Towards Neural Information Extraction without Manual Annotated Data

by

Peng Xu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Information extraction (IE) is one of the most important technologies in the information age. Applying information extraction to text is linked to the problem of text simplification in order to create a structured view of the information present in free text. However, information extraction is a very challenging task, due to the inherent difficulties to understand natural language and the high cost to obtain large manual annotated training data. In this thesis, we build on the premise of performing automatic information extraction without manual annotated data following the distant supervision paradigm and present novel neural models for different IE tasks which are particularly suited for this setting.

In the first part of the thesis, we focus on one IE task – fine-grained entity type classification (FETC) and propose the NFETC model – a single, much simpler and more elegant neural network model that attempts FETC "end-to-end" without post-processing or ad-hoc features. We study two kinds of noise, namely *out-of-context* noise and *overly-specific noise*, for noisy type labels and investigate their effects on FETC systems. We propose a neural network based model which jointly learns representations for entity mentions and their context. A variant of cross-entropy loss function is used to handle *out-of-context noise*. Hierarchical loss normalization is introduced into our model to alleviate the effect of *overly-specific* noise.

In the second part of the thesis, we focus on another IE task – relation extraction (RE) and propose a neural model with multiple level of attention

mechanisms. The model can make full use of all informative words and sentences and alleviate the wrong labelling problem for distant supervised relation extraction.

In the third part of the thesis, we attempt to leverage knowledge base embedding methods to facilitate relation extraction and describe a novel neural framework HRERE to jointly learning heterogeneous representations from both text information and facts in an existing knowledge base. A novel loss function is introduced to connect the heterogeneous representations seamlessly allowing them to enhance each other.

Overall, the work in this thesis tackles different tasks of IE under the setting of distant supervision with DNNs, different attentions, different loss functions and the help of knowledge base embeddings. All the proposed models got state-of-the-art performance in representative tasks.

Preface

This thesis is an original work by Peng Xu. Publications accompanying it are listed in the following:

1. P. Xu, and D. Barbosa. "Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss". *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2018.* Long oral paper, New Orleans, LA, U.S.A., Jun. 1-6, ACL.

2. P. Xu, and D. Barbosa. "Connecting Language and Knowledge with Heterogeneous Representations for Neural Relation Extraction". Submitted to *Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018.*

3. P. Xu, and D. Barbosa. "Investigations on Knowledge Base Embedding for Relation Prediction and Extraction". *arXiv preprint, arXiv:1802.02114, 2018.*

To my parents.

We are drowning in information but starved for knowledge.

– John Naisbitt.

Acknowledgements

I would like to sincerely thank my great supervisor Denilson Barbosa for his insightful advice and patient support. I am very grateful to my superviosr for his encouragement and the freedom to explore and pursue various research directions. I am deeply grateful to Greg Kondrak for serving in my supervisory committee, for many fruitful discussions we had during my master study.

I would also like to thank the open-sourced community of DBpedia for taking me as a student at Google Summer of Code in 2016 to bring me into the field of knowledge bases. I gained valuable research experience in information extraction and knowledge base embeddings. In 2017 and 2018, I am honored to serve as a mentor for DBpedia to guide student projects at Google Summer of Code.

I am also grateful for all my friends and colleagues, who have made my experiences in Edmonton extremely worthwhile and memorable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The process of Information Extraction (IE) is the task of automatically extracting structured information from unstructured documents, for example for populating Knowledge Bases (KBs). In most of the cases, this process concerns processing human language texts by means of Natural Language Processing (NLP). Imagine that you are not a big fan of basketball but want to know more about it for socializing. One day, you randomly found one news article started with:

> *Steve Kerr has turned down Phil Jackson and the New York Knicks to accept a five-year, $25 million offer to become the Golden State Warriors' next coach, saying "it just felt like the right move on many levels."*

To understand the information encoded in this sentence, you will need to at least know the people and organizations mentioned and the semantic relations among them. The first step in most IE tasks is to find the **named entities** in a text. The task of **named entity recognition** (NER) is to find each **mention** of a named entity in the text and label its type. While traditional named entity recognition systems (Manning et al. 2014; Ren, El-Kishky, et al. 2015) focus on a small set of coarse types (typically fewer than 10), recent studies (Dan Gillick et al. 2014; Ling and Weld 2012) work on a much larger set of fine-grained types which form a tree-structured hierarchy. We call this task as **fine-grained entity type classification** (FETC).

Having located all of the mentions of named entities in a text, we need to find the semantic relations among the entities, for example, the relation between *Steve Kerr* and *Golden State Warriors* becomes **coach-of** given the above sentence. The task of **relation extraction** (RE) is to find and classify semantic relations among entity mentions.

A major challenge in information extraction tasks like FETC and RE is the absence of human-annotated data. The process of manually labeling a training set with large number of fine-grained types or semantic relations is too expensive and error-prone (hard for annotators to distinguish over many types or relations consistently). Current systems resort to distant supervision (Mintz et al. 2009) and annotate training corpora automatically using KBs. A typical workflow of distant supervision is as follows: (1) identify entity mentions in the documents; (2) link mentions to entities in KB; (3) assign, to each entity mention or entity pair, all types or relations associated in a KB. However, this approach introduces *label noise* to the mentions since it fails to take the semantics of the mentions' local context into account when assigning the labels. This thesis mainly studies information extraction under the setting of distant supervision which doesn't require massive manual annotated data and addresses the introduced noise in various ways.

A few years ago, almost all NLP problems were dominated by shallow machine learning methods with intensive feature engineering. The resurgence of deep neural networks (DNNs) enables the possibility of solving NLP problems via deep systems with no or fewer manual features. Most NLP tasks have acquired state-of-the-art by DNN systems. A brief introduction to deep neural networks for NLP is given in Appendix A.1 with a special focus in the subarea of information extraction. This thesis presents our work employing neural models for the tasks of information extraction.

Since the training data are automatically labelled based on KBs, a related task is that of Knowledge Base Embedding (KBE), which is concerned with representing KB entities and relations in a vector space for link prediction. Appendix A.2 will give a short introduction into the basic background and popular methods of KBE. This thesis also investigates various ways to leverage

KBE to facilitate neural information extraction.

1.1 Thesis Statement and Contributions

The present significance of IE pertains to the growing amount of information available in unstructured form. Due to the inherent noise existing in the automatically labelled data and the difficulties in understanding natural language, traditional information extraction methods can hardly suffice nowadays. To overcome these problems, this thesis is concerned with neural information extraction without manual annotated data. More precisely, this thesis investigates whether information extraction can be effectively performed under the setting of distant supervision with the help of deep neural networks and information from state-of-the-art open knowledge bases. The main contributions are as follows:

Neural Fine-Grained Entity Type Classification with Hierarchy-Aware Loss. We propose an end-to-end solution with a neural network model that uses a variant of cross-entropy loss function to handle out-of-context labels, and hierarchical loss normalization to cope with overly-specific ones. Also, previous work solve FETC a multi-label classification followed by ad-hoc post-processing. In contrast, our solution is more elegant: we use public word embeddings to train a single-label that jointly learns representations for entity mentions and their context. We show experimentally that our approach is robust against noise and consistently outperforms the state-of-the-art on established benchmarks for the task. This work has been accepted as a long oral paper in the *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2018.*

Neural Relation Extraction with Multi-Level Attention Mechanisms. We propose a neural model with multi-level attention mechanisms for relation extraction. In this model, we employ bi-directional long short term memory to embed the semantics of sentences. Afterwards, we build both word-level and sentence-level attention, which is expected to dynamically reduce the weights of those noisy instances. Experimental results on real-world datasets show that, our model achieves significant and consistent improvements on relation extraction as compared with baselines.

Incorporating Knowledge Base Embedding to Facilitate Neural Relation Extraction. We describes and evaluates a novel neural framework for jointly learning representation for RE and KBE tasks that uses a cross-entropy loss function to ensure both representations are learned together, resulting in significant improvements over the current state-of-the-art for relation extraction. This work has been submitted to *Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018.*

Chapter 2

Neural Fine-Grained Entity Type Classification (NFETC)

In this chapter, we first give an introduction to the task of FETC in Section 2.1. In Section 2.2, we briefly review some important methods for FETC, as well as their advantages and disadvantages. In Section 2.3, we define the problem of FETC in a formal way. At last, we propose a single, much simpler and more elegant neural network model that attempts FETC "end-to-end" without post-processing or ad-hoc features and improves on the state-of-the-art for the task in Section 2.4 and 2.5.

2.1 Introduction

Entity type classification aims at assigning types or labels such as **Person**, **Organization** to entity mentions in a document. There has been considerable amount of work on Named Entity Recognition (NER) (Manning et al. 2014; Ren, El-Kishky, et al. 2015), which classifies entity mentions into a small set of mutually exclusive types. However, these types are not enough for some NLP applications such as relation extraction, knowledge base completion, entity linking and question answering.

Fine-grained Entity Type Classification (FETC) aims at labeling entity mentions in context with one or more specific types organized in a hierarchy (e.g., **actor** as a subtype of **artist**, which in turn is a subtype of **person**). In relation extraction and knowledge base completion, knowing fine-grained

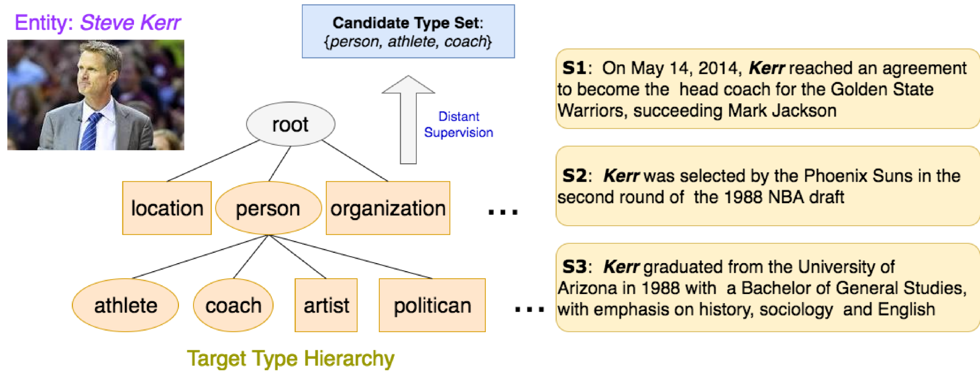

Figure 2.1: With distant supervision, all the three mentions of *Steve Kerr* shown are labeled with the same types in oval boxes in the target type hierarchy. While only part of the types are correct: **person** and **coach** for **S1**, **person** and **athlete** for **S2**, and just **person** for **S3**.

types for entities can significantly increase the performance of the relation extractor (Dong et al. 2014; Mintz et al. 2009) since it helps in filtering out candidate relation types that do not follow the type constrain. For entity linking and question answering, fine-grained entity types provide additional information while matching entities or questions to its potential candidates and significantly improves the performance (Li and D. Roth 2002; T. Lin, Etzioni, et al. 2012).

There are several characteristics of FETC, as compared to traditional coarse-grained entity type classification:

Context dependent labeling. Typically, the set of acceptable labels for a mention is constrained to what can be deduced from its context (Dan Gillick et al. 2014). It's reasonable to assume that one mention can only have one *type-path* along the hierarchy depending on the context. Because of the high cost in labeling large training corpora with fine-grained types, current FETC systems resort to distant supervision (Mintz et al. 2009) and annotate mentions in the training corpus with *all types* associated with the entity in a knowledge graph without considering local context. This is illustrated in Figure 2.1, with three training sentences about entity *Steve Kerr*. Note that while the entity belongs to three fine-grained types (**person**, **athlete**, and **coach**), some sentences provide evidence of only some of the types: **person** and **coach** from **S1**, **person** and **athlete** from **S2**, and just **person** for **S3**. The importance of the context motivates us to use attentive neural models to effectively encode the contextual information.

Hierarchical structure of entity types. As shown in Figure 2.1, fine-grained types form a tree-structured is-a hierarchy. This means that each *type-path* can be uniquely represented by the terminal type (not necessarily a leaf node). For example, a mention with *type-path* **person** $\rightarrow$ **artist** can be uniquely represented by **artist**. The hierarchy can be seen as the prior information which can be exploited to help the learning procedure.

Collapse of the mutual exclusion assumption. The assumption of the traditional entity type classification that the labels of entities are mutually exclusive, does not hold for FETC. For example, *Steve Kerr* is both **coach**, **athlete** and **person** as shown in Figure 2.1. As a result, it's natural to formulate the task as a multi-label classification problem, rather than a single label classification problem as in the case of traditional entity type classification.

However, based on the assumption that each mention can only have one *type-path* depending on the context and the fact that each *type-path* can be uniquely represented by the terminal type, FETC can actually be transformed into a single label classification problem. There are many advantages in adopting the single label setting. For example, it can simplify the problem and benefit from previous research on hierarchical classification. Moreover, no post-processing is needed anymore compared to the multi-label setting. Aside from the advantages, one obvious disadvantage can be seen. That is, the performance upper bounds of our proposed model are no longer 100%, since there are samples in the testing set which violate the assumption that each mention can only have *type-path*. But, as we can see in Section 2.5 later, the influence of this disadvantage is negligible.

Noise in automatically annotated data. As discussed above, direct distant supervision leads to noisy training data which can hurt the accuracy of the FETC model. One kind of noise introduced by distant supervision is assigning labels that are *out-of-context* (**athlete** in **S1** and **coach** in **S2**). The most direct strategies to handle *out-of-context* noise are ignoring *out-of-context* types or using simple pruning heuristics like discarding examples with multiple types. However, both strategies are inelegant and hurt accuracy.

Another source of noise introduced by distant supervision is when the type is *overly-specific* for the context. For instance, example **S3** does not support the inference that Mr. Kerr is either an **athlete** or a **coach**. Since existing knowledge graphs give more attention to notable entities with more specific types, *overly-specific* labels bias the model towards popular subtypes instead of generic ones, *i.e.*, prefer **athlete** over **person**. Instead of treating each type equally and independently, the key to handle this kind of noise is to make the models understand the given type hierarchy which encodes the underlying type correlations.

2.2 Methods for Fine-Grained Entity Type Classification

The first work to use distant supervision (Mintz et al. 2009) to induce a large but noisy training set and manually label a significantly smaller dataset to evaluate their FETC system, was Ling and Weld (2012) who introduced both a training and evaluation dataset FIGER (GOLD). They used a linear classifier perceptron for multi-label classification. While initial work largely assumed that mention assignments could be done independently of the mention context, Dan Gillick et al. (2014) introduced the concept of context-dependent FETC where the types of a mention are constrained to what can be deduced from its context and introduced a new OntoNotes-derived manually annotated evaluation dataset. Yogatama, Daniel Gillick, and Lazic (2015) proposed an embedding-based model where user-defined features and labels were embedded into a low dimensional feature space to facilitate information sharing among labels. Ma, Cambria, and Gao (2016) presented a label embedding method that incorporates prototypical and hierarchical information to learn pre-trained label embeddings and adapted a zero-shot framework that can predict both seen and previously unseen entity types.

Shimaoka et al. (2016) proposed an attentive neural network model that used LSTMs to encode the context of an entity mention and used an attention mechanism to allow the model to focus on relevant expressions in such context. Shimaoka et al. (2017) summarizes many neural architectures for FETC task. These models ignore the *out-of-context noise*, that is, they assume that all labels obtained via distant supervision are "correct" and appropriate for every context in the training corpus.

