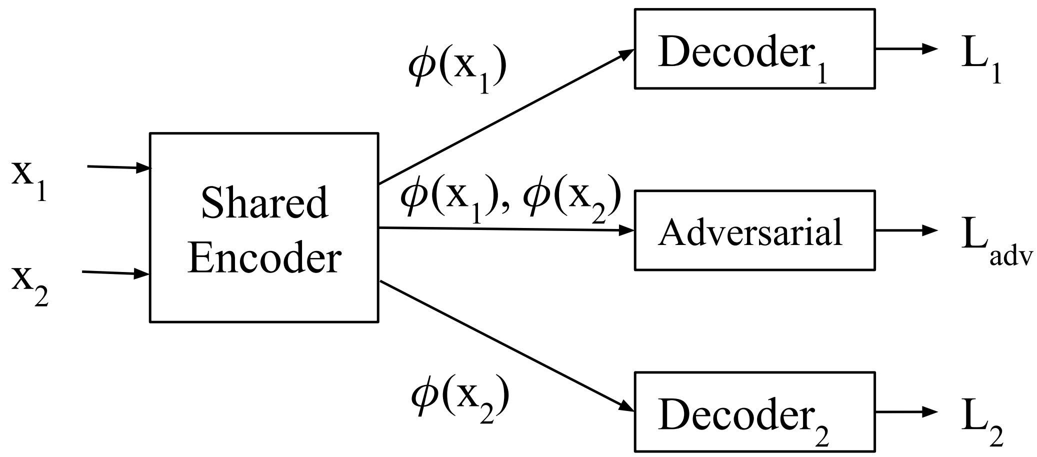

Figure 3.1: Multi-task model with regularization

main task, while the other can be seen as the auxiliary task. The benefit of joint learning to the main task may vary depending on how much attention the model pays to the auxiliary task.

3.4.2 Regularization by Adversarial Training

Being optimized simultaneously by different decoders, the model could still learn very different representation for similar examples coming from different tasks. We want to prevent this and to further push the model to learn similar representation for similar examples even if they come from different tasks. We attempt to regularize the representation using adversarial training between the two tasks.

Given the representation $\phi(x_1)$ and $\phi(x_2)$ learned from the two tasks, we build a classifier to predict which task the examples come from (3.11). We add a gradient reversal layer (Ganin and Lempitsky, 2015) as introduced in Section 2.3.2 at the input of this classifier (3.10) to implement the adversarial training.

$$\phi(x) = GRL(\phi(x)), \tag{3.10}$$

$$p = softmax(W\phi(x) + b). \tag{3.11}$$

While the classifier learns to distinguish the sources of the input representation, the input representation is learned in the opposite direction to confuse the classifier thanks to GRL. Thus, the input representation ($\phi(x_1)$ and $\phi(x_2)$) will be pushed to be close to each other.

We also use the cross-entropy loss for this adversarial training, and combine the loss L_{adv} with the two relation tasks.

$$L = (1 - \lambda)L_1 + \lambda L_2 + \beta L_{adv}, \tag{3.12}$$

where we can use β to control how close the representations are between the two relation tasks.

3.5 Experiments

3.5.1 Datasets

To apply the multi-task learning, we need at least two datasets. We pick ACE05 and ERE for our case study. The ACE05 dataset provides a cross-domain evaluation setting . It contains 6 domains: broadcast conversation (bc), broadcast news (bn), telephone conversation (cts), newswire (nw), usenet (un) and weblogs (wl). Previous work (Fu et al., 2017; Gormley et al., 2015; Nguyen and Grishman, 2016) used newswire as training set (bn & nw), half of bc as the development set, and the other half of bc, cts and wl as the test sets. We followed their split of documents and their split of the relation types for asymmetric relations. The ERE dataset has a similar relation schema to ACE05, but is different in some annotation guidelines (Aguilar et al., 2014). It also has more data than ACE05, which we expect to be helpful in the multi-task learning. It contains documents from newswire and discussion forums. We did not find an existing split of this dataset, so we randomly split the documents into train (80%), dev (10%) and test (10%).

3.5.2 Model Configurations

We use word embedding pre-trained on newswire with 300 dimensions from Word2vec (Mikolov et al., 2013). We fix the word embeddings during the training. We follow Nguyen and Grishman (2016) to set the position and entity type embedding size to be 50. We use 150 dimensions for the GRU state, 100 dimensions for the word context vector and use 300 dimensions for the hidden layer in the decoders. We train the model using Adam (Kingma and Ba, 2014) optimizer with learning rate 0.001. We tune λ linearly from 0 to 1, and β logarithmically from $5 \cdot 10^{-1}$ to 10^{-4} For all scores, we run experiments 10 times and take the average.

3.5.3 Augmentation between ACE05 and ERE

Training separately on the two corpora (row “Supervised” in Table 3.1), we obtain results on ACE05 comparable to previous work (Gormley et al., 2015) with substantially fewer features. With syntactic features as (Fu et al., 2017; Nguyen and Grishman, 2016) did, it could be further improved. In this section, however, we want to focus on representation learning from scratch first. Our experiments focus on whether we can improve the representation with more sources of data.

A common way to do so is pre-training. As a baseline, we pre-train the encoder of the supervised model on ERE and then fine-tune on ACE05, and vice versa (row “Pretraining” in Table 3.1). We observe improvement on both fine-tuned datasets. This shows the similarity between the encoders of the two datasets. However, if we fix the encoder trained from one dataset, and only train the decoder on the other dataset, we will actually obtain a much worse model. This indicates that neither dataset contains enough data to learn a universal feature representation layer for classification. This leaves the possibility to further improve the representation by learning a better encoder.