Ren, He, Qu, L. Huang, et al. (2016) have proposed AFET, an FETC system, that separates the loss function for *clean* and *noisy* entity mentions and uses label-label correlation information obtained by given data in its parametric loss function. Considering the noise reduction aspects for FETC systems, Ren, He, Qu, Voss, et al. (2016) introduced a method called LNR to reduce label noise without data loss, leading to significant performance gains on both the evaluation dataset of FIGER(GOLD) and OntoNotes. Although these works consider both *out-of-context noise* and *overly-specific noise*, they rely on hand-crafted features which become an impediment to further improvement of the model performance. For LNR, because the noise reduction step is separated from the FETC model, the inevitable errors introduced by the noise reduction will be propagated into the FETC model which is undesirable.

Most recently, following the idea from AFET, Abhishek, Anand, and Awekar (2017) proposed a simple neural network model which incorporates noisy label information using a variant of non-parametric hinge loss function and gain great performance improvement on FIGER(GOLD). However, their work overlooks the effect of *overly-specific noise*, treating each type label equally and independently when learning the classifiers and ignores possible correlations among types.

We can see that all of these methods have flaws from different aspects, including using manual features, without using attentive neural networks and ignoring *out-of-context* and *overly-specific* noise. Besides these limitations, all these methods tread FETC as a multi-label classification problem: during type inference they predict a plausibility score for each type, and then, either classify types with scores above a threshold or perform a top-down search in the given type hierarchy. While this post-processing step is inelegant and can be removed by adopting the single label setting.

Our Approach. We adopt the single label setting and propose a neural model to overcome all the above drawbacks of existing FETC methods. With publicly available word embeddings as input, we learn two different entity representations and use bidirectional long-short term memory (LSTM) with attention to learn the context representation. We propose a variant of cross entropy loss function to handle *out-of-context* labels automatically during the training phase. Also, we introduce hierarchical loss normalization to adjust the penalties for correlated types, allowing our model to understand the type hierarchy and alleviate the negative effect of *overly-specific* labels. Finally, we report on an experimental validation against the state-of-the-art on established benchmarks that shows that our model can adapt to noise in training data and consistently outperform previous methods. In summary, we describe a single, much simpler and more elegant neural network model that attempts FETC "end-to-end" without post-processing or ad-hoc features and improves on the state-of-the-art for the task.

In the next section, we first give the formal problem formulation of FETC in the single label setting, and then, describe the details of our proposed model in Section 2.4.

2.3 Problem Formulation

The task of FETC is to automatically reveal the type information for entity mentions in context. The input is a knowledge graph Ψ with type schema $\mathcal{Y}_\Psi$, a target type hierarchy $\mathcal{Y}$ which covers a subset of types in Ψ, *i.e.*, $\mathcal{Y} \subseteq \mathcal{Y}_\Psi$, and an automatically labeled training corpus $\mathcal{D}$ obtained by distant supervision

	Attentive	AFET	LNR	AAA	**NFETC**
without manual features	✗	✗	✗	✓	✓
use attentive neural network	✓	✗	✗	✗	✓
adopt single label setting	✗	✗	✗	✗	✓
handle *out-of-context* noise	✗	✓	✓	✓	✓
handle *overly-specifc* noise	✗	✓	✓	✗	✓

Table 2.1: Summary comparison to related FETC work. FETC systems listed in the table: (1) **Attentive** (Shimaoka et al. 2017); (2) **AFET** (Ren, He, Qu, L. Huang, et al. 2016); (3) **LNR** (Ren, He, Qu, Voss, et al. 2016); (4) **AAA** (Abhishek, Anand, and Awekar 2017).

with $\mathcal{Y}$. The output is a *type-path* in $\mathcal{Y}$ for each mention from the test corpus $\mathcal{D}_t$.

A KB with a set of entities $\mathcal{E}_\Psi$ contains human-curated facts on both entity-entity facts of various relation types and entity-type facts. For FETC, what we care about is *entity-type facts* in a KB Ψ (with type schema $\mathcal{Y}_\Psi$) as $\mathcal{T}_\Psi = \{(e, y)\} \subset \mathcal{E}_\Psi \times \mathcal{Y}_\Psi$. A *target type hierarchy* is a tree where nodes represent types of interests from $\mathcal{Y}_\Psi$.

Formally, an automatically labeled corpus for entity type classification consists of a set of extracted *entity mentions* $\{m_i\}_{i=1}^N$ (*i.e.*, token spans representing entities in the text), the associated *entities* $\{e_i\}_{i=1}^N$, the *context* (*e.g.*, sentence, paragraph) of each mention $\{c_i\}_{i=1}^N$, and the *candidate type sets* $\{\mathcal{Y}_i\}_{i=1}^N$ automatically generated for each mention. We represent the training corpus using a set of mention-based triples $\mathcal{D} = \{(m_i, e_i, c_i, \mathcal{Y}_i)\}_{i=1}^N$.

If $\mathcal{Y}_i$ is free of *out-of-context noise*, the type labels for m_i should form a single *type-path* in $\mathcal{Y}_i$, which serves as a *context-dependent* type annotation for m_i. However, $\mathcal{Y}_i$ may contain *type-paths* that are irrelevant to m_i in c_i if there exists *out-of-context noise*. We denote the type set including all terminal types for each *type-path* as the target type set $\mathcal{Y}_i^t$. In the example type hierarchy shown in Figure 1, if $\mathcal{Y}_i$ contains type **person**, **athlete**, **coach**, $\mathcal{Y}_i^t$ should contain **athlete**, **coach**, but not **person**. In order to check the trade-off between effect of *out-of-context noise* and the size of the training set, we construct two different training sets. $\mathcal{D}_{filtered}$ only with triples whose $\mathcal{Y}_i$ form a single *type-path* in $\mathcal{D}$, and $\mathcal{D}_{raw}$ with all triples in $\mathcal{D}$.

Then, the FETC problem can be formulated as a single-label classification problem as follows:

Definition 1 *Given an entity mention* $m_i = (w_p, \ldots, w_t)(p, t \in [1, T], p \leq t$, *its associated entity* e_i *and its context* $c_i = (w_1, \ldots, w_T)$ *where* T *is the context length, our task is to predict its most specific type* $\hat{y}_i$ *which is the terminal type of the predicted* type-path.

In practice, c_i is generated by truncating the original context with words beyond the context window size C both to the left and to the right of m_i. Specifically, we compute a probability distribution over all the $K = |\mathcal{Y}|$ types in the target type hierarchy $\mathcal{Y}$. The type with the highest probability is classified as the predicted type $\hat{y}_i$ which is the terminal type of the predicted *type-path*.

2.4 NFETC with Hierarchy-Aware Loss

In this section we describe Neural Fine-Grained Entity Type Classification (NFETC) model in detail. Figure 2.2 shows the architecure of the NFETC model.

2.4.1 Input Representation

As stated in Section 2.3, the input is an entity mention m_i with its context c_i. First, we transform each word in the context c_i into a real-valued vector to provide lexical-semantic features. Given a word embedding matrix W^{wrd} of size $d_w \times |V|$, where V is the input vocabulary and d_w is the size of word embedding, we map every w_i to a column vector $\mathbf{w}_i^d \in \mathbb{R}^{d_w}$.

To additionally capture information about the relationship to the target entities, we incorporate word position embeddings (D. Zeng, K. Liu, Lai, et al. 2014) to reflect relative distances between the i-th word to the entity mention. Every relative distance is mapped to a randomly initialized position vector in $\mathbb{R}^{d_p}$, where d_p is the size of position embedding. For a given word, we obtain the position vector $\mathbf{w}_i^p$. The overall embedding for the i-th word is $\mathbf{w}_i^E = [(\mathbf{w}_i^d)^\top, (\mathbf{w}_i^p)^\top]^\top$.

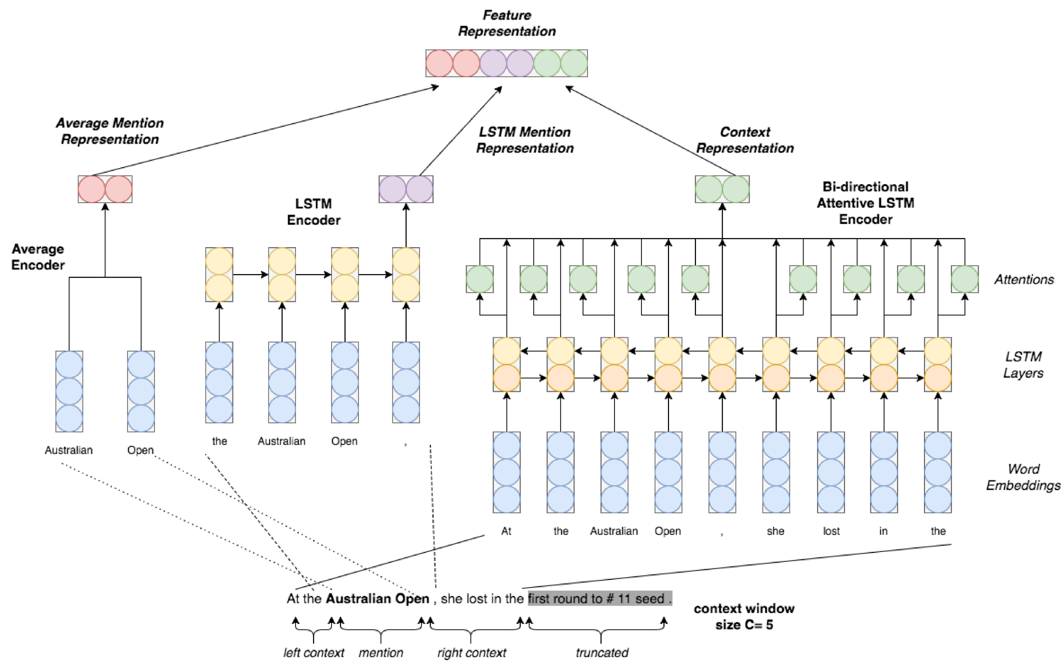

Figure 2.2: The architecture of the NFETC model.

2.4.2 Context Representation

For the context c_i, we want to apply a non-linear transformation to the vector representation of c_i to derive a context feature vector $h_i = f(c_i; \theta)$ given a set of parameters θ. In this paper, we adopt bidirectional LSTM with d_s hidden units as $f(c_i; \theta)$. The network contains two sub-networks for the forward pass and the backward pass respectively. Here, we use element-wise sum to combine the forward and backward pass outputs. The output of the i-th word in shown in the following equation:

$$h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}] \tag{2.1}$$

Following (P. Zhou et al. 2016), we employ word-level attention mechanism, which makes our model able to softly select the most informative words during training. Let H be a matrix consisting of output vectors $[h_1, h_2, \ldots, h_T]$ that the LSTM produced. The context representation r is formed by a weighted sum of these output vectors:

$$G = \tanh(H) \tag{2.2}$$

$$\alpha = softmax(w^\top G) \tag{2.3}$$

$$r_c = H\alpha^\top \tag{2.4}$$

where $H \in \mathbb{R}^{d_s \times T}$, w is a trained parameter vector. The dimension of w, α, r_c are d_s, T, d_s respectively.

2.4.3 Mention Representation

Averaging encoder: Given the entity mention $m_i = (w_p, \ldots, w_t)$ and its length $L = t - p + 1$, the averaging encoder computes the average word embedding of the words in m_i. Formally, the averaging representation r_a of the mention is computed as follows:

$$r_a = \frac{1}{L} \sum_{i=p}^{t} \mathbf{w}_i^d \tag{2.5}$$

This relatively simple method for composing the mention representation is motivated by it being less prone to overfitting (Shimaoka et al. 2017).

LSTM encoder: In order to capture more semantic information from the mentions, we add one token before and another after the target entity to the mention. The extended mention can be represented as $m_i^* = (w_{p-1}, w_p, \ldots, w_t, w_{t+1})$. The standard LSTM is applied to the mention sequence from left to right and produces the outputs $h_{p-1}, \ldots, h_{t+1}$. The last output h_{t+1} then serves as the LSTM representation r_l of the mention.

2.4.4 Optimization

We concatenate context representation and two mention representations together to form the overall feature representation of the input $R = [r_c, r_a, r_l]$. Then we use a softmax classifier to predict $\hat{y}_i$ from a discrete set of classes for a entity mention m and its context c with R as input:

$$\hat{p}(y|m,c) = \text{softmax}(WR + b) \tag{2.6}$$

$$\hat{y} = \arg\max_{y} \hat{p}(y|m,c) \tag{2.7}$$

where W can be treated as the learnt type embeddings and b is the bias.

The traditional cross-entropy loss function is represented as follows:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}(y_i|m_i,c_i)) + \lambda\|\Theta\|^2 \tag{2.8}$$

where y_i is the only element in $\mathcal{Y}_i^t$ and $(m_i, c_i, \mathcal{Y}_i) \in \mathcal{D}_{filtered}$. λ is an L2 regularization hyperparameter and Θ denotes all parameters of the considered model.

In order to handle data with *out-of-context noise* (in other words, with multiple labeled types) and take full advantage of them, we introduce a simple yet effective variant of the cross-entropy loss:

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log(\hat{p}(y_i^*|m_i,c_i)) + \lambda\|\Theta\|^2 \tag{2.9}$$

where $y_i^* = \arg\max_{y\in\mathcal{Y}_i^t} \hat{p}(y|m_i,c_i)$ and $(m_i, c_i, \mathcal{Y}_i) \in \mathcal{D}_{raw}$. With this loss function, we assume that the type with the highest probability among $\mathcal{Y}_i^t$ during training as the correct type. If there is only one element in $\mathcal{Y}_i^t$, this loss function is equivalent to the cross-entropy loss function. Wherever there are multiple elements, it can filter the less probable types based on the local context automatically.

2.4.5 Hierarchical Loss Normalization

Since the fine-grained types tend to form a forest of type hierarchies, it is unreasonable to treat every type equally. Intuitively, it is better to predict an ancestor type of the true type than some other unrelated type. For instance, if one example is labeled as **actor**, it is reasonable to predict its type as **person**. However, predicting other high level types like **location** or **politician** would be inappropriate. In other words, we want to penalize less the cases where types are related in our loss function. Based on the above idea, we adjust the estimated probability as follows:

$$p^*(\hat{y}|m,c) = p(\hat{y}|m,c) + \beta * \sum_{t \in \Gamma} p(t|m,c) \tag{2.10}$$

where Γ is the set of ancestor types along the *type-path* of $\hat{y}$, β is a hyperparameter to tune the penalty. Afterwards, we re-normalize it back to a probability distribution, which we denote this process as *hierarchical loss normalization*.

As discussed in Section 1, there exists *overly-specific noise* in the automatically labeled training sets which hurt the model performance severely. With *hierarchical loss normalization*, the model will get less penalty when it predicts the actual type for one example with *overly-specific noise*. Hence, it can alleviate the negative effect of *overly-specific noise* effectively. Generally, *hierarchical loss normalization* can make the model somewhat understand the given type hierarchy and learn to detect those overly-specific cases. During classification, it will make the models prefer generic types unless there is a strong indicator for a more specific type in the context.

2.4.6 Regularization

Dropout, proposed by Hinton et al. (2012), prevents co-adaptation of hidden units by randomly omitting feature detectors from the network during forward propagation. We employ both input and output dropout on LSTM layers. In addition, we constrain L2-norms for the weight vectors as shown in equation 2.8, 2.9 and use early stopping to decide when to stop training.

2.5 Experiments

2.5.1 Datasets

We evaluate the proposed model on two standard and publicly available datasets, provided in a pre-processed tokenized format by Shimaoka et al. (2017). Table 2.2 shows statistics about the benchmarks. The details are as follows:

	FIGER(GOLD)	Ontonotes
# types	113	89
# raw training mentions	2009898	253241
# raw testing mentions	563	8963
% filtered training mentions	64.46	73.13
% filtered testing mentions	88.28	94.00
Max hierarchy depth	2	3

Table 2.2: Statistics of the datasets

FIGER(GOLD): The training data consists of Wikipedia sentences and was automatically generated with distant supervision, by mapping Wikipedia identifiers in Wikipedia articles to Freebase. The test data, mainly consisting of sentences from news reports, was manually annotated as described by Ling and Weld (2012).

OntoNotes: The OntoNotes dataset consists of sentences from newswire documents present in the OntoNotes text corpus (Weischedel et al. 2013). DBpedia spotlight (Daiber et al. 2013) was used to automatically link entity mention in sentences to Freebase. For this corpus, manually annotated test data was shared by Dan Gillick et al. (2014).

Because the type hierarchy can be somewhat understood by our proposed model, the quality of the type hierarchy can also be a key factor to the performance of our model. We find that the type hierarchy for FIGER(GOLD) dataset following Freebase has some flaws. For example, **software** is not a subtype of **product** and **government** is not a subtype of **organization**. Following the proposed type hierarchy described by Ling and Weld 2012, we refine the Freebase-based type hierarchy. The process is a one-to-one mapping for types in the original dataset and we didn't add or drop any type or sentence in the original dataset. As a result, we can directly compare the results of our proposed model with or without this refinement.

2.5.2 Baselines

We compared the proposed model with state-of-the-art FETC systems [1]: (1) **Attentive** (Shimaoka et al. 2017); (2) **AFET** (Ren, He, Qu, L. Huang, et al. 2016); (3) **LNR+FIGER** (Ren, He, Qu, Voss, et al. 2016); (4) **AAA** (Abhishek, Anand, and Awekar 2017).

We compare these baselines with variants of our proposed model: (1) **NFETC(f)**: basic neural model trained on $\mathcal{D}_{filtered}$ (recall Section 2.4.4); (2) **NFETC-hier(f)**: neural model with hierarichcal loss normalization trained on $\mathcal{D}_{filtered}$. (3) **NFETC(r)**: neural model with proposed variant of cross-entropy loss trained on $\mathcal{D}_{raw}$; (4) **NFETC-hier(r)**: neural model with proposed variant of cross-entropy loss and hierarchical loss normalization trained on $\mathcal{D}_{raw}$.

2.5.3 Experimental Setup

For evaluation metrics, we adopt the same criteria as Ling and Weld (2012), that is, we evaluate the model performance by strict accuracy, loose macro, and loose micro F-scores. These measures are widely used in existing FETC systems (Abhishek, Anand, and Awekar 2017; Ren, He, Qu, L. Huang, et al. 2016; Ren, He, Qu, Voss, et al. 2016; Shimaoka et al. 2017).

We use pre-trained word embeddings that were not updated during training to help the model generalize to words not appearing in the training set. For this purpose, we used the freely available 300-dimensional cased word embedding trained on 840 billion tokens from the Common Crawl supplied by Pennington, Socher, and Manning (2014). For both datasets, we randomly sampled 10% of the test set as a development set, on which we do the hyperparameters tuning. The remaining 90% is used for final evaluation. We run each model with the well-tuned hyperparameter setting five times and report their average strict accuracy, macro F1 and micro F1 on the test set. The proposed model was implemented using the TensorFlow framework.

[1] The results of the baselines are all as reported in their corresponding papers.

2.5.4 Hyperparameter Setting

In this paper, we search different hyperparameter settings for FIGER(GOLD) and OntoNotes separately, considering the differences between the two datasets. All the hyperparameters were obtained by evaluating the model performance on the development set. The hyperparameters include learning rate lr for Adam Optimizer, size of word position embeddings (WPE) d_p, state size for LSTM layers d_s, input dropout keep probability p_i and output dropout keep probability p_o for LSTM layers [2], L2 regularization parameter λ and parameter to tune hierarchical loss normalization β. Values of these hyperparameters for the two datasets can be found in Table 2.3.

Parameter	FIGER(GOLD)	Ontonotes
lr	0.0002	0.0002
d_p	85	20
d_s	180	440
p_i	0.7	0.5
p_o	0.9	0.5
λ	0.0	0.0001
β	0.4	0.3

Table 2.3: Hyperparameter Settings

2.5.5 Performance comparison and analysis

	FIGER(GOLD)			OntoNotes		
Model	Strict Acc.	Macro F1	Micro F1	Strict Acc.	Macro F1	Micro F1
Attentive	59.68	78.97	75.36	51.74	70.98	64.91
AFET	53.3	69.3	66.4	55.1	71.1	64.7
LNR+FIGER	59.9	76.3	74.9	57.2	71.5	66.1
AAA	65.8	**81.2**	77.4	52.2	68.5	63.3
NFETC(f)	57.9 ± 1.3	78.4 ± 0.8	75.0 ± 0.7	54.4 ± 0.3	71.5 ± 0.4	64.9 ± 0.3
NFETC-hier(f)	68.0 ± 0.8	$\mathbf{81.4 \pm 0.8}$	77.9 ± 0.7	59.6 ± 0.2	$\mathbf{76.1 \pm 0.2}$	69.7 ± 0.2
NFETC(r)	56.2 ± 1.0	77.2 ± 0.9	74.3 ± 1.1	54.8 ± 0.4	71.8 ± 0.4	65.0 ± 0.4
NFETC-hier(r)	$\mathbf{68.9 \pm 0.6}$	$\mathbf{81.9 \pm 0.7}$	$\mathbf{79.0 \pm 0.7}$	$\mathbf{60.2 \pm 0.2}$	$\mathbf{76.4 \pm 0.1}$	$\mathbf{70.2 \pm 0.2}$

Table 2.4: Strict Accuracy, Macro F1 and Micro F1 for the models tested on the FIGER(GOLD) and OntoNotes datasets.

Table 2.4 compares our models with other state-of-the-art FETC systems on FIGER(GOLD) and OntoNotes. The proposed model performs better than

[2]Following TensorFlow terminology.

the existing FETC systems, consistently on both datasets. This indicates benefits of the proposed representation scheme, loss function and hierarchical loss normalization.

Discussion about *Out-of-context Noise*: For dataset FIGER(GOLD), the performance of our model with the proposed variant of cross-entropy loss trained on $\mathcal{D}_{raw}$ is significantly better than the basic neural model trained on $\mathcal{D}_{filtered}$, suggesting that the proposed variant of the cross-entropy loss function can make use of the data with *out-of-context noise* effectively. On the other hand, the improvement introduced by our proposed variant of cross-entropy loss is not as significant for the OntoNotes benchmark. This may be caused by the fact that OntoNotes is much smaller than FIGER(GOLD) and the proportion of examples without *out-of-context noise* is also higher, as shown in Table 2.2.

Investigations on *Overly-Specific Noise*: With hierarchical loss normalization, the performance of our models are consistently better no matter whether trained on $\mathcal{D}_{raw}$ or $\mathcal{D}_{filtered}$ on both datasets, demonstrating the effectiveness of this hierarchical loss normalization and showing that *overly-specific noise* has a potentially significant influence on the performance of FETC systems.

2.5.6 T-SNE Visualization of Type Embeddings

By visualizing the learned type embeddings (Figure 2.3), we can observe that the parent types are mixed with their subtypes and forms clear distinct clusters without hierarchical loss normalization. While the parent types tend to cluster together and the general pattern is more complicated with hierarchical loss normalization. It indicates that our model can learn rather subtle intricacies and correlations among types latent in the data with the help of hierarchical loss normalization, instead of sticking to a pre-defined hierarchy.

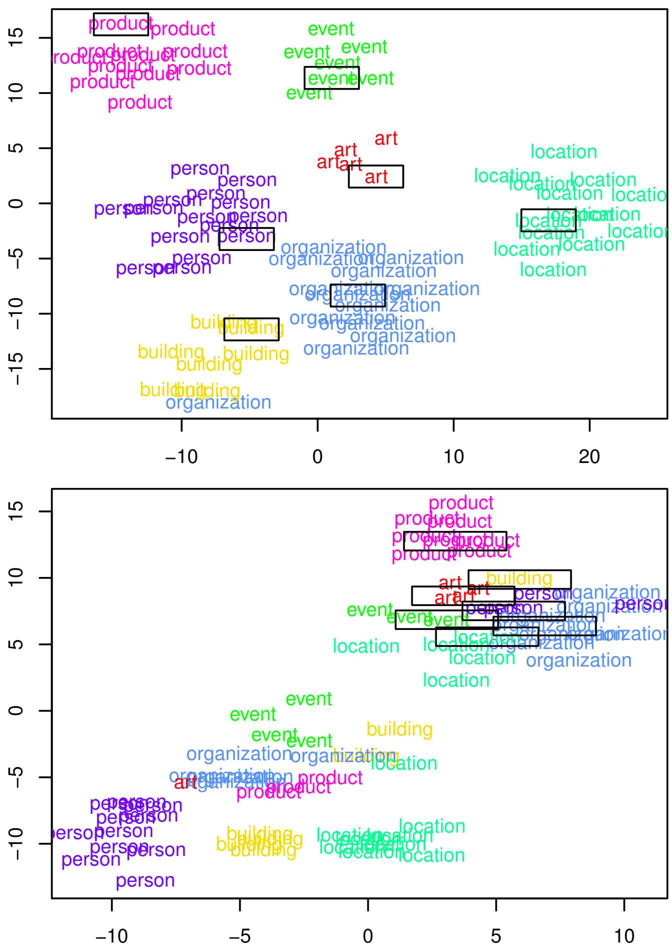

Figure 2.3: T-SNE visualization of the type embeddings learnt from FIGER(GOLD) dataset where subtypes share the same color as their parent type. The seven parent types are shown in the black boxes. The below sub-figure uses the hierarchical loss normalization, while the above not.

2.5.7 Error Analysis on FIGER(GOLD)

Since there are only 563 sentences for testing in FIGER(GOLD), we look into the predictions for all the test examples of all variants of our model. Table 2.5 shows 5 examples of test sentence. Without hierarchical loss normalization, our model will make too aggressive predictions for S1 with **Politician** and for S2 with **Software**. This kind of mistake is very common and can be effectively reduced by introducing hierarchical loss normalization leading to significant improvements on the model performance. Using the changed loss function to handle multi-label (noisy) training data can help the model distinguish

Test Sentence	Ground Truth
S1: ***Hopkins*** said four fellow elections is curious , considering the ...	**Person**
S2: ... for WiFi communications across all ***the SD cards.***	**Product**
S3: A handful of professors in the ***UW*** Department of Chemistry ...	**Educational Institution**
S4: Work needs to be done and, in ***Washington state***, ...	**Province**
S5: ***ASC*** Director Melvin Taing said that because the commission is ...	**Organization**

Table 2.5: Examples of test sentences in FIGER(GOLD) where the entity mentions are marked as bold italics.

ambiguous cases. For example, our model trained on $\mathcal{D}_{filtered}$ will misclassify S5 as **Title**, while the model trained on $\mathcal{D}_{raw}$ can make the correct prediction.

However, there are still some errors that can't be fixed with our model. For example, our model cannot make correct predictions for S3 and S4 due to the fact that our model doesn't know that ***UW*** is an abbreviation of *University of Washington* and ***Washington state*** is the name of a province. In addition, the influence of *overly-specific noise* can only be alleviated but not eliminated. Sometimes, our model will still make too aggressive or conservative predictions. Also, mixing up very ambiguous entity names is inevitable in this task.

2.6 Summary

In this chapter, we studied two kinds of noise, namely *out-of-context* noise and *overly-specific* noise, for noisy type labels and investigate their effects on FETC systems. We proposed a neural network based model which jointly learns representations for entity mentions and their context. A variant of cross-entropy loss function was used to handle *out-of-context* noise. Hierarchical loss normalization was introduced into our model to alleviate the effect of *overly-specific* noise. Experimental results on two publicly available datasets demonstrate that the proposed model is robust to these two kind of noise and outperforms previous state-of-the-art methods significantly.

Chapter 3

Neural Relation Extraction (NRE)

In this chapter, we first give an introduction to the task of RE in Section 3.1. In Section 3.2, we briefly review some important methods for RE. In Section 3.3, we give the necessary background and define the problem of RE formally. Then, we improve the existing neural architectures for RE by introducing multi-level attention mechanisms and outperform the state-of-the-art for the task in Section 3.4 and Section 3.5.

3.1 Introduction

Knowledge Bases (KBs) provide structured information about the world and are used in support of many important natural language processing applications such as semantic search and question answering. Building KBs is a non-trivial and never-ending task because, as the world changes, new knowledge needs to be harvested while old knowledge needs to be revised. This motivates the work on Relation Extraction (RE) task, whose goal is to assign a KB relation to a *phrase* connecting a pair of entities, which in turn can be used for updating the KB.

Most existing supervised RE systems require a large amount of labelled relation-specific training data, which is very time-consuming and labor intensive. Mintz et al. (2009) proposes distant supervision to automatically generate training data via aligning KBs and texts. Although distant supervision is an effective strategy to automatically label training data, it always suffers from wrong labelling problem. Hence, Riedel, Yao, and McCallum (2010), Hoffmann et al. (2011) and Surdeanu et al. (2012) adopt multi-instance learning to alleviate the wrong labelling problem. The main weakness of these conventional methods is that most features are explicitly derived from NLP tools such as POS tagging and the errors generated by NLP tools will propagate in these methods. Recently, neural models have shown superior performance over approaches using hand-crafted features in the task of RE. We'll review some important methods in this category in the next section.

3.2 Methods for Neural Relation Extraction

Some recent works (Santos, Xiang, and B. Zhou 2015; D. Zeng, K. Liu, Lai, et al. 2014; P. Zhou et al. 2016) attempt to use deep neural networks in relation classification without handcrafted features. These methods build classifier based on sentence-level annotated data, which cannot be applied in large-scale KBs due to the lack of human-annotated training data. Therefore, D. Zeng, K. Liu, Y. Chen, et al. (2015) incorporate multi-instance learning with neural network model, which can build a relation extractor based on distant supervision data. Although the method achieves significant improvement in relation extraction, it is still far from satisfactory. The method assumes that at least one sentence that mentions these two entities will express their relation, and only selects the most likely sentence for each entity pair in training and prediction. It's apparent that the method will lose a large amount of rich information containing in neglected sentences.

Y. Lin, Shen, et al. (2016) propose a sentence-level attention-based convolutional neural network for distant supervised relation extraction. They employ a CNN to embed the semantics of sentences. Afterwards, to utilize all informative sentences, they represent the relation as semantic composition of sentence embeddings. To address the wrong labelling problem, they build sentence-level attention over multiple instances, which is expected to dynamically reduce the weights of those noisy instances.

In the past few years, various neural models are proposed to improve the performance by different techniques, including adversarial learning (Y. Wu, Bamman, and Russell 2017), deep residual learning (Y. Y. Huang and W. Y. Wang 2017), incorporating relation path (W. Zeng et al. 2016), leveraging information from other sources (L. Liu et al. 2017), to name a few.