We then attempt to learn a multi-task model using a shared encoder. We use 14K words as the vocabulary from ACE05 and 20K from ERE. There are about 8K words shared by the two datasets (same for both pretrained and multi-task models). By multi-task learning, we expect the model to conceive the embeddings for words better and construct more general representation. Experiments determined that the multi-task learning works best at $\lambda = 0.8$ for both ACE05 and ERE datasets (Table 3.1). It obtains improvement on both the out-of-domain evaluation on ACE

	ACE05				ERE
Method	**bc**	**wl**	**cts**	**avg**	**test**
Supervised	61.44	52.40	52.38	55.40	55.78
Pretraining	60.21	53.34	56.10	56.55	56.39
Multi-task	61.67	55.03	**56.47**	57.72	57.29
+ Regularization	**62.24**	**55.30**	56.27	**57.94**	**57.75**

Table 3.1: Multi-task Learning and Regularization (100% training data).

and in-domain evaluation on ERE. It works especially well on weblogs (wl) and telephone conversation (cts) domains on ACE, which possibly benefits from the discussion forum data from ERE.

On the other hand, we use the adversarial training between the two datasets to further enforce the representation to be close to each other. There is strong dependency between the schemas of these two datasets. Two examples from different datasets could have the same semantics in terms of relation type. We try to force the representation of these examples to be similar. With appropriate amount of this regularization ($\beta = 0.001$), the model can be further improved (Table 3.1). The amount of improvement is modest compared to sharing the encoder. This may show that the multi-task model can already balance representation with enough labels on both sides. We also artificially remove half of the training data of each dataset to see the performance in a relatively low-resource setting (Table 3.2). We observe larger improvement with both multi-task learning and regularization. Because of the decrease of the training data, the best λ is 0.9 for ACE05 and 0.7 for ERE. We also use slightly stronger regularization ($\beta = 0.01$).

	ACE05				ERE
Method	**bc**	**wl**	**cts**	**avg**	**test**
Supervised	56.03	47.81	48.65	50.83	53.60
Pretraining	55.39	49.17	52.91	52.49	54.66
Multi-task	57.39	51.44	54.28	54.37	55.72
+ Regularization	**57.73**	**52.30**	**54.63**	**54.89**	**55.91**

Table 3.2: Multi-task Learning and Regularization (50% training data).

3.5.4 More Features on ACE05

Since ACE05 has been studied for a long time, numerous features have been found to be effective on this dataset. (Nguyen and Grishman, 2016) incorporated some of those features into the neural net and beat the state-of-art on the dataset. Although representation learning from scratch could be more general across multiple datasets, we compare the effect of multi-task learning with extra features on this specific dataset.

We add *chunk embedding* and *on_dep_path embedding* (Nguyen and Grishman, 2016). Similar to entity type embedding, *chunk embedding* is created according to each token's chunk type, we set the embedding size to 50. *On_dep_path embedding* is a vector indicating whether the token is on the dependency path between the two entities. In the multi-task model, the shared encoder is a bidirectional RNN (BiRNN) without attention (Equation (3.1-3.3)). These two embeddings will be concatenated to the output of the BiRNN to obtain the new h_i and then passed to Equation (3.4).

As to the result (Table 3.3), our supervised baseline is slightly worse than the previous state-of-the-art neural model with extra features, but the multi-task

Method	**bc**	**wl**	**cts**	**avg**
(Nguyen and Grishman, 2016)	63.07	**56.47**	53.65	57.73
Supervised	61.82	55.68	55.15	57.55
Multi-task	**63.59**	56.11	**56.78**	**58.83**

Table 3.3: Multi-task Learning with extra features on ACE05 (100% training data).

Method	**bc**	**wl**	**cts**	**avg**
Supervised	56.81	50.49	50.10	52.47
Multi-task	**58.24**	**52.90**	**53.09**	**54.37**

Table 3.4: Multi-task Learning with extra features on ACE05 (50% training data).

learning can consistently help. The improvement is more obvious with 50% training data (Table 3.4). It is also worth noting that with 50% training data, the extra features improve the supervised base model, but not the multi-task learning model. It shows the effectiveness of the multi-task model when there is less training data.

3.6 Conclusion and Future Work

We attempt to learn unified representation for relations by multi-task learning between ACE05 and ERE datasets. We use a shared encoder to learn the unified feature representation and then apply regularization by adversarial training. The improvement on both datasets shows the promising future of learning representation for relations in this unified way. This will require less training data for new relation schemas. It will be interesting future work to further explore the multi-task learning between different datasets, especially to capture the dependency between different schemas in the decoder.

Chapter 4

Learning Relatedness between Types with Prototypes[1]

4.1 Introduction

In this chapter, we focus on the relationships between relation types across different relation schemas. These relation schemas are mostly pre-defined in existing datasets. The definition of the relation type depends on the annotation guide. There is no clear intrinsic Ontology for relation types. In practice, relation types can be created based on interests. This leaves datasets with similar, related or overlapping schemas. For example, the annotation guidelines for Automatic Content Extraction (ACE) 03-05 changed from year to year. The later created Entities, Relations and Events (ERE) dataset was similar in the schema, but differs

1. I did the majority of the research and implementation for the chapter. The co-authors helped provide data, give high-level advice, and refine the paper.

in details. Because of the difficulty of annotating relations, these datasets are all small individually and hard to be utilized together.