Our Approach. Based on the neural model with selective attention over sentences proposed by Y. Lin, Shen, et al. (2016), we replace the sentence encoder with bi-directional long short term memory networks with word-level attention (P. Zhou et al. 2016) as compared to the original convolutional neural networks. Experimental results show that our model can consistently outperform previous methods.

In the next section, we first give the necessary background and define the problem of RE formally. The details of our proposed model is described in Section 3.4.

3.3 Background and Problem

The task of RE is to predict a KB relation that holds for a pair of entities given several text mentions, or *NA* if no such relation exists. The input is a knowledge base Ψ with relation set $\mathcal{R}_\Psi$, a target relation set $\mathcal{R}$, $\mathcal{R} \subseteq \mathcal{R}_\Psi$, and an automatically labeled training dataset $\mathcal{D}$ obtained via distant supervision. (For example, in our setting $\mathcal{R}_\Psi$ is all 23K Freebase relations while $\mathcal{R}$ is a smaller subset of relations of interest). The output is a relation in $\mathcal{R} \cup \{NA\}$ for a test sentence. The catch-all relation *NA* applies when there is no relation in $\mathcal{R}$ that holds over h, t (given the set of sentences).

3.3.1 Knowledge Base and Distant Supervision

As customary, we denote a KB Ψ with relation scheme $\mathcal{R}_\Psi$ as a set of *triples* $\mathcal{T}_\Psi = \{(h, r, t) \in \mathcal{E}_\Psi \times \mathcal{R}_\Psi \times \mathcal{E}_\Psi\}$, where $\mathcal{E}_\Psi$ is the set of entities of interest. Distant supervision exploits the KB to automatically annotate sentences in a corpus containing mentions of entities with the relations they participate in. Formally, a labeled dataset for relation extraction consists of fact triples $\{(h_i, r_i, t_i)\}_{i=1}^{N}$ and a multi-set of extracted sentences for each triple $\{\mathcal{S}_i\}_{i=1}^{N}$, such that each sentence $s \in \mathcal{S}_i$ contains mentions to the head entity h_i and the tail entity t_i.

3.3.2 Problem Statement

Given an entity pair (h, t) and a set of sentences $\mathcal{S}$ with mentions to them, the RE task is to estimate the probability of each relation in $\mathcal{R} \cup \{NA\}$. Formally, for each relation r, we want to predict $P(r \mid h, t, \mathcal{S})$.

In practice, the input set of sentences $\mathcal{S}$ can have arbitrary size. For the sake of computational efficiency, we normalize the set size to a fixed number T by splitting large sets and oversampling small ones. We also restrict the length of each sentence in the set by a constant L by truncating long sentences and padding short ones.

3.4 Bi-LSTM with Multi-Level Attention Mechanisms

3.4.1 Input Representation

We represent each word in each sentence as a real-valued vector capturing lexical and semantic features pertaining to relation extraction. Given a word embedding matrix W^{wrd} of size $d_w \times |V|$, we map every word w_i in s to a column vector $\mathbf{w}_i^d \in \mathbb{R}^{d_w}$ where V is the input vocabulary and d_w is the size of word embedding. Further, we incorporate word position embeddings to capture the distances between each word to the entities mentioned in the text. Similarly to D. Zeng, K. Liu, Lai, et al. (2014), each relative distance is mapped to a randomly initialized position vector in $\mathbb{R}^{d_p}$, where d_p is the size of position embedding. For word w_i, we obtain the position vector $\mathbf{w}_i^p$. Finally, the overall embedding of word w_i is $\mathbf{w}_i^E = [(\mathbf{w}_i^d)^\top, (\mathbf{w}_i^p)^\top]^\top$.

3.4.2 Sentence Encoder

For a sentence s_i, we want to apply a non-linear transformation to the vector representation of s_i to derive a feature vector $z_i = f(s_i; \theta)$ given a set of parameters θ. In this paper, we adopt bidirectional LSTM with d_s hidden units as $f(s_i; \theta)$. The network contains two sub-networks for the forward pass and the backward pass respectively. Here, we use element-wise sum to combine the forward and backward pass outputs. The output of the i-th word is shown in the following equation:

$$z_i = [\overrightarrow{z_i} \oplus \overleftarrow{z_i}] \tag{3.1}$$

3.4.3 Multi-level Attention Mechanisms

We employ attention mechanisms at both word-level and sentence-level to allow the model to softly select the most informative words and sentences during training (Y. Lin, Shen, et al. 2016; P. Zhou et al. 2016).

Word-level attention. Let $H_w = [z_1, \dots, z_L]$ be the matrix with output vectors produced at the LSTM layer; the representation of sentence $\mathbf{s}$ (truncated to length L) is:

$$G_w = \tanh(H_w) \tag{3.2}$$

$$\alpha_w = \text{softmax}(w_w^\top G_w) \tag{3.3}$$

$$\mathbf{s} = H\alpha_w^\top \tag{3.4}$$

where $H_w \in \mathbb{R}^{d_s \times L}$, $w_w \in \mathbb{R}^{d_s \times 1}$ $\alpha_w^\top \in \mathbb{R}^{L \times 1}$, $\mathbf{s} \in \mathbb{R}^{d_s \times 1}$ and w_w is a trained parameter vector.

Sentence-level attention. Similarly, the language representation $\mathbf{s}_L$ can be obtained from the matrix of sentence representations $H_s = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_T]$ as follows:

$$G_s = \tanh(H_s) \tag{3.5}$$

$$\alpha_s = \text{softmax}(w_s^\top G_s) \tag{3.6}$$

$$\mathbf{s}_L = H_s\alpha_s^\top \tag{3.7}$$

where $H_s \in \mathbb{R}^{d_s \times T}$, $w_w \in \mathbb{R}^{d_s \times 1}$ $\alpha_w^\top \in \mathbb{R}^{T \times 1}$, $\mathbf{s}_L \in \mathbb{R}^{d_s \times 1}$ and w_s is a trained parameter vector.

With the language representation $\mathbf{s}_L$, the conditional probability $p(r|\mathcal{S};\Theta^{(L)})$ is computed through a softmax layer as follows:

$$p(r|\mathcal{S};\Theta^{(L)}) = \text{softmax}(W^{(L)}\mathbf{s}_L + b^{(L)}) \tag{3.8}$$

where $b^{(L)} \in \mathbb{R}^{K \times 1}$ is a bias vector, $W^{(L)} \in \mathbb{R}^{K \times d_s}$ is the representation matrix of relations, $K = |\mathcal{R} \cup \{NA\}|$ and $\Theta^{(L)}$ is the parameters of the model to learn language representation.

3.4.4 Optimization and Implementation Details

Here we introduce the learning and optimization details of our proposed model. We define the objective using cross-entropy as follows:

$$\mathcal{J}_L = -\frac{1}{N}\sum_{i=1}^{N} \log p(r_i|\mathcal{S}_i;\Theta^{(L)}) + \lambda\|\Theta^{(L)}\|_2^2 \tag{3.9}$$

where N denotes the size of the training set. To solve the optimization problem, we adopt the stochastic gradient descent with mini-batches and Adam (Kingma and Ba 2014) to update $\Theta^{(L)}$. In the implementation, we employ both input on output dropout (Hinton et al. 2012) on LSTM layers to prevent overfitting. In addition, we constrain L2-norms for the weight vectors as shown in Eq. 4.8.

3.5 Experiments

We now report on an experimental evaluation of our framework against the state-of-the-art for the RE task.

3.5.1 Datasets

We evaluate our model on the widely used **NYT** dataset, developed by Riedel, Yao, and McCallum (2010) by aligning Freebase relations mentioned in the New York Times Corpus. The Freebase relations are divided into two parts,

learning rate	lr	5×10^{-4}
size of word position embedding	d_p	25
state size for LSTM layers	d_s	320
input dropout keep probability	p_i	0.9
output dropout keep probability	p_o	0.7
L2 regularization parameter	λ	0.0003

Table 3.1: Hyperparameter setting

one for training and one for testing. Articles from years 2005-2006 are used for training while articles from 2007 are used for testing.

3.5.2 Experimental Settings

Baselines: We choose five baseline approaches following Y. Lin, Shen, et al. (2016) [1]: Three feature-based methods, **Mintz** (Mintz et al. 2009), **MultiR** (Hoffmann et al. 2011), **MIML** (Surdeanu et al. 2012), and two convolutional neural network based methods, **CNN+ATT** and **PCNN+ATT** (Y. Lin, Shen, et al. 2016).

Evaluation Protocol: Following previous work (Mintz et al. 2009), we evaluate our model using held-out evaluation which approximately measures the precision without time-consuming manual evaluation. We report both Precision/Recall curves and Precision@N (P@N) in our experiments, ignoring the probability predicted for the *NA* relation. Moreover, to evaluate each sentence in the test set as in previous methods, we append T copies of each sentence into $\mathcal{S}$ for each testing sample.

Word Embeddings: In this paper, we use the freely available 300-dimensional pre-trained word embeddings distributed by Pennington, Socher, and Manning (2014) to help the model generalize to words not appearing in the training set.

Hyperparameter Settings: For hyperparameter tuning, we randomly sample 10% of the training set as a development set. All the hyperparameters are obtained by evaluating the model on the development set. With the well-tuned hyperparameter setting, we run each model five times on the whole training set and report the average P@N. For Precision/Recall curves, we just select

[1]The results of baseline approaches are obtained from Y. Lin, Shen, et al. (2016)

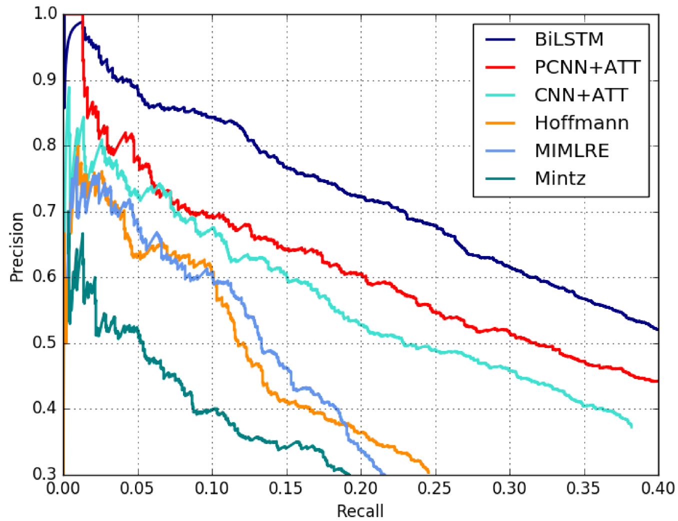

Figure 3.1: The Precision/Recall curves of baseline approaches and our proposed model named as BiLSTM.

the results from the first run of each model. For training, we set the iteration number over all the training data as 30. Values of the hyperparameters used in the experiments can be found in Table 4.3.

3.5.3 Performance Comparison and Analysis

Figure 3.1 shows the Precision/Recall curves for all baseline approaches and the variant of our model **BiLSTM**. As one can see, **BiLSTM** significantly outperforms all baseline approaches over the entire range of recall. Possible reasons for the improvements can be attributed to the usage of better word embeddings, different neural architectures and improved attention mechanisms.

3.6 Summary

In this chapter, we develop BiLSTM with multi-level attention mechanisms. Our model can make full use of all informative words and sentences and alleviate the wrong labelling problem for distant supervised relation extraction. In experiment, we evaluate our model on relation extraction task. The experimental results show that our model significantly and consistently outperforms state-of-the-art feature-based methods and neural network methods.

Chapter 4

Incorporating Encoded Knowledge Information

In this chapter, we first give an introduction to the efforts we've made to incorporate knowledge base embeddings to facilitate relation extraction in Section 4.1. In Section 4.2, we review the literature about combining text and knowledge information. In Section 4.3, we investigate on the effectiveness of knowledge base embedding models on the task of relation prediction and extraction. Then, we describe and evaluate a novel neural framework for jointly learning representations for RE and KBE tasks that uses a cross-entropy loss function to ensure both representations are learned together, resulting in significant improvements over the current state-of-the-art for relation extraction in Section 4.4 and 4.5.

4.1 Introduction

Knowledge Bases (KBs) provide structured information about the world and are used in support of many important natural language processing applications such as semantic search and question answering. Building KBs is a non-trivial and never-ending task because, as the world changes, new knowledge needs to be harvested while old knowledge needs to be revised. This motivates the work on the Relation Extraction (RE) task, whose goal is to assign a KB relation to a *phrase* connecting a pair of entities, which in turn can be used for updating the KB. State-of-the-art techniques for this task build on neural methods leveraging distant (a.k.a. weak) supervision (Mintz et al. 2009) on large-scale corpora for training.

A related task is that of Knowledge Base Embedding (KBE), which is concerned with representing KB entities and relations in a vector space for link prediction. Weston et al. (2013) was the first to show that *combining* predictions from RE and KBE models, trained in isolation, improves the effectiveness on the RE task. However, the way in which they combine RE and KBE predictions is rather naive (namely, by adding those scores). In Section 4.3, extensive experiments have been conducted to evaluate the effectiveness of the existing knowledge base embedding models for relation prediction and for relation extraction on a wide range of benchmarks. The results demonstrate that knowledge base embedding models are generally effective for relation prediction but unable to give improvements for the state-of-the-art neural relation extraction model with the existing strategies, while pointing limitations of existing methods.

In the next section, we will first review some existing works which attempt to combine text and knowledge information. To the best of our knowledge, there have been no systematic attempts to further unify RE and KBE, particularly during model *training*. We seek to close this gap in this chapter.

4.2 Related Work

Some works attempt to combine text information and knowledge base to facilitate knowledge base embedding. Z. Wang et al. (2014a) combined text information and knowledge base by embedding entities and the words in their names in the same vector space. Neelakantan, B. Roth, and Mc-Callum (2015) learn the co-occurrence based textual relation representations to help with knowledge base completion. Toutanova, D. Chen, et al. (2015) train continuous representations of knowledge base and textual relations jointly, which allows for deeper interactions between the sources of information and achieved significant improvement. The success of this joint model on knowledge base embedding inspires us to employ a similar idea on relation extraction.

There are also some works trying to combine text information and knowledge base to facilitate relation extraction, but not leveraging the knowledge base embedding models. Riedel, Yao, McCallum, and Marlin (2013) propose universal schema to transmit information between relations of KGs and textual patterns via their common entity pairs. Verga, Belanger, et al. (2015) further incorporate neural networks to relax constraints imposed by entity pairs in universal schema. Ren, Z. Wu, et al. (2017) extract typed entities and relations jointly by learning embeddings from text corpora and knowledge bases.

Despite rapid progress in both RE and KBE, limited efforts have gone into connecting these two areas to improve relation extraction. Weston et al. (2013) describe a method for doing so, although in their work the two representations are trained independently with different loss functions and only combined at inference time. Nevertheless, the authors show that taking the KBE score into account improves on the predictions made by their RE model alone. However, advances in RE and KBE models since then have reduced the net gains achievable with simple combination schemes described in this work. This claim is validated by the experimental results described in the next section.

4.3 Investigations on Knowledge Base Embedding for Relation Prediction and Extraction

4.3.1 Introduction

Representing information about real-world entities and their relations in structured knowledge bases (KBs) enables various applications such as structured search, factual question answering, and intelligent virtual assistants. A major challenge for using discrete representation of knowledge base is the lack of capability of accessing the similarities among different entities and relations. Knowledge base embedding (KBE) techniques have been proposed in recent years to deal with this issue. The main idea is to represent the entities and relations in a vector space, and one can use machine learning technique to learn the continuous representation of the knowledge base in the latent space.

Motivated by the fact that the existing KBs suffer from the problem of low coverage, considerable effort has been committed in automatically deriving new facts to extend a manually built knowledge base with information from the existing knowledge base. As a result, most of KBE models focus on predicting missing entities or accessing the plausibility of a missing triple instead of predicting missing relations. Another reason for such choice is that predicting relations is relatively easier than predicting entities because the number of relations is significantly smaller than the number of entities. Moreover, existing KBE models reach nearly perfect results for relation prediction on some benchmark datasets (Y. Lin, Z. Liu, Luan, et al. 2015).

To the best of our knowledge, the literature lacks a comprehensive effort in validating the *effectiveness* of state-of-the-art KBE models on the task of relation prediction and extraction. To fill this gap, we report extensive experiments conducted with different KBE models and different datasets for relation prediction. We choose three KBE models considering their simplicity, effectiveness, and flexibility: `TransE`, `DistMult` and `ComplEx`. We test on four established benchmarks as well as on a new one we developed, covering different levels of text complexity and corpus size.

Weston et al. (2013) were the first to combine representation learnt by KBE models and representation learned from textual mentions for relation prediction. The two representations were trained independently of each other, and were only combined with a simple strategy at inference time. Since they got remarkable improvement on performance of relation extraction, we expect that more improvement can also be obtained with the recent development in areas of both KBE and relation extraction. Following the same strategy, we use the chosen KBE models to help the state-of-art neural models for relation extraction. In addition, various combining strategies have been tried in order to squeeze the most improvement from KBE models.

The experimental results reported here show that the chosen KBE models are enough to achieve satisfactory performance on all of the datasets we studied for relation prediction. However, the improvements we can squeeze from KBE models are negligible for relation extraction which goes against our expectation. This observation indicates that the strategies that combine representations at inference time is not effective anymore due to the recent development in areas of both KBE and relation extraction. New methods should be proposed to make advantage of KBE to facilitate relation extraction.

4.3.2 Datasets

To evaluate KBE models for relation prediction, we use four common knowledge base completion datasets from the literature and introduce a new one. WN18 (Bordes et al. 2013) is a subset of WordNet which consists of 18 relations and 40,943 entities. WN18RR is a subset of WN18 introduced by Dettmers et al. (2017) which removes and dramatically increases the difficulty of reasoning. FB15k (Bordes et al. 2013) is a subset of Freebase which contains about 15k entities with 1,345 different relations. Likewise, FB15k-237 is a subset of FB15k introduced by Toutanova and D. Chen (2015). FB15k-237 removed redundant relations in FB15k and greatly reduced the number of relations.

To investigate the effectiveness of KBE models to facilitate relation extraction, we use the New York Times corpus (NYT) released by Riedel, Yao, and McCallum (2010) as training and testing data. A new dataset, namely FB3M, is introduced which is also a subset of Freebase restricted to the top 3 million entities - where top is defined as the ones with the largest number of relations to other entities. This dataset uses a large amount of entities and all possible relationships in Freebase. Hence, it covers most entities and all the relations to be predicted in the NYT dataset. Following Weston et al. (2013), we removed all the entity pairs present in the NYT test set from this dataset and translate the deprecated relationships into their new variants. This ensures that we cannot just memorize the true relations for an entity pair - we have to learn to generalize from other entities and relations. As the NYT dataset was built on an earlier version of Freebase we also had to translate the deprecated relationships into their new variants.

Model	TransE		DistMult		ComplEx	
Dataset	Filter	Raw	Filter	Raw	Filter	Raw
WN18	**0.971**	**0.969**	0.623	0.622	**0.991**	**0.989**
FB15k	0.883	0.773	0.695	0.644	**0.971**	**0.840**
WN18RR	0.843	0.842	0.871	0.866	**0.894**	**0.893**
FB15k-237	**0.955**	**0.950**	0.926	0.921	**0.956**	**0.950**
FB3M	0.475	0.464	0.620	0.607	**0.683**	**0.639**

Table 4.1: **MRR** measures on relation prediction.

Model	TransE		DistMult		ComplEx	
Dataset	Filter	Raw	Filter	Raw	Filter	Raw
WN18	**0.956**	**0.952**	0.256	0.256	**0.987**	**0.983**
FB15k	0.829	0.650	0.463	0.408	**0.950**	**0.726**
WN18RR	0.735	0.734	0.810	0.802	**0.813**	**0.813**
FB15k-237	**0.930**	**0.921**	0.880	0.871	**0.933**	**0.922**
FB3M	0.364	0.347	0.439	0.432	**0.460**	**0.410**

Table 4.2: **Hits@1** measures on relation prediction.

4.3.3 Relation Prediction

Relation prediction aims to predict relations given two entities. For each testing triple with missing relation, models are asked to compute the scores for all candidate entities and rank them in descending order.

Following Y. Lin, Z. Liu, Luan, et al. (2015), we use two measures as our evaluation metrics: the mean reciprocal of correct relation ranks (MRR) and the proportion of valid relations ranked in top-1(Hits@1). For each metric, we follow evaluation regimes "Raw" and "Filter" as described by Bordes et al. (2013).

Evaluation results of relation prediction are shown in Table 4.1 and 4.2. From there we observe that: (1) Generally, KBE models are doing well in the task of relation prediction. It indicates that relation information between entities can be captured by the existing KBE models without any specific modifications to adapt this task; (2) `ComplEx` achieves the best performance on all datasets. In addition, it significantly and consistently outperforms `DistMult` which matches the observations for the task of link prediction; (2) Surprisingly, `TransE` has competitive performance with `ComplEx` on some datasets which disagrees with the observations for the task of link prediction. It indicates that there exists intrinsic difference between the task of link prediction and relation prediction.

4.3.4 Facilitating Relation Extraction

Relation extraction from text aims at extracting relational facts from plain text to enrich existing KBs. Recent works regard large-scale KBs as source for distant supervision to annotate sentences as training instances and build relation classifiers using neural models (Y. Lin, Shen, et al. 2016; Y. Wu, Bamman, and Russell 2017; D. Zeng, K. Liu, Y. Chen, et al. 2015). All these methods reason new facts only based on plain text. In this task, we explore the effectiveness of KBE models to facilitate relation extraction from text.

In the experiments, we implemented the RNN-based model presented in Y. Wu, Bamman, and Russell 2017. We combine the ranking scores from the neural model with those from KBE to rank testing triples, and generate precision-recall curves for both `TransE` and `ComplEx`. Formally, for each pair of entities (s, o) that appear in the test set, all the corresponding sentences S in the test set are collected to form a set $\mathcal{S}$ and a prediction is performed with

$$\hat{r}_{s,o} = \underset{r \in \mathcal{R}}{\operatorname{argmax}}\ S_{RE}(r|\mathcal{S}) \tag{4.1}$$

where $S_{RE}(r|\mathcal{S})$ indicates the plausibility of relation r given the sentences set $\mathcal{S}$ predicted by the neural model. The predicted relation can either be a valid relation or NA - a marker that means there is no relation between s and o (NA is added to $\mathcal{R}$ during training and is treated like other relations). If $\hat{r}_{s,o}$ is a relation, a composite score is defined:

$$S(s, \hat{r}_{s,o}, o) = \alpha S_{RE}(\hat{r}_{s,o}|\mathcal{S}) + (1-\alpha) f_{\hat{r}_{s,o}}(s, o) \tag{4.2}$$

where $\alpha \in (0, 1]$ is a hyper-parameter to tune the balance between the text information and the KB. That is, only the top scoring non-NA predictions are re-scored. Hence, our final composite model favors predictions that agree with both the text information and the KB. If $\hat{r}_{s,o}$ is NA, the score is unchanged. If $\alpha = 1$, it's the same as the neural relation extraction model; if $\alpha = 0.5$, it follows the same combining strategy described in Weston et al. 2013.

The evaluation curves for `TransE` and `ComplEx` are shown in Figure 4.1 and 4.2. We can see that the improvement is negligible when $\alpha = 0.9$ and the combined models perform consistently worse than the original neural model with smaller α.

We also tried other strategies for combining the models: (1) use geometric average, harmonic average instead of weighted average; (2) apply the average operation on all non-NA predictions or all predictions instead of just the top non-NA predictions; (3) transform the scores into a probability distribution by applying a *softmax* operation. However, all these changes make no improvement or sometimes hurt the performance.

The experimental results indicate that the existing strategies which perform the combining at inference time is not effective any more due to the recent development in areas of both KBE and relation extraction. New methods should be proposed to make better advantage of KBE to facilitate relation extraction.

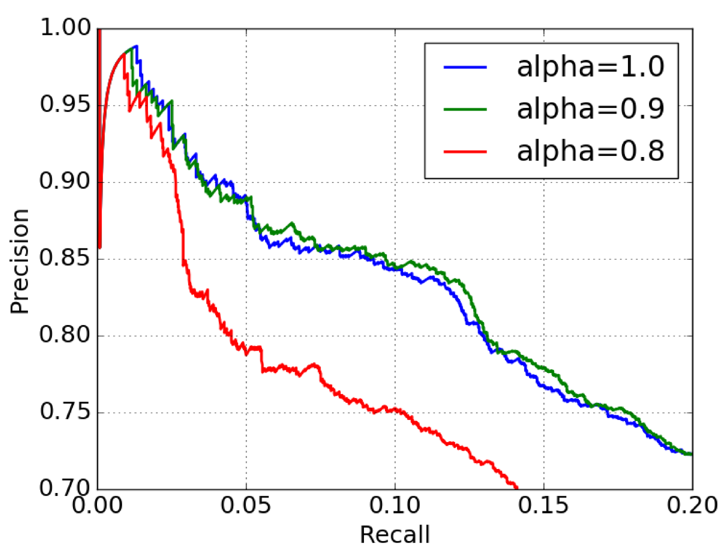

Figure 4.1: Precision-recall curves of **TransE** with different alpha.

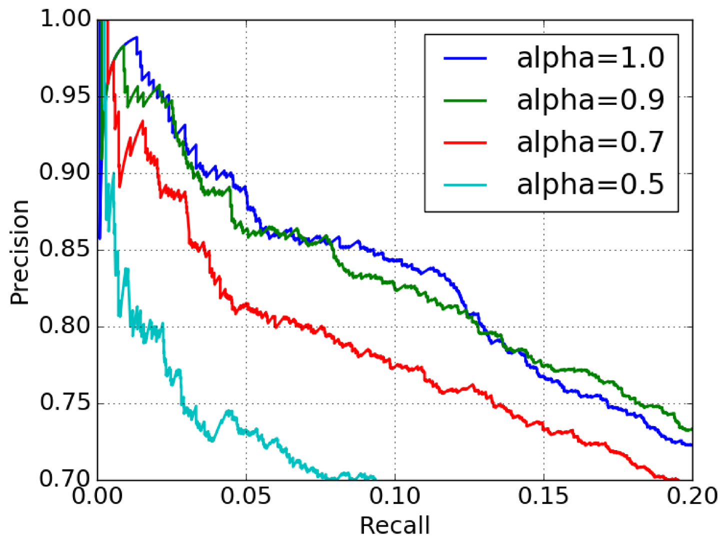

Figure 4.2: Precision-recall curves of **ComplEx** with different alpha.

4.4 Connecting Lanugage and Knowledge with Heterogeneous Representations for Neural Relation Extraction

We now go over the details of our framework outlined in Figure 4.3 for unifying the learning of the language and the knowledge representations used for relation extraction. In a nutshell, we use LSTM with attention mechanisms for language representation and we follow the approach of Trouillon et al. (2016) for knowledge encoding since it's the most effective model according to Section 4.3.

4.4.1 Problem Formulation and Language Representation

The problem formulation is the same as what's described in Section 3.3. We learn the language representation based on the input set of sentences for each entity pair with the neural architecture illustrated in Section 3.4.

With the language representation $\mathbf{s}_L$, the conditional probability $p(r|\mathcal{S};\Theta^{(L)})$

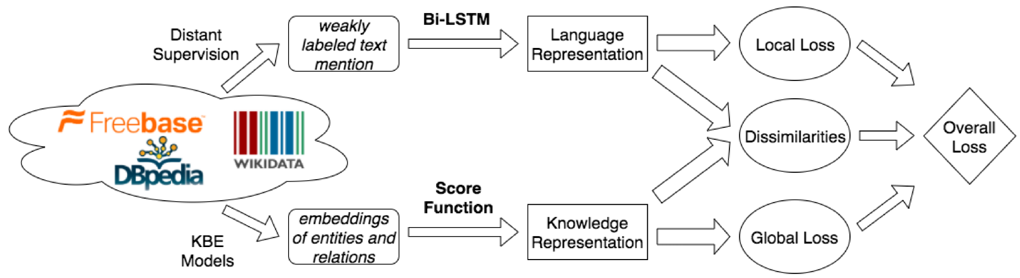

Figure 4.3: Workflow of the proposed framework.

is computed through a softmax layer as follows:

$$p(r|\mathcal{S};\Theta^{(L)}) = \text{softmax}(W^{(L)}\mathbf{s}_L + b^{(L)}) \tag{4.3}$$

where $b^{(L)} \in \mathbb{R}^{K\times 1}$ is a bias vector, $W^{(L)} \in \mathbb{R}^{K\times d_s}$ is the representation matrix of relations, $K = |\mathcal{R} \cup \{NA\}|$ and $\Theta^{(L)}$ is the parameters of the model to learn language representation.

4.4.2 Knowledge Representation

Large-scale KBs bring prior knowledge about entities that may complement the training done on weakly labeled text mentions. We learn the knowledge representation for entities and relations in the KB following the approach of Trouillon et al. (2016), which scales well to large datasets while consistently outperforming previous approaches.With the set of (all) relations $\mathcal{R}_\Psi$ and entities $\mathcal{E}_\Psi$ in the KB Ψ, the model aims to recover the latent matrix of scores $\mathbf{X}^{(r)}$ for each relation $r \in \mathcal{R}_\Psi$.

Given two entities $h, t \in \mathcal{E}_\Psi$, the log-odd of the probability with which the fact (h, r, t) holds is:

$$p(\mathbf{Y}_{ht}^{(r)} = 1) = \sigma(\mathbf{X}_{ht}^{(r)}) = \sigma(\phi(h, r, t)) \tag{4.4}$$

where ϕ is a scoring function that is typically based on a factorization of the observed relations.

The goal here is to learn representations of the entities and relations capable of predicting probabilities of $\mathbf{Y}_{h't'}^{(r')}$ being true for unobserved fact $(h', r', t') \notin \Omega$. From a set of candidate facts $\Omega \subset \mathcal{E}_\Psi \times \mathcal{R}_\Psi \times \mathcal{E}_\Psi$, the training set $\{\mathbf{Y}_{ht}^{(r)}\}_{(h,r,t)\in\Omega}$

is produced by labeling facts (-1,1) depending on whether or not they belong to Ψ.