However, we can observe the connections between the relation types from different datasets based on relation names or annotation guidelines. This is the prior knowledge that one relation type in one dataset may have closer relationships to some types than the others in another dataset. We design a model to recognize this similarity. We propose to use prototypical examples to represent each relation type. We rank these representations higher for related types, and lower for unrelated types using a pair-wise loss function. Our base model is a multi-task learning model which focuses on learning a strong encoder using multiple datasets regardless of the relation schemas. We take a step further to explore utilizing the relatedness between the relation types. Experiments on ACE05 and ERE show that it can further boost the performance, especially in the low-resource settings.

4.2 Related Work

Relation type dependency: There have been a few ways to model the relationships between types in a multi-label relation dataset where we can learn the similarity or dependency from annotated examples. Surdeanu et al. (2012) used a two-layer hierarchical model. The object-level classifier is able to capture the label dependency, while the mention-level classifier is focused on multi-label classification. Riedel et al. (2013) used a neighborhood model to explicitly model the dependency between the labels in a matrix factorization framework. Both models are designed to work on multi-label examples, which require annotation to capture the dependency between labels. Among the recent work on neural methods for relation extraction, most (Lin et al., 2016; Liu et al., 2017; Zeng et al., 2015) ignores the multi-label setting and does not explicitly model the label dependency. Ye et al. (2017), on the other hand, ranks the similarity between feature representation of the instance and the label embedding. In addition to ranking the positive classes higher than the negative ones, it ranks positive classes against each other to learn the connections between the positive classes. These methods all require annotated examples to learn the connections. In the case of relation types across different datasets, such annotations do not exist. We attempt to learn the similarity nevertheless using prototypes from each type.

Multi-task learning: Training multiple relation datasets at the same time could improve the robustness of the model and reduce annotation cost for relation extraction. In chapter 3, we introduced a shared encoder to learn more general feature representation. In this chapter, we will use a similar multi-task learning base model and incorporate the similarity between the relation schemas to further improve the performance.

4.3 Relation Model with Multi-task Learning

The majority of neural relation models (Lin et al., 2016; Nguyen and Grishman, 2015a; Zeng et al., 2014, 2015) encode a sentence using a deep architecture to a vector representation followed by a softmax classifier, while the others (Dos Santos et al., 2015; Ye et al., 2017) use a function to compute the score between label embedding and sentence representation. In the previous chapter, we have showed that the shared encoder can help in the case of the multi-task learning. Inspired by this, we choose the latter so that all relation types (including from different datasets) will share all model parameters except the label embeddings. Suppose we obtain the sentence representation $\phi(x)$ with a neural architecture. We define the label embedding as $W_l \in \mathbb{R}^D$, a D-dimension vector for each type. We compute the L_1 distance between them and learn a scoring function to estimate the scores $S_\theta(x)$ for every type:

$$S_\theta(x)_l = W_o \cdot |\phi(x) - W_l| + b_o, \tag{4.1}$$

where $W_o \in \mathbb{R}^D$ and $b_o \in \mathbb{R}$ are shared for all types.

We do not use the dot product (Dos Santos et al., 2015; Ye et al., 2017) as the scoring function because the L_1 distance works slightly better in the multi-task learning experiments. The probability of every class is computed as the softmax output of the scores. Similar to (Fu et al., 2018), we jointly train two relation tasks at the same time with cross-entropy losses.

$$L = \lambda L_{r1} + (1 - \lambda) L_{r2}, \tag{4.2}$$

where L_{r1} and L_{r2} are the cross-entropy losses for the two relation tasks. λ is the hyperparameter to control the learning speed between the two tasks. This would give a strong baseline of utilizing the two datasets together.

4.3.1 Prototypes of Relation Types for Learning Similarity

For each relation type, we randomly select k examples (S_k) from the training set as supporting examples. We use the mean of the representations of these examples as the prototype for the relation type:

$$\bar{x}_c = \frac{1}{k} \sum_{x_i \in S_k} \phi(x_i) \tag{4.3}$$

These prototypes are inspired by the Prototypical Networks (Snell et al., 2017). However, in the training procedure, these supporting examples are randomly selected for every mini-batch. We have dynamic prototypes during training.

We define $S_\theta(\bar{x}_c)_l$ as the similarity score to type c for type l. We hypothesize that if the two relation types are similar in semantics, they should obtain high similarity score. Within the dataset, if the relation types in the schema are mutually exclusive, then we would expect a high similarity score to itself and low scores to the other types. Across the datasets, the prototypes would obtain high scores for related types and low scores for unrelated types. We use a pair-wise ranking loss (Dos Santos et al., 2015) to learn this relatedness across the datasets.

For $l \in L$ and $c \in C$, $S_\theta(\bar{x}_c)_l$ gives the score for the similarity between the type l in the relation schema L and the type c in the relation schema C. Let $c^+ \in C$ be a class related to l and $c^- \in C$ be a class unrelated to l. The similarity scores would be $S_\theta(\bar{x}_{c^+})_l$ and $S_\theta(\bar{x}_{c^-})_l$ respectively. We define the pair-wise ranking loss as:

$$\begin{aligned} L_s = log(1 + exp(\gamma(m^+ - S_\theta(\bar{x}_{c^+})_l)) \\ +log(1 + exp(\gamma(m^- + S_\theta(\bar{x}_{c^-})_l)) \end{aligned} \tag{4.4}$$

m^+ and m^- are the margins and γ is the scaling factor. This loss function would push $S_\theta(\bar{x}_{c^+})_l$ higher for related type pair between c^+ and l and $S_\theta(\bar{x}_{c^-})_l$ lower for unrelated type pair between c^- and l. We manually create a relatedness matrix to state whether the two types are related or not between the types in C and L based on the definition of the relation types. For each step of training, we pick the highest scored c^- from unrelated types and lowest scored c^+ for related types corresponding to type l.

$$c^- = \underset{c \in C^-}{argmax} S_\theta(\bar{x}_c)_l \tag{4.5}$$

$$c^+ = \underset{c \in C^+}{argmin} S_\theta(\bar{x}_c)_l \tag{4.6}$$

where C^- are types unrelated to l and C^+ are types related to l. In experiments, we use looser margins ($m^+ = 0.5$, $m^- = 0.5$, $\gamma = 1.0$) compared to (Dos Santos et al., 2015) as we are learning the relatedness between types rather than doing classification for individual instances. The ranking loss is jointly trained as an auxiliary task with the main relation tasks:

$$L = \lambda L_{r1} + (1 - \lambda) L_{r2} + \beta L_s, \tag{4.7}$$

where we use β to control the weight for learning the relatedness. If β is too large, it could disrupt the learning of main relation tasks. With appropriate weight, it could help augment the label embeddings for the relation types by considering the similarity between them.

4.4 Experiments

4.4.1 Datasets

We select the same two datasets in the experiments as we did in the previous chapter. There is overlapping of relation types between ACE05 and ERE, but the annotation guidelines are different in details for types that have the same name (Aguilar et al., 2014). Thus, we can not combine the datasets directly as a single dataset. We have shown that the multi-task learning with the two datasets at the same time could obtain better feature representation in the previous chapter. We take a step further and try to learn the similarity between the types. Most of the types in one dataset can be mapped (similar) to the other as a one-to-one mapping, while the *artifact* type does not exist in ERE. We manually reviewed the relatedness between the relation types in the schemas as the labels (related or unrelated) for learning similarity. This can be easily done by comparing the names of the relation types in the two datasets. For preprocessing the data, we follow previous work (Fu et al., 2017, 2018; Gormley et al., 2015; Nguyen and Grishman, 2016) on ACE05. We use the same document split as theirs and as in the previous chapter. We also followed their split of the relation types for asymmetric relations (directionality taken into account except for *physcial* and *person-social* types). We perform the same preprocessing for the ERE dataset, and follow the document split from the previous chapter.

4.4.2 Multi-task Learning Baseline

Following Chapter 3, we use a similar encoder to obtain the feature representation $\phi(x)$ as our baseline. Following previous work (Fu et al., 2018; Nguyen and Grishman, 2016), the input layer is the concatenation of word embedding, entity embedding and position embeddings. We use pretrained word2vec (Mikolov et al., 2013) as the word embedding with embedding size d_w. The entity embedding and position embeddings are randomly initialized vectors according to the entity type of the token and relative distance to the two arguments of the relation. The embedding sizes are d_e and d_p respectively. We follow previous work for these input embedding sizes as $d_w, d_e, d_p = 300, 50, 50$. It is followed by Bidirectional RNN with attention and a fully connected layer to match the size for the label embedding. We use 150 for the RNN state size and 200 for the label embedding size. The output of this encoder is $\phi(x)$. Then we can perform classification using the scores obtained from Equation 4.1.

In a mini-batch of training step, we randomly select examples from both datasets proportionally according to the dataset size so that the model can finish reading both datasets at the same time after every epoch. Because the difference of the number of examples for the two datasets in the batch, we set $\lambda = \lambda_d \cdot \frac{|D_1|}{|D_1|+|D_2|}$, where $|D_1|$ and $|D_2|$ are the number of examples for each dataset in a single batch. In a special case where the two datasets are actually split from one original dataset, we can set $\lambda_d = 1.0$, and then the two datasets are going to be learned at the same speed. In our case, we use $\lambda_d = 0.8$ so that the two relation tasks are roughly learning at the same speed. As the result, our multi-task model using label embedding is comparable to (Fu et al., 2018) (Table 4.1), which serves as a strong baseline since it is already better than training a single relation task.

4.4.3 Learning the Relatedness between Two Relation Schemas

By learning the relatedness at the same time (Equation 4.4,4.7, $\beta = 0.001$), we obtain better results at the full training set (Table 4.1). The improvement is more obvious with a smaller training set (Table 4.2 at 50%). We also set up a low-resource setting where we only have N examples for each relation type (Figure 4.1 at $N = [10, 20, 30, 40, 50]$). The negatives are randomly selected according to the pos/neg ratio. We can observe larger improvement with less training data. This is also to consider the skewed data distribution in the dataset where there are far more examples for some types than the others. The k supporting prototype examples are drawn randomly at every step. We use $k = 50$ for the experiments and $k = N$ for the low-resource settings. Overall, the improvement is impressive, especially for the low-resource settings. It is also worth noting that the single task models for these low-resource settings obtain virtually zero scores without multi-task learning as there is not enough data to train the encoder. The multi-task learning between two relation tasks is better than training on a single task and more effective for a smaller training set. We now show that learning the relatedness between the types could further improve the model.

	ACE05			
Method	**bc**	**wl**	**cts**	**avg**
Single-task	60.22	53.77	52.01	55.33
Multi-task	60.60	**56.20**	56.72	57.84
+ Relatedness	**62.05**	56.10	**59.12**	**59.08**
Fu et al., 2018	61.67	55.03	56.47	57.72

Table 4.1: Learning the relatedness between types (full training set).

	ACE05			
Method	**bc**	**wl**	**cts**	**avg**
Single-task	54.80	47.27	48.42	50.17
Multi-task	56.67	51.39	55.23	54.43
+ Relatedness	**58.31**	**53.13**	**56.50**	**55.98**
Fu et al., 2018	57.39	51.44	54.28	54.37

Table 4.2: Learning the relatedness between types (50% training).