Furthermore, the choice of scoring function $\phi(h, r, t)$ used to predict the entries of the tensor $\mathbf{X}$ determines the model. Following Trouillon et al. (2016), we use the following scoring function:

$$\begin{aligned} \phi(h, r, t) &= \mathrm{Re}(\langle e_h, w_r, \bar{e}_t \rangle) \\ &= \mathrm{Re}(\sum_{k=1}^{d_k} e_{hk} w_{rk} e_{tk}) \\ &= \langle \mathrm{Re}(e_h), \mathrm{Re}(w_r), \mathrm{Re}(e_t) \rangle \\ &\quad + \langle \mathrm{Im}(e_h), \mathrm{Re}(w_r), \mathrm{Im}(e_t) \rangle \\ &\quad + \langle \mathrm{Re}(e_h), \mathrm{Im}(w_r), \mathrm{Im}(e_t) \rangle \\ &\quad - \langle \mathrm{Im}(e_h), \mathrm{Im}(w_r), \mathrm{Re}(e_t) \rangle \end{aligned} \tag{4.5}$$

where d_k is the representation size, $w_r \in \mathbb{C}^{d_k}$ is the relation representation and $e_h, e_t \in \mathbb{C}^{d_k}$ are the entity representations for the head and tail entity respectively. In Equation 4.5, $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ stand for the real and imaginary parts of each complex number. One advantage of this scoring function is that the composition of complex valued representations can handle different binary relations, including symmetric and antisymmetric ones commonly found in KBs.

Following the training procedure in Trouillon et al. (2016), we can get the knowledge representations $e_h, w_r, e_t \in \mathbb{C}^{d_k}$ by minimizing the negative log-likelihood of the logistic model:

$$\sum_{(h,r,t) \in \Omega} \log(1 + \exp(-\mathbf{Y}_{ht}^{(r)} \phi(h, r, t))) \tag{4.6}$$

With the knowledge representations and the scoring function defined by Eq. 4.5, we can obtain the conditional probability $p(r|(h, t); \Theta^{(G)})$ for each relation r:

$$p(r|(h, t); \Theta^{(G)}) = \frac{e^{\phi(h,r,t)}}{\sum_{r' \in \mathcal{R} \cup \{NA\}} e^{\phi(h,r',t)}} \tag{4.7}$$

where $\Theta^{(G)}$ corresponds to the knowledge representations $e_h, w_r, e_t \in \mathbb{C}^{d_k}$. Since $NA \notin \mathcal{R}_\Psi$, we use a randomized complex vector as w_{NA}.

4.4.3 Connecting Heterogeneous Representations

As stated, this chapter seeks an elegant way of connecting language and knowledge representations for the RE task. In order to achieve that, we use separate loss functions (recall Figure 4.3) to guide the language and knowledge representation learning and a third loss function that ties the predictions of these models thus nudging the parameters towards agreement.

We found that better results were achieved if we started from stable knowledge representations. In other words, we first train knowledge representations e_h, w_r, e_t (as in the previous section) on the whole KB independently and then use them as the initialization point for the *joint* learning of the final knowledge representations with the language representation.

Loss Functions. The cross-entropy loss based on language representation (recall Eq. 4.3) is defined as:

$$\mathcal{J}_L = -\frac{1}{N}\sum_{i=1}^{N} \log p(r_i|\mathcal{S}_i; \Theta^{(L)}) \tag{4.8}$$

where N denotes the size of the training set. Since the language representation is learned from local context, the subscript L in $\mathcal{J}_L$ means local.

Then the cross-entropy loss based on knowledge representations (recall Eq. 4.7) can be defined as:

$$\mathcal{J}_G = -\frac{1}{N}\sum_{i=1}^{N} \log p(r_i|(h_i, t_i); \Theta^{(G)}) \tag{4.9}$$

Since the knowledge representation is learned from the whole KB, the subscript G in $\mathcal{J}_G$ means global.

Finally, we use a cross-entropy loss to measure the dissimilarity between two distributions, thus connecting them, and formulate model learning as minimizing $\mathcal{J}_D$:

$$\mathcal{J}_D = -\frac{1}{N}\sum_{i=1}^{N} \log p(r_i^*|\mathcal{S}_i; \Theta^{(L)}) \tag{4.10}$$

where $r_i^* = \arg\max_{r\in\mathcal{R}\cup\{NA\}} p(r|(h_i, t_i); \Theta^{(G)})$.

We also tried to use KL-divergence as $\mathcal{J}_D$ but the cross-entropy loss generally performed better.

4.4.4 Model Learning

Based on Eq. 4.8, 4.9, 4.10, we form the joint optimization problem for model parameters as

$$\min_{\Theta} \mathcal{J} = \mathcal{J}_L + \mathcal{J}_G + \mathcal{J}_D + \lambda \|\Theta\|_2^2 \tag{4.11}$$

where $\Theta = \Theta^{(L)} \cup \Theta^{(G)}$ is all the parameters of the considered model.

Collectively optimizing Eq. 4.11 allows the heterogeneous representations enhance each other. The language representation can leverage prior knowledge existing in the whole KB but not in the training dataset $\mathcal{D}$. Also, the knowledge representations can be refined with the text information related to the facts.

In order to solve the joint optimization problem in Eq. 4.11, we adopt the stochastic gradient descent with mini-batches and Adam (Kingma and Ba 2014) to update Θ. We employ different learning rates lr_1 and lr_2 on $\Theta^{(L)}$ and $\Theta^{(G)}$ respectively, where lr_2 is significantly smaller than lr_1. The regularizations employed are the same as described in Section 3.4.

4.4.5 Relation Inference

We now discuss the strategy for relation prediction, which is essentially the same as the one of Weston et al. (2013). In order to get the conditional probability $p(r|(h,t),\mathcal{S};\Theta)$, we use the weighed average to combine the two distribution $p(r|\mathcal{S};\Theta^{(L)})$ and $p(r|(h,t);\Theta^{(G)})$:

$$\begin{aligned} p(r|(h,t),\mathcal{S};\Theta) = \alpha * p(r|\mathcal{S};\Theta^{(L)}) \\ +(1-\alpha) * p(r|(h,t);\Theta^{(G)}). \end{aligned} \tag{4.12}$$

where α is the combining weight of the weighted average. Then, the predicted relation $\hat{r}$ is

$$\hat{r} = \operatorname*{argmax}_{r \in \mathcal{R} \cup \{NA\}} p(r|(h,t),\mathcal{S};\Theta). \tag{4.13}$$

4.5 Experiments

We now report on an experimental evaluation of our framework against the state-of-the-art for the RE task.

4.5.1 Datasets

We evaluate our model on the widely used NYT dataset, developed by Riedel, Yao, and McCallum (2010) by aligning Freebase relations mentioned in the New York Times Corpus. Articles from years 2005-2006 are used for training while articles from 2007 are used for testing.

As our KB, we used a subset with the 3M entities with highest degree (i.e., participating in most relations). Freebase is a manually curated Web-scale KB with approximately 80M entities, 23k kinds of relations and 1.2B facts. While Freebase is no longer maintained, it remains an invaluable resource in this area. Moreover, to prevent the knowledge representation from memorizing the true relations for entity pairs in the test set, we removed all entity pairs present in the NYT. In this way our model has to learn to generalize from other entities and relations.

4.5.2 Experimental Settings

Baselines: We choose five baseline appoaches following Y. Lin, Shen, et al. (2016) [1]: Three feature-based methods, **Mintz** (Mintz et al. 2009), **MultiR** (Hoffmann et al. 2011), **MIML** (Surdeanu et al. 2012), and two convolutional neural network based methods, **CNN+ATT** and **PCNN+ATT** (Y. Lin, Shen, et al. 2016). We also implement a baseline **Weston** based on the strategy following Weston et al. (2013), namely use Equation 4.12 to combine the scores directly without joint learning. Using α as 0.5 suggested by Weston et al. (2013), the performance is significantly worse. We found that with the models we trained $\alpha = 0.7$ leads to much better performance.

There are four variants of our proposed framework: (1) Hrere**-base**: basic neural model with local loss only; (2) Hrere**-naive**: neural model with both

[1] The results of baseline approaches are obtained from Y. Lin, Shen, et al. 2016

learning rate on $\Theta^{(L)}$	lr_1	5×10^{-4}
learning rate on $\Theta^{(K)}$	lr_2	1×10^{-5}
size of word position embedding	d_p	25
state size for LSTM layers	d_s	320
input dropout keep probability	p_i	0.9
output dropout keep probability	p_o	0.7
L2 regularization parameter	λ	0.0003
combining weight parameter	α	0.6

Table 4.3: Hyperparameter setting

local and global loss; (3) HRERE-**full**: neural model with both local and global loss along with their dissimilarities. (4) HRERE-**text**: same as HRERE-**full** except that the relation is inferred only by language representation (which is equivalent to set $\alpha = 1.0$ in Equation 4.12).

Evaluation Protocol: Following previous work (Mintz et al. 2009), we evaluate our model using held-out evaluation which approximately measures the precision without time-consuming manual evaluation. We report both Precision/Recall curves and Precision@N (P@N) in our experiments, ignoring the probability predicted for the *NA* relation. Moreover, to evaluate each sentence in the test set as in previous methods, we append T copies of each sentence into $\mathcal{S}$ for each testing sample.

Word Embeddings: We used the freely available 300-dimensional pre-trained word embeddings distributed by Pennington, Socher, and Manning (2014) to help the model generalize to words not appearing in the training set.

Hyperparameter Settings: For hyperparameter tuning, we randonly sampled 10% of the training set as a development set. All the hyperparameters were obtained by evaluating the model on the development set. With the well-tuned hyperparameter setting, we run each model five times on the whole training set and report the average P@N. For Precision/Recall curves, we just select the results from the first run of each model. For training, we set the iteration number over all the training data as 30. Values of the hyperparameters used in the experiments can be found in Table 4.3.

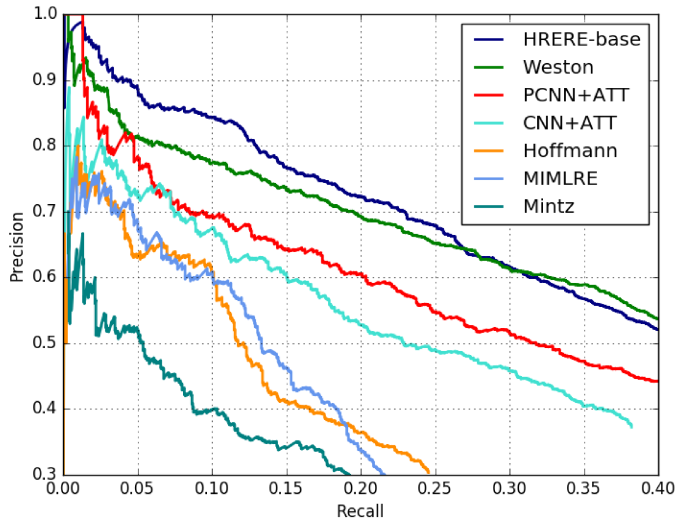

Figure 4.4: The Precision/Recall curves of baseline approaches and HRERE-**base**.

P@N(%)	10%	30%	50%
HRERE-base	81.8 ± 3.0	70.1 ± 2.2	60.7 ± 1.2
HRERE-naive	83.6 ± 2.7	74.4 ± 3.0	65.7 ± 2.6
HRERE-text	83.0 ± 3.0	71.8 ± 2.2	62.4 ± 1.7
HRERE-full	$\mathbf{86.1 \pm 2.7}$	$\mathbf{76.6 \pm 3.0}$	$\mathbf{68.1 \pm 2.4}$

Table 4.4: P@N of variants of our proposed model.

4.5.3 Performance Comparison and Analysis

Figure 4.4 shows the Precision/Recall curves for all baseline approaches and the variant of our model based on text only, HRERE-**base**. As one can see, HRERE-**base** significantly outperforms all baseline approaches and is very competitive with our tuned version of **Weston** over the entire range of recall. Possible reasons for the improvements can be attributed to the usage of better word embeddings, different neural architectures and improved attention mechanisms. For **Weston**, we can find that simple strategies to incorporate KBE are not able to give the expected improvements as observed in Weston *et al*. Weston et al. 2013 to our RE model anymore. In any case, since HRERE-**base** outperforms among all baselines for relation extraction, we only compare the variants of our proposed model in the rest of this section.

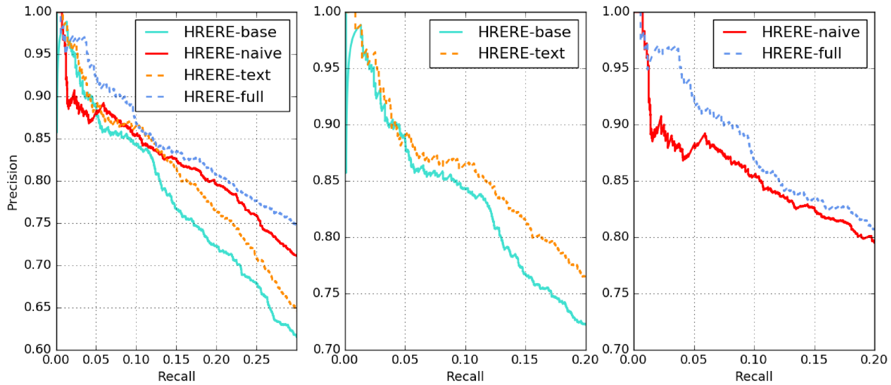

Figure 4.5: The Precision/Recall curves of variants of our proposed model. Left: All variants of our proposed model in the recall [0-0.3] region; Middle: The same plot zoomed to the recall [0-0.2] region with only HRERE-**base** and HRERE-**text**; Right: The same plot with only HRERE-**naive** and HRERE-**full**.

Improvements by using knowledge representation. From the leftmost plot in Figure 4.5 and Table 4.4, we can observe that HRERE-**base** performs worst compared to all other variants over the entire range of recall and all measurements for P@N, while HRERE-**full** always performs best. This suggests that introducing knowledge representation consistently results in improvements, which validates our hypothesis that knowledge representations can indeed improve on the RE task.

Although HRERE-**naive** and HRERE-**text** perform somewhere in between HRERE-**base** and HRERE-**full**, they have different behaviors compared with each other. They can be seen as degenerations of HRERE-**full**: HRERE-**naive** does not use dissimilarities between heterogeneous representations as part of the loss, while HRERE-**text** depends only on language representation to infer the relation. The difference in performance of these methods suggests that knowledge representation can affect the relation inference in different ways.

Improvements by connecting heterogeneous representations. From the middle and rightmost plots in Figure 4.5, we can better compare the different variants of our model. The middle plot shows that HRERE-**text** outperforms HRERE-**base** consistently. Note that both variants use only language representation to perform relation inference. The difference between them is that HRERE-**text** is trained with loss considering heterogeneous representations and their dissimilarities while HRERE-**base** is trained with language representation only. The plot suggests that jointly learning the heterogeneous representations bring mutual benefits which are out of reach of previous methods that learn each independently.

HRERE-**naive** simply optimizes both local and global loss at the same time without attempting to connect them. In a sense, this is the closest approach to the combining strategy of Weston et al. (2013), except of course for the training procedure. As shown in the rightmost plot in Figure 4.5, HRERE-**full** is not only consistently superior but also more stable when the recall is less than 0.1. One possible reason for the instability is that the results may be dominated by one of the representations and biased toward it, since there is no connection between the two heterogeneous representations. This suggests that connecting heterogeneous representations can increase the robustness of our model.

In any case, these results demonstrate that connecting heterogeneous representations can bring consistent improvements to our model.

4.5.4 Case Study

Table 4.5 shows four examples in the testing data. For each example, we show the relation, the sentence along with entity mentions and the corresponding probabilities predicted by HRERE-**base**, HRERE-**naive**, HRERE-**text** and HRERE-**full**. The entity pairs in the sentence are highlighted with bold formatting.