4.5 Conclusion

We use prototypes of relation types to learn the relatedness between them. In the multi-task learning framework, this joint learning incorporates more knowledge of a relation type for classification. It is impressive that the model can obtain further improvement in addition to sharing the encoder of the sentence. It shows that learning the relationships between relation types could be useful to handle different relation schemas from different datasets. In this paper, we separate the relationships between relation types as related and unrelated. It would be interesting to further explore the relationships between relation types as a more dynamic metric.

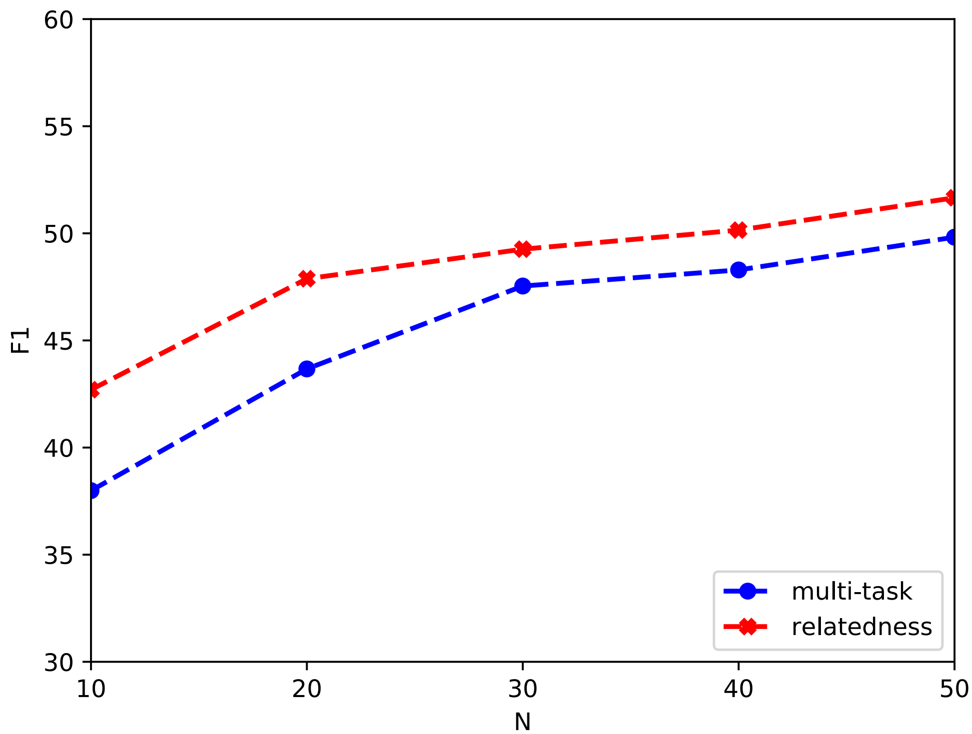

Figure 4.1: Low-resource setting with N examples for each positive relation type on ACE05.

Chapter 5

Conclusion

In this dissertation, I present a multi-task toolkit for neural relation extraction. The frameworks aim to improve the representations of relation models through different auxiliary resources. With new unlabeled entity pairs in the different domains, the model can obtain representations that are better generalized to the target domain. With new labels from different datasets, the model can learn better feature representations with a unified encoder. With prior knowledge of the relationships between relation types, the model can learn more general label embeddings. This set of techniques can be easily applied to relation datasets and tasks. This toolkit can be an effective module for the relation extraction community.

This toolkit is also compatible with most of the other techniques in the community. If someone wants to extract relations in practice, this toolkit can be directly applied. Depending on whether the types exist in the current datasets, we might have to develop a small dataset for the relations first. We can then find the related existing datasets to learn a better encoder according to Chapter 3. If there are multiple related datasets, we would better learn them at the same time to achieve the best performance. When the number of datasets grows, how to keep them learning at the same speed might be a challenge in practice. The next step would be to learn the relatedness between the relation schemas to further improve the model according to Chapter 4. In this chapter, I did experiments with two closely related schemas. Thus, the relatedness between the types is relatively easy to judge. In practice, the decision should be made based on the semantics of the types. There may not be an easily-found label set for capturing the relatedness of two complex ontologies. Even though it is easy for people to judge whether two relation types are related, the answers might not be consistent in a large complicated ontology. Thus, it would be more desirable to have a more flexible or dynamic metric for the relatedness in practice. If the relation task may be applied to some domain-specific documents, we can use the adaptation from Chapter 2 to make the model work better for the target domain. In practice, it might be more ideal to have a domain embedding for each domain and a unified model that can work on the target domain automatically after reading the domain embedding. In this way, the adaptation would focus on learning the domain embedding in the target domain instead of fine-tuning the original model.

At the time of this thesis, Shi et al. (2018) has extended our model in Chapter 2 by adding autoencoders. This might be currently the state-of-the-art domain adaptation model for relation extraction. On the other hand, there have been a variety of model architectures for relation extraction. Currently, Soares et al. (2019) used pretrained language models (BERT) to achieve the best performance on several datasets. Even though they did not evaluate on datasets such as ACE or ERE, it is likely that the BERT-based model will achieve state-of-the-art performance on these datasets in the near future because of the additional knowledge from the pretrained language models. I would recommend using BERT-based models as the shared encoder according to Chapter 3.

5.1 Future Directions of Relation Extraction

As discussed in Chapter 1.3, the majority of current research of relation extraction focuses on how to design a model architecture that better fits the data in the supervised setting. This research assumes we have enough labeled data, which is unlikely to be true in practice. Because of the sparsity of relations and the flexibility of defining relation types, we would better take the opposite assumption that we would only have limited labeled data. Given this constraint, the directions for solving the relation extraction problem could be quite different.

First of all, we could have different models from the supervised setting that better suit low-resource settings or unseen relations. Levy et al. (2017) proposed to use a reading comprehension model for zero-shot relation extraction. Han et al. (2018a) released a few-shot dataset for relations. There are some fine-grained relation types whose names are descriptive enough. Extracting these types of relations can mostly be converted to answering a question. E.g., $place\_of\_birth \rightarrow$ *Where was X born?* These types of relations are more likely to be solved by zero-shot learning if there is a similar type existing in the training or if there is a rich pre-trained model. At least, it would be reasonable to expect good performance from a few-shot model. Soares et al. (2019) has already shown impressive results on the few-shot dataset. However, the dataset is a balanced dataset which contains the same number of negatives as each of the positive relations. In practice, the large skewed positive to negative ratio for relations would be a challenge for these methods. There is no clear way to define the semantics of the negative class of relations. It is only a complement of the positives. The relation types of the dataset are also mostly self-descriptive. Extending these models to more abstract types would be another challenge. Improvement of these methods would give a better practical baseline for relation extraction.

Secondly, in order to improve relation extraction models with minimal labels, we have to utilize more prior knowledge and resources. In this thesis, I attempt to use three kinds of resources: unlabeled entity pairs, multiple labeled datasets, and relationships between relation types. All three of these directions can be further explored. The objective would be to optimize the model given the labeled data plus these additional resources instead of just a single dataset. In addition, any knowledge that helps to explain the entities or the context could be a useful resource. For example, it is hard to generalize from *professor* to include *coach*. Pretrained distributional semantics could be helpful in this case, but may not be sufficient to always link one job title to another one. In the traditional feature-based systems, we can use a list of words as a dictionary and create a feature based on whether a word belongs to this job title list. These kinds of features could be useful for some datasets, but not in general when the list is limited. A more comprehensive knowledge resource is needed for better generalization such as the relational nouns in Nombank (Meyers et al., 2004). With more complete knowledge resources in the future, this external knowledge could be more effective. It can also be better utilized with multi-task learning or other methods instead of just checking the list. The additional knowledge may also not have to be perfect to be useful (Vashishth et al., 2018). The capability of external knowledge could be underestimated currently for relation extraction.

Last, the most direct way to improve relation extraction models is still by acquiring more labels, especially positive labels. There is currently not much research about how to obtain labels more efficiently for relation extraction. ICE (He and Grishman, 2015) is a tool for acquiring more relation labels interactively. The assumption is that relation tasks are often domain-specific and require customization. it is also based on the research that it is more efficient to obtain labels in a interactive way (Fu and Grishman, 2013). This kind of research focused on active learning in a simulated environment. Because of the sparsity of relations, active learning could help a lot in finding positive relations and thus substantially improve the model with limited annotations. Nevertheless, there has been little research about how to acquire more labels interactively. It could be due to the difficulty of developing better query functions for active learning in a simulated environment. However, even a basic uncertainty-based active learning strategy would give large improvement over random or sequential annotation strategy for relations. In real applications, this interactive approach could be a better choice for acquiring more labels. It might be more important to address the other concerns of using these kinds of interactive tools in practice. Active learning would often give a skewed sample from large unlabeled data. It is not easy to show that this skewed sample would be as useful if evaluating in settings different from the original simulated experiments. The annotation cost for different examples could also be dramatically different in practice, even though they are all counted as one example in a simulated experiment. Annotators would spend far less time on a short easy sentence than a long ambiguous sentence. Since we have large amounts of unlabeled data, is it sufficient to just annotate short simple examples? The hidden structures of relations are often just simple patterns or some keywords. Can we generalize simple sentences to more complicated context if we have enough labels for simple examples? In other words, if we change the way we count the annotation cost (e.g., use the number of words as the cost instead of the number of sentences), then the query function could be very different. The potential factors of obtaining labels interactively for efficient models are not well explored at the moment.

5.2 Applications of Knowledge of Relations

The direct result of extracting relations between entities is adding more edges in the knowledge graph. Applications could be built based on the knowledge graph such as question answering or summarization in the form of personal assistants or search engine results. In addition to the general purpose knowledge base, the applications of relation tasks are often considered domain-specific. Relation tasks in the biomedical domain have a long history such as the Protein-Protein Interaction. There are also potential applications for other specialized areas such as legal documents or scientific papers. The automatic tools built in each domain could work as the assistants for the professionals.

For the biomedical domain, there has been a lot of research and many datasets (Pyysalo et al., 2008) such as AIMed (Bunescu et al., 2005), BioInfer (Pyysalo et al., 2007), HPRD50 (Fundel et al., 2006), IEPA (Ding et al., 2001) and more recently the BioNLP shared tasks. Researchers try to build knowledge bases for medical entities such as proteins, drugs, and diseases from text. For scientific documents, finding relations between research projects can potentially predict future directions of technology (Meyers et al., 2014). For legal documents, automatic extraction of different parties, judges and other participants in a case could provide enormous data for quantitative analysis. This could lead to more equal decisions for different cases and artificial judges in the future. Text in all these domains is hard for an average person to read. Annotation would require domain experts.

Building knowledge bases for these professional tasks is not an easy job, and may require substantial manpower to build a real application. There might be enough funding to support such an effort in the biomedical domain or in legal document processing to replace doctors and lawyers. It would be hard for people to build a customized knowledge base for a narrow scope or less profitable domain. This would only be possible in the future if we can substantially improve the models and the customization tools without requiring much annotation.

Bibliography

Aguilar, Jacqueline, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis (2014). "A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards". In: *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pp. 45–53.

Ajakan, Hana, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand (2014). "Domain-adversarial neural networks". In: *arXiv preprint arXiv:1412.4446*.