From the table, we have the following observations: (1) The predicted probabilities of four variants of our model in the table match the observations and validate the analysis discussed in Section 4.5.3. (2) From the text of the first two sentences, we can easily infer that *middle east contains Iran* and *Henry Fonda was born in Omaha*. However, HRERE-**base** fails to detect these relations, suggesting that it is hard for models based on language representations alone to detect implicit relations, which is reasonable to expect. With

Relation	Textual Mention	base	naive	text	full
contains	Much of the **middle east** tension stems from the sense that shiite power is growing, led by **Iran**.	0.311	0.864	0.687	**0.884**
place_of_birth	Sometimes I rattle off the names of movie stars from **Omaha**: Fred Astaire, **Henry Fonda**, Nick Nolte ...	0.109	0.605	0.229	**0.646**
neighborhood_of	Most of them also grew up in **New York City** neighborhoods like Hell's Kitchen, Forest Hills, Washington Heights and **Kew Gardens**, whiling away countless hours playing ...	0.196	0.427	0.740	**0.946**
country	Spokesmen for **Germany** and Italy in Washington said yesterday that they would reserve comment until the report is formally released at a news conference in **Berlin** today.	0.237	0.200	0.479	**0.880**

Table 4.5: Some examples in NYT corpus and the predicted probabilities of the true relations.

the help of KBE, the model can effectively identify implicit relations present in the text. (3) Even if the relation is explicitly stated in the text, the third example shows that models based only on language representation may fail when the two entities are far apart in the text. Likewise, the model can be improved for these cases with the help of KBE. (4) It may happen that the relation cannot be inferred by the text as shown in the last example. It is a case of an incorrectly labeled instance, a typical occurrence in distant supervision. However, the fact is obviously true in the KBs. As a result, Hrere-**full** gives the underlying relation according to the KBs. This observation may point to one direction of de-noising weakly labeled textual mentions generated by distant supervision.

4.6 Summary

This chapter describes a novel neural framework for jointly learning heterogeneous representations from both text information and facts in an existing knowledge base. A novel loss function is introduced to connect the heterogeneous representations seamlessly, also during training, allowing them to enhance each other. Our framework was tested on established benchmarks and built on publicly available datasets. We observe not only substantial improvements over state-of-the-art RE methods but also gains over the previous approach of Weston et al. (2013). Furthermore, our experiments suggest the textual mentions are likely to be incorrectly labeled by distant supervision when the heterogeneous representations disagree with each other, pointing out one direction of future as applying our framework to help de-noise the training corpus generated by distant supervision.

Chapter 5

Conclusion

In this thesis, we explored neural information extraction without manual annotated data following the distant supervision paradigm and presented novel neural models for different IE tasks which are particularly suited for this setting. All the proposed models achieved state-of-the-art performance in representative tasks.

In the first part of this thesis, we proposed an end-to-end solution with a neural network model for FETC that uses a variant of cross-entropy loss function to handle out-of-context labels, and hierarchical loss normalization to cope with overly-specific ones. Also, previous work solve FETC a multi-label classification followed by ad-hoc post-processing. In contrast, our solution is more elegant: we used public word embeddings to train a single-label that jointly learns representations for entity mentions and their context. We showed experimentally that our approach is robust against noise and consistently outperforms the state-of-the-art on established benchmarks for the task.

The second part of this thesis is a neural model with multi-level attention mechanisms for relation extraction. In this model, we employed bi-directional long short term memory to embed the semantics of sentences. Afterwards, we built both word-level and sentence-level attention, which is expected to dynamically reduce the weights of those noisy instances. Experimental results on real-world datasets show that, our model achieves significant and consistent improvements on relation extraction as compared with baselines.

The last main part of this thesis is that we described and evaluated a novel neural framework for jointly learning representation for RE and KBE tasks that uses a cross-entropy loss function to ensure both representations are learned together, resulting in significant improvements over the current state-of-the-art for relation extraction.

Lastly, there are also some future directions for our proposed models. While we proposed various neural models to effectively handle the noisy data due to direct distant supervision, it remains open how one can effectively de-noise the noisy data with different signals and different strategies instead of learning in the noisy setting as we do in this thesis. In addition, we can also incorporate more external information and different sources of supervision into our models to further improve the model performance.

References

Abhishek, Abhishek, Ashish Anand, and Amit Awekar (2017). “Fine-grained entity type classification by jointly learning representations and label embeddings.” In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Vol. 1, pp. 797–807 (cit. on pp. 9, 11, 18).

Bordes, Antoine et al. (2013). “Translating embeddings for modeling multi-relational data.” In: *Advances in neural information processing systems*, pp. 2787–2795 (cit. on pp. 35, 36, 69).

Collobert, Ronan and Jason Weston (2008). “A unified architecture for natural language processing: Deep neural networks with multitask learning.” In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 160–167 (cit. on pp. 65, 66).

Collobert, Ronan, Jason Weston, et al. (2011). “Natural language processing (almost) from scratch.” In: *Journal of Machine Learning Research* 12.Aug, pp. 2493–2537 (cit. on pp. 65, 66).

Daiber, Joachim et al. (2013). “Improving efficiency and accuracy in multilingual entity extraction.” In: *Proceedings of the 9th International Conference on Semantic Systems*. ACM, pp. 121–124 (cit. on p. 17).

Denil, Misha et al. (2012). “Learning where to attend with deep architectures for image tracking.” In: *Neural computation* 24.8, pp. 2151–2184 (cit. on p. 67).

Dettmers, Tim et al. (2017). “Convolutional 2d knowledge graph embeddings.” In: *arXiv preprint arXiv:1707.01476* (cit. on pp. 35, 70).

Dong, Xin et al. (2014). “Knowledge vault: A web-scale approach to probabilistic knowledge fusion.” In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 601–610 (cit. on p. 6).

Gillick, Dan et al. (2014). “Context-dependent fine-grained entity type tagging.” In: *arXiv preprint arXiv:1412.1820* (cit. on pp. 1, 6, 8, 17).

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. MIT press Cambridge (cit. on p. 61).

Han, Xu, Zhiyuan Liu, and Maosong Sun (2018). “Neural Knowledge Acquisition via Mutual Attention between Knowledge Graph and Text Neural Knowledge Acquisition via Mutual Attention between Knowledge Graph and Text.” In: *The 32th AAAI Conference on Artificial Intelligence (AAAI)* (cit. on p. 68).

Hinton, Geoffrey E et al. (2012). “Improving neural networks by preventing co-adaptation of feature detectors.” In: *arXiv preprint arXiv:1207.0580* (cit. on pp. 16, 28).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory.” In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 63).

Hoffmann, Raphael et al. (2011). “Knowledge-based weak supervision for information extraction of overlapping relations.” In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pp. 541–550 (cit. on pp. 24, 29, 44).

Huang, Yi Yao and William Yang Wang (2017). “Deep Residual Learning for Weakly-Supervised Relation Extraction.” In: *arXiv preprint arXiv:1707.08866* (cit. on p. 25).

Ji, Guoliang et al. (2015). “Knowledge graph embedding via dynamic mapping matrix.” 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. 687–696 (cit. on p. 69).

Ji, Xin et al. (2018). “Improving Neural Fine-Grained Entity Typing with Knowledge Attention.” In: *The 32th AAAI Conference on Artificial Intelligence (AAAI)* (cit. on p. 68).

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). “A convolutional neural network for modelling sentences.” In: *arXiv preprint arXiv:1404.2188* (cit. on p. 66).

Kim, Yoon (2014). “Convolutional neural networks for sentence classification.” In: *arXiv preprint arXiv:1408.5882* (cit. on p. 66).

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 28, 43).

Larochelle, Hugo and Geoffrey E Hinton (2010). “Learning to combine foveal glimpses with a third-order Boltzmann machine.” In: *Advances in neural information processing systems*, pp. 1243–1251 (cit. on p. 67).

LeCun, Yann et al. (1998). “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on p. 61).

Li, Xin and Dan Roth (2002). “Learning question classifiers.” In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pp. 1–7 (cit. on p. 6).

Lin, Thomas, Oren Etzioni, et al. (2012). “No noun phrase left behind: detecting and typing unlinkable entities.” In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 893–903 (cit. on p. 6).

Lin, Yankai, Zhiyuan Liu, Huanbo Luan, et al. (2015). “Modeling relation paths for representation learning of knowledge bases.” In: *arXiv preprint arXiv:1506.00379* (cit. on pp. 34, 36).

Lin, Yankai, Zhiyuan Liu, Maosong Sun, et al. (2015). “Learning entity and relation embeddings for knowledge graph completion.” In: *AAAI*. Vol. 15, pp. 2181–2187 (cit. on p. 69).

Lin, Yankai, Shiqi Shen, et al. (2016). “Neural relation extraction with selective attention over instances.” In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1, pp. 2124–2133 (cit. on pp. 24, 25, 27, 29, 37, 44, 68).

Ling, Xiao and Daniel S Weld (2012). “Fine-Grained Entity Recognition.” In: *AAAI* (cit. on pp. 1, 8, 17, 18).

Liu, Liyuan et al. (2017). “Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach.” In: *arXiv preprint arXiv:1707.00166* (cit. on p. 25).

Liu, Weiyang et al. (2016). “Large-Margin Softmax Loss for Convolutional Neural Networks.” In: *ICML*, pp. 507–516 (cit. on p. 67).

Luo, Bingfeng et al. (2017). “Learning with noise: enhance distantly supervised relation extraction with dynamic transition matrix.” In: *arXiv preprint arXiv:1705.03995* (cit. on p. 68).

Ma, Yukun, Erik Cambria, and Sa Gao (2016). “Label embedding for zero-shot fine-grained named entity typing.” In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 171–180 (cit. on p. 9).

Manning, Christopher et al. (2014). “The Stanford CoreNLP natural language processing toolkit.” In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60 (cit. on pp. 1, 5).

Mikolov, Tomas, Kai Chen, et al. (2013). “Efficient estimation of word representations in vector space.” In: *arXiv preprint arXiv:1301.3781* (cit. on p. 65).

Mikolov, Tomas, Ilya Sutskever, et al. (2013). “Distributed representations of words and phrases and their compositionality.” In: *Advances in neural information processing systems*, pp. 3111–3119 (cit. on p. 65).

Mintz, Mike et al. (2009). “Distant supervision for relation extraction without labeled data.” In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pp. 1003–1011 (cit. on pp. 2, 6, 8, 23, 29, 32, 44, 45).

Neelakantan, Arvind, Benjamin Roth, and Andrew Mc-Callum (2015). “Compositional vector space models for knowledge base inference.” In: *2015 aaai spring symposium series* (cit. on p. 32).

Nickel, Maximilian, Lorenzo Rosasco, Tomaso A Poggio, et al. (2016). “Holographic Embeddings of Knowledge Graphs.” In: *AAAI*, pp. 1955–1961 (cit. on p. 70).

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel (2011). “A Three-Way Model for Collective Learning on Multi-Relational Data.” In: *ICML*. Vol. 11, pp. 809–816 (cit. on p. 70).

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global vectors for word representation.” In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543 (cit. on pp. 18, 29, 45, 65).

Ren, Xiang, Wenqi He, Meng Qu, Lifu Huang, et al. (2016). “AFET: Automatic fine-grained entity typing by hierarchical partial-label embedding.” In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1369–1378 (cit. on pp. 9, 11, 18).

Ren, Xiang, Wenqi He, Meng Qu, Clare R Voss, et al. (2016). “Label noise reduction in entity typing by heterogeneous partial-label embedding.” In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 1825–1834 (cit. on pp. 9, 11, 18).

Ren, Xiang, Ahmed El-Kishky, et al. (2015). “Clustype: Effective entity recognition and typing by relation phrase-based clustering.” In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 995–1004 (cit. on pp. 1, 5).

Ren, Xiang, Zeqiu Wu, et al. (2017). “CoType: Joint extraction of typed entities and relations with knowledge bases.” In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1015–1024 (cit. on p. 33).

Riedel, Sebastian, Limin Yao, and Andrew McCallum (2010). “Modeling relations and their mentions without labeled text.” In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 148–163 (cit. on pp. 24, 28, 35, 44).

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 (cit. on p. 33).

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). “Learning representations by back-propagating errors.” In: *nature* 323.6088, p. 533 (cit. on p. 61).

Santos, Cicero Nogueira dos, Bing Xiang, and Bowen Zhou (2015). “Classifying relations by ranking with convolutional neural networks.” In: *arXiv preprint arXiv:1504.06580* (cit. on pp. 24, 66).

Shimaoka, Sonse et al. (2016). “An attentive neural architecture for fine-grained entity type classification.” In: *arXiv preprint arXiv:1604.05525* (cit. on pp. 9, 68).

Shimaoka, Sonse et al. (2017). “Neural architectures for fine-grained entity type classification.” In: *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)* . (Cit. on pp. 9, 11, 14, 16, 18).

Surdeanu, Mihai et al. (2012). “Multi-instance multi-label learning for relation extraction.” In: *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*. Association for Computational Linguistics, pp. 455–465 (cit. on pp. 24, 29, 44).

Toutanova, Kristina and Danqi Chen (2015). “Observed versus latent features for knowledge base and text inference.” In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66 (cit. on p. 35).

Toutanova, Kristina, Danqi Chen, et al. (2015). “Representing text for joint embedding of text and knowledge bases.” In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509 (cit. on p. 32).

Trouillon, Théo et al. (2016). “Complex embeddings for simple link prediction.” In: *International Conference on Machine Learning*, pp. 2071–2080 (cit. on pp. 39–41, 70).

Verga, Patrick, David Belanger, et al. (2015). “Multilingual relation extraction using compositional universal schema.” In: *arXiv preprint arXiv:1511.06396* (cit. on p. 33).

Verga, Patrick and Andrew McCallum (2016). “Row-less universal schema.” In: *arXiv preprint arXiv:1604.06361* (cit. on p. 68).

Vu, Ngoc Thang et al. (2016). “Combining recurrent and convolutional neural networks for relation classification.” In: *arXiv preprint arXiv:1605.07333* (cit. on p. 66).

Wang, Zhen et al. (2014a). “Knowledge graph and text jointly embedding.” In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1591–1601 (cit. on p. 32).

— (2014b). “Knowledge Graph Embedding by Translating on Hyperplanes.” In: *AAAI*. Vol. 14, pp. 1112–1119 (cit. on p. 69).

Weischedel, Ralph et al. (2013). “Ontonotes release 5.0 ldc2013t19.” In: *Linguistic Data Consortium, Philadelphia, PA* (cit. on p. 17).

Weston, Jason et al. (2013). “Connecting language and knowledge bases with embedding models for relation extraction.” In: *arXiv preprint arXiv:1307.7973* (cit. on pp. 32–35, 38, 43, 44, 46, 48, 50).

Wu, Yi, David Bamman, and Stuart Russell (2017). “Adversarial training for relation extraction.” In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1778–1783 (cit. on pp. 25, 37).

Xiao, Han, Minlie Huang, and Xiaoyan Zhu (2015). “From one point to a manifold: Knowledge graph embedding for precise link prediction.” In: *arXiv preprint arXiv:1512.04792* (cit. on p. 69).

Yang, Bishan et al. (2014). "Embedding entities and relations for learning and inference in knowledge bases." In: *arXiv preprint arXiv:1412.6575* (cit. on p. 70).