Blitzer, John, Ryan McDonald, and Fernando Pereira (2006). "Domain Adaptation with Structural Correspondence Learning". In: *EMNLP*.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pp. 1247–1250.

Bunescu, Razvan C. and Raymond J. Mooney (2005a). "A shortest path dependency kernel for relation extraction". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 724-731. Association for Computational Linguistics.*

Bunescu, Razvan C. and Raymond J. Mooney (2005b). “Subsequence kernels for relation extraction”. In: *Advances in Neural Information Processing Systems, pp. 171-178.*

Bunescu, Razvan, Ruifang Ge, Rohit J Kate, Edward M Marcotte, Raymond J Mooney, Arun K Ramani, and Yuk Wah Wong (2005). “Comparative experiments on learning information extractors for proteins and their interactions”. In: *Artificial intelligence in medicine* 33.2, pp. 139–155.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning phrase representations using rnn encoder–decoder for statistical machine translation”. In: *Proceedings of EMNLP*.

Cortes, Corinna and Vladimir Vapnik (1995). “Support-vector networks”. In: *Machine learning* 20.3, pp. 273–297.

Daume, Hal (2007). “Frustratingly Easy Domain Adaptation”. In: *ACL*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Ding, Jing, Daniel Berleant, Dan Nettleton, and Eve Wurtele (2001). “Mining MEDLINE: abstracts, sentences, or phrases?” In: *Biocomputing 2002*. World Scientific, pp. 326–337.

Dos Santos, Cicero, Bing Xiang, and Bowen Zhou (2015). “Classifying Relations by Ranking 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 1: Long Papers)*. Vol. 1, pp. 626–634.

Du, Jinhua, Jingguang Han, Andy Way, and Dadong Wan (2018). "Multi-Level Structured Self-Attentions for Distantly Supervised Relation Extraction". In: *arXiv preprint arXiv:1809.00699*.

Duong, Long, Trevor Cohn, Steven Bird, and Paul Cook (2015). "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser". 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), vol. 2, pp. 845-850.*

Fu, Lisheng and Ralph Grishman (2013). "An efficient active learning framework for new relation types". In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 692–698.

Fu, Lisheng, Thien Huu Nguyen, Bonan Min, and Ralph Grishman (2017). "Domain Adaptation for Relation Extraction with Domain Adversarial Neural Network". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), vol. 2, pp. 425-429.*

Fu, Lisheng, Bonan Min, Thien Huu Nguyen, and Ralph Grishman (2018). "A Case Study on Learning a Unified Encoder of Relations". In: *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pp. 202–207.

Fundel, Katrin, Robert Küffner, and Ralf Zimmer (2006). "RelEx—Relation extraction using dependency parse trees". In: *Bioinformatics* 23.3, pp. 365–371.

Ganin, Yaroslav and Victor Lempitsky (2015). "Unsupervised Domain Adaptation by Backpropagation". In: *ICML*.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Domain adaptation for large-scale sentiment classification: A deep learning approach". In: *ICML*.

Gormley, Matthew R., Mo Yu, and Mark Dredze (2015). “Improved Relation Extraction with Feature-Rich Compositional Embedding Models”. In: *EMNLP*.

Han, Xu, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun (2018a). “FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 4803–4809. DOI: 10.18653/v1/D18-1514.

Han, Xu, Pengfei Yu, Zhiyuan Liu, Maosong Sun, and Peng Li (2018b). “Hierarchical Relation Extraction with Coarse-to-Fine Grained Attention”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2236–2245.

He, Yifan and Ralph Grishman (2015). “ICE: Rapid Information Extraction Customization for NLP Novices”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Denver, Colorado: Association for Computational Linguistics, pp. 31–35. DOI: 10.3115/v1/N15-3007.

Hendrickx, Iris, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz (2009). “Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals”. In: *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, pp. 94–99.

Hoffmann, Raphael, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld (2011). “Knowledgebased weak supervision for information extraction of overlapping relations”. In: *Proceedings of ACL*.

Jing, Jiang and ChengXiang Zhai (2007). “Instance weighting for domain adaptation in NLP”. In: *ACL*.

Kambhatla, Nanda (2004). “Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction”. In: *Proceedings of the ACL Interactive Poster and Demonstration Sessions*.

Kingma, Diederik P. and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *ICLR*.

LeCun, Yann, Leon Bottou, Yoshua Bengio, and Patrick Haffner (1998). “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of IEEE*.

Levy, Omer, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer (2017). “Zero-Shot Relation Extraction via Reading Comprehension”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 333–342. DOI: 10.18653/v1/K17-1034.

Li, Qi and Heng Ji (2014). “Incremental joint extraction of entity mentions and relations”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 402–412.

Lin, Yankai, Shiqi Shen, Zhiyuan Liu, Luan Huanbo, and Sun Maosong (2016). “Neural Relation Extraction with Selective Attention over Instances”. In: *Proceedings of ACL*.

Liu, Tianyi, Xinsong Zhang, Wanhao Zhou, and Weijia Jia (2018). “Neural relation extraction via inner-sentence noise reduction and transfer learning”. In: *arXiv preprint arXiv:1808.06738*.

Liu, Tianyu, Kexiang Wang, Baobao Chang, and Zhifang Sui (2017). “A soft-label method for noise-tolerant distantly supervised relation extraction”. In:

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1790–1795.

Liu, Yang, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang (2015). "A dependency-based neural network for relation classification". In: *arXiv preprint arXiv:1507.04646*.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman (2004). "Annotating Noun Argument Structure for NomBank." In: *LREC*. Vol. 4, pp. 803–806.

Meyers, Adam, Giancarlo Lee, Angus Grieve-Smith, Yifan He, and Harriet Taber (2014). "Annotating Relations in Scientific Articles." In: *LREC*, pp. 4601–4608.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *ICLR*.

Min, Bonan, Zhuolin Jiang, Marjorie Freedman, and Ralph Weischedel (2017). "Learning Transferable Representation for Bilingual Relation Extraction via Convolutional Neural Networks". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), vol. 1, pp. 674-684.*

Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky (2009). "Distant supervision for relation extraction without labeled data". In: *Proceedings of ACL*.

Miwa, Makoto and Mohit Bansal (2016). "End-to-end relation extraction using lstms on sequences and tree structures". In: *arXiv preprint arXiv:1601.00770*.

Nguyen, Minh Luan, Ivor W. Tsang, Kian Ming Adam Chai, and Hai Leong Chieu (2014). "Robust Domain Adaptation for Relation Extraction via Clustering Consistency". In: *ACL*.

Nguyen, Thien Huu and Ralph Grishman (2014). “Employing Word Representations and Regularization for Domain Adaptation of Relation Extraction”. In: *ACL*.

Nguyen, Thien Huu and Ralph Grishman (2015a). “Relation Extraction: Perspective from Convolutional Neural Networks”. In: *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Nguyen, Thien Huu and Ralph Grishman (2016). “Combining Neural Networks and Log-linear Models to Improve Relation Extraction”. In: *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Nguyen, Thien Huu, Barbara Plank, and Ralph Grishman (2015b). “Semantic Representations for Domain Adaptation: A Case Study on the Tree Kernel-based Method for Relation Extraction”. In: *ACL-IJCNLP*.

Plank, Barbara and Alessandro Moschitti (2013). “Embedding Semantic Similarity in Tree Kernels for Domain Adaptation of Relation Extraction”. In: *ACL*.

Pyysalo, Sampo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski (2007). “BioInfer: a corpus for information extraction in the biomedical domain”. In: *BMC bioinformatics* 8.1, p. 50.

Pyysalo, Sampo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski (2008). “Comparative analysis of five protein-protein interaction corpora”. In: *BMC bioinformatics*. Vol. 9. 3. BioMed Central, S6.

Ratnaparkhi, Adwait (1999). “Learning to parse natural language with maximum entropy models”. In: *Machine learning* 34.1-3, pp. 151–175.

Riedel, Sebastian, Limin Yao, and Andrew McCallum (2010). “Modeling relations and their mentions without labeled text”. In: *Proceedings of ECML*.

Riedel, Sebastian, Limin Yao, Andrew McCallum, and Benjamin M Marlin (2013). "Relation extraction with matrix factorization and universal schemas". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 74–84.

Shi, Ge, Chong Feng, Lifu Huang, Boliang Zhang, Heng Ji, Lejian Liao, and Heyan Huang (2018). "Genre Separation Network with Adversarial Training for Cross-genre Relation Extraction". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1018–1023.

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*, pp. 4077–4087.

Soares, Livio Baldini, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski (2019). "Matching the Blanks: Distributional Similarity for Relation Learning". In: *arXiv preprint arXiv:1906.03158*.

Socher, Richard, Brody Huval, Christopher D Manning, and Andrew Y Ng (2012). "Semantic compositionality through recursive matrix-vector spaces". In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pp. 1201–1211.

Song, Zhiyi, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma (2015). "From light to rich ERE: annotation of entities, relations, and events". In: *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pp. 89–98.

Surdeanu, Mihai, Julie Tibshirani Tibshirani, Ramesh Nallapati, and Christopher D Manning (2012). "Multi-instance multi-label learning for relation extraction". In: *Proceedings of EMNLP*.

Vashishth, Shikhar, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar (2018). "RESIDE: Improving Distantly-Supervised Neural Relation Extraction using Side Information". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 1257–1266. DOI: 10.18653/v1/D18-1157.

Xu, Yan, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin (2015). "Classifying relations via long short term memory networks along shortest dependency paths". In: *proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1785–1794.

Ye, Hai, Wenhan Chao, Zhunchen Luo, and Zhoujun Li (2017). "Jointly Extracting Relations with Class Ties via Effective Deep Ranking". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Zelenko, Dmitry, Chinatsu Aone, and Anthony Richardella (2003). "Kernel methods for relation extraction". In: *Journal of machine learning research* 3.Feb, pp. 1083–1106.

Zeng, Daojian, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao (2014). "Relation Classification via Convolutional Deep Neural Network". In: *COLING*.

Zeng, Daojian, Kang Liu, Yubo Chen, and Jun Zhao (2015). "Distant supervision for relation extraction via piecewise convolutional neural networks". In: *Proceedings of EMNLP*.

Zhang, Min, Jie Zhang, Jian Su, and Guodong Zhou (2006). "A composite kernel to extract relations between entities with both flat and structured features". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 825–832.

Zhang, Shu, Dequan Zheng, Xinchen Hu, and Ming Yang (2015). "Bidirectional Long Short-Term Memory Networks for Relation Classification". In: *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*.

Zhao, Shubin and Ralph Grishman (2005). "Extracting relations with integrated information using kernel methods". In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 419–426.

Zhou, GuoDong, Jian Su, Jie Zhang, and Min Zhang (2005). "Exploring Various Knowledge in Relation Extraction". In: *ACL*.

Zhou, Guodong, Min Zhang, DongHong Ji, and Qiaoming Zhu (2007). "Tree kernel-based relation extraction with context-sensitive structured parse tree information". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Zhou, Peng, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu (2016). "Attention-based bidirectional long short-term memory networks for relation classification". In: *Proceedings of ACL*.