Yin, Wenpeng and Hinrich Schütze (2016). "Multichannel variable-size convolution for sentence classification." In: *arXiv preprint arXiv:1603.04513* (cit. on p. 66).

Yogatama, Dani, Daniel Gillick, and Nevena Lazic (2015). "Embedding methods for fine grained entity type classification." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Vol. 2, pp. 291–296 (cit. on p. 8).

Yu, Yang et al. (2016). "End-to-end answer chunk extraction and ranking for reading comprehension." In: *arXiv preprint arXiv:1610.09996* (cit. on p. 66).

Zeng, Daojian, Kang Liu, Yubo Chen, et al. (2015). "Distant supervision for relation extraction via piecewise convolutional neural networks." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762 (cit. on pp. 24, 37).

Zeng, Daojian, Kang Liu, Siwei Lai, et al. (2014). "Relation classification via convolutional deep neural network." In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 2335–2344 (cit. on pp. 12, 24, 26, 66).

Zeng, Wenyuan et al. (2016). "Incorporating relation paths in neural relation extraction." In: *arXiv preprint arXiv:1609.07479* (cit. on p. 25).

Zhou, Peng et al. (2016). "Attention-based bidirectional long short-term memory networks for relation classification." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vol. 2, pp. 207–212 (cit. on pp. 13, 24, 25, 27).

Appendix A

Background

A.1 Deep Neural Networks for NLP

Over the past few years, we have witnessed the huge success of neural networks as powerful machine-learning models, yielding state-of-the-art results in many fields including NLP. In this section, a brief introduction is given on deep neural network models from the perspective of NLP research with a focus on information extraction.

A.1.1 Deep Neural Network Basics

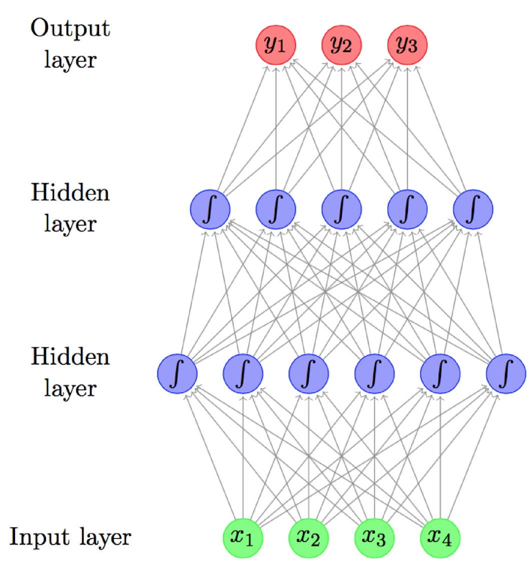

Figure A.1: Feed-forward neural network with two hidden layers.

As shown in Figure A.1, a typical feed-forward DNN consists at least of three layer types: the input layer, the hidden layer and the output layer. All these layers are formed by neurons which are denoted by circles, with incoming arrows being the neuron's inputs and outgoing arrows being the neuron's outputs. The input layer has no incoming arrows, and is the input to the network. The output layer has no outgoing arrows, and is the output of the network. The other layers are considered as "hidden". By adding more hidden layers, the neural network can describe highly complex functions. In the figure, each neuron is connected to all of the neurons in the next layer, which is called a *fully-connected* layer.

The units of the input layer represent different features x_i of the input data, while the units of the output layer represent one or more classes y_i. A DNN describes a function $\hat{y} = f^*(x)$ which maps the input features X over several hidden layers to the output classes Y. This function thereby approximates a real while unknown function $y = f(x)$. A DNN approximates the function $f()$ by fitting the model's parameters θ so that predicted outputs $\hat{y}$ are as close as possible to the real outputs y.

The learning process of a DNN consists of two iterative steps: 1) *Forward propagation* - computing the prediction $\hat{y}$ by current parameters θ; 2) Backpropagation - updating the parameters by the current loss between the prediction $\hat{y}$ and the ground truth y.

In the forward propagation, a neuron in successive layer is computed by a weighted summation over all neurons of previous layer. The values of each row of neurons in the network can be thought of as a vector. In Figure A.1 the input layer is a 4 dimensional vector $\mathbf{x}$, and the layer above it is a 6 dimensional vector $\mathbf{h}^1$. The fully connected layer can be thought of as a linear transformation from 4 dimensions to 6 dimension. A fully-connected layer implements a vector-matrix multiplication, $\mathbf{h} = \mathbf{x}\mathbf{W}$ where the weight of the connection from the ith neuron in the input row to the jth neuron in the output row is $\mathbf{W}_{ij}$. The values of $\mathbf{h}$ are then transformed by a non-linear function g that is applied to each value before being passed on to the next input. In a similar way, all neuron values in subsequent layers can be derived layer-by-layer, until reaching the output layer. At the output layer, a loss function $L(\hat{y}, y; \theta)$ is needed to estimate how good the prediction as compared to the ground truth.

In order to minimize the loss function in neural networks, variants of the gradient descent algorithm are used. The key insight of the gradient descent is that the value of the loss function decreases if we adjust the parameters along the direction of the gradients of the loss function with respect to the current parameters. In gradient descent, the parameters are initialized randomly and the parameters are iteratively moved toward the direction of the gradients until the loss function coverages. The parameters θ can be updated as follows:

$$\theta_{new} = \theta_{old} - \eta \frac{\partial L(\hat{y}, y; \theta)}{\partial \theta}\Big|_{\theta=\theta_{old}} \tag{A.1}$$

where coefficient η is called learning rate, affecting network learning speed.

The gradients of deep neural networks seem to be difficult to compute, but there is an efficient algorithm called backpropagation popularized by Rumelhart, Hinton, and Williams (1986) in the middle eighties. Essentially, backpropagation is just another name for the chain rule used in the basic calculus. The details of this algorithm are well explained in the book by Goodfellow et al. (2016).

Convolutional Neural Network. Due to the fact that the fully-connected networks have complete connection between consecutive layers, the parameter matrices become very large and the matrix multiplications are computationally expensive. The idea of Convolutional Neural Networks (CNNs) (LeCun et al. 1998) is to reduce the connections between the input units and the hidden units, instead of fully connecting them. Each hidden unit will obtain weighted inputs only from selected input units. With a sequence of words as input, this means only a local phrase (*i.e.*, n-gram) will be processed by a hidden unit in CNN. Then, we elaborate the architecture of a CNN for processing textual inputs.

The input to the CNN is a sequence x containing m entries. Each entry is represented by a d-dimensional dense vector; thus the input x is represented as a feature map of dimensionality $d \times m$. Convolution layer is used for representation learning by sliding n-grams. For an input sequence with m entries: $x_1, x_2, \ldots, x_m$, let vector $\mathbf{c}_i \in \mathbb{R}^{nd}$ be the concatenated embeddings of n entries $x_{i-n+1}, \ldots, x_i$ where n is the filter width and $0 < i < m + n$. Embeddings for x_i, $i < 1$ or $i > m$, are zero padded. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d'}$ for the n-gram $x_{i-n+1}, \ldots, x_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d' \times nd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c_i} + \mathbf{b}) \tag{A.2}$$

where bias $\mathbf{b} \in \mathbb{R}^{d'}$. All n-gram representations $\mathbf{p}_i (i = 1, \ldots, m + n - 1)$ are used to generate the representation of input sequence x by maxpooling: $\mathbf{x}_j = \max(\mathbf{p}_{1,j}, \mathbf{p}_{2,j}, \ldots)(j = 1, \ldots, d)$. The objective of maxpooling is to down-sample an input representation, reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions. This is done in part to help prevent over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

Recurrent Neural Network. When dealing with text, it is very common to work with sequences, such as words (sequences of letters), sentences (sequences of words) and documents. Recurrent neural networks (RNNs) allow representing arbitrarily sized structured inputs in a fixed-size vector, while paying attention to the structured properties of the input. For RNN, each current input x_t is composed with the previous hidden state h_{t-1} to generate a new hidden state at time t as follow:

$$\mathbf{h}_t = \sigma(\mathbf{V}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{A.3}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ represents the token in x at position t, $\mathbf{h}_t \in \mathbb{R}^h$ is the hidden state at t, supposed to encode the history $x_1, \ldots, x_t$. $\mathbf{V} \in \mathbb{R}^{h \times d}$ and $\mathbf{U} \in \mathbb{R}^{h \times h}$ are parameters.

By this recurrent procedure, all inputs in X can be encoded sequentially into a global representation. Unfortunately, vanishing gradient problem prevents standard RNNs from learning long-term dependencies. Long Short Term Memory (LSTMs) (Hochreiter and Schmidhuber 1997) were designed to combat vanishing gradients through a gating mechanism. LSTM models the word sequence x as follows:

$$\mathbf{i}_t = \sigma(\mathbf{x}_t \mathbf{U}^i \mathbf{x}_t + \mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{A.4}$$

$$\mathbf{f}_t = \sigma(\mathbf{U}^f \mathbf{x}_t + \mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{A.5}$$

$$\mathbf{o}_t = \sigma(\mathbf{U}^o \mathbf{x}_t + \mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{A.6}$$

$$\mathbf{q}_t = \tanh(\mathbf{U}^q \mathbf{x}_t + \mathbf{W}^q \mathbf{h}_{t-1} + \mathbf{b}_q) \tag{A.7}$$

$$\mathbf{p}_t = \mathbf{f}_t \circ \mathbf{p}_{t-1} + \mathbf{i}_t \circ \mathbf{q}_t \tag{A.8}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{p}_t) \tag{A.9}$$

LSTM has three gates: input gate i_t, forget gate f_t and ouput gate o_t. All gates are generated by a *sigmoid* function over the ensemble of input x_t and the preceding hidden state h_{t-1}. In order to generate the hidden state at current step t, it first generates a temporary result q_t by a *tanh* non-linearity over the ensemble of input x_t and the preceding hidden state h_{t-1}, then combines this temporary result q_t with history p_{t-1} by input gate i_t and forget gate f_t respectively to get updated history p_t, finally uses output gate o_t over this updated history p_t to get the final hidden state h_t.

A.1.2 Word Distributed Representations

The success of DNNs in NLP partially relies on the outstanding distributed representations of words. We can also treat a DNN as a system with two modules, one is representing input units, the other is composing the unit representations. In this perspective, effective word representations acts as the first backbone of a successful DNN system.

Word distributed representations, also called "word embeddings", are low-dimensional, dense vectors with continuous values. Given a vocabulary with size V, one-hot representation denotes each single word as a binary vector of length $|V|$ with one value 1 at the word-specific index and remaining values 0. It enjoys simplicity, however, it is memory inefficient and the word similarity is unable to be detected. Unlike the conventional one-hot representations, a single word in embedding space is a d-dimensional vector (mostly $d << |V|$); similar words will have similar vectors - information is shared between similar words. One benefit of using dense and low-dimensional vectors is computational, the other is generalization power if we believe some features may provide similar clues, it is worthwhile to provide a representation that is able to capture these similarities.

Due to the importance of word embeddings in DNN systems, there are large numbers of works specifically studying the learning of high-quality word embeddings. When enough supervised training data is available, one can just treat the embeddings the same as the other model parameters: initialize the embedding vectors to random values, and let the network-training procedure tune them into "good" vectors. However, the common case is that we do not have sufficient annotated data. In such cases, we resort to "unsupervised" methods, which can be trained on huge amounts of unannotated text.

The key idea behind the unsupervised approaches is that one would like the embedding vectors of "similar" words to be similar. While word similarity is hard to define and is usually very task-dependent, the current approaches derive from the distributional hypothesis, stating that *words are similar if they appear in similar contexts*. The different methods all created supervised training instances in which the goal is to either predict the word from its context, or predict the context from the word.

An important benefit of training word embeddings on large amounts of unannotated data is that it provides vector representations for words that do not appear in the supervised training set. Ideally, the representations for these words will be similar to those of related words that do appear in the training set, allowing the model to generalize better on unseen events. It is thus desired that the similarity between word vectors learned by the unsupervised algorithm captures the same aspects of similarity that are useful for performing the intended task of the network.

Common unsupervised word-embedding algorithms include `word2vec` (Mikolov, K. Chen, et al. 2013; Mikolov, Sutskever, et al. 2013), `GloVe` (Pennington, Socher, and Manning 2014) and the Collobert and Weston (Collobert and Weston 2008; Collobert, Weston, et al. 2011) embeddings algorithm. These models are inspired by neural networks and are based on stochastic gradient training. However, they are deeply connected to another family of algorithms which evolved in the NLP and IR communities, and that are based on matrix factorization. In addition, it's worth noting that the word embeddings derived by unsupervised training algorithms have a wide range of applications in NLP beyond using them for initializing the word-embeddings layer of a neural-network model.

A.1.3 Context Representation Learning

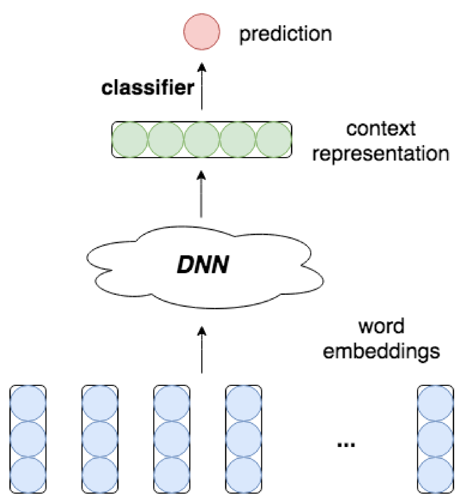

Figure A.2: Common paradigm of context representation learning by DNN.

As neural information extraction addressed in this thesis replies on context representation learning, this section briefly describes the progresses of context representation learning with DNNs. Similar to image processing, contexts are initialized into a matrix representation, such as a matrix with columns denoting word embeddings in sequence. Then, CNNs model sentences locally and hierarchically while RNNs model them sequentially.

A common paradigm of context representation learning by DNN is illustrated in Figure A.2. As input, each word is denoted by an embedding (randomly initialized or pretrained). A DNN system works on this context input format to generate a global context representation, which is finally fed into a classifier to make final predictions. Literature mainly made progresses in terms of *enriching input representations*, *enriching DNN expressivity* and *improving loss function*.

Enriching Input Representation. The pioneering work (Collobert and Weston 2008; Collobert, Weston, et al. 2011) obtained great success by presenting words into *word embeddings*. This kind of initialization afterwards acts as a mainstream in downstream NLP tasks. Some work (Kim 2014; Yin and Schütze 2016) explored initializing words by multiple pretrained word embeddings, as different pretrained embedding versions are supposed to provide complementary information.

In addition to the word embeddings as input layer, linguistic features are often incorporated into DNNs for better performance. For example, Yu et al. (2016) add part-of-speech tags to the words in machine comprehension task. D. Zeng, K. Liu, Lai, et al. (2014) considered position features between generic words and entity mentions for relation classification. Generally, linguistic features can provide strong support to the DNN systems, especially in the case of limited training set.

Enriching DNN expressivity. Collobert and Weston (2008) and Collobert, Weston, et al. (2011) used basic convolution layer and max-pooling layer to model sentences. Kalchbrenner, Grefenstette, and Blunsom (2014) proposed *k-max pooling* for CNN. Vu et al. (2016) combined CNN and RNN for sentence-level relation classification.

Improving Loss Functions. For sentence classification tasks, the most commonly-used loss function is negative likelihood (a.k.a cross-entropy loss). Santos, Xiang, and B. Zhou (2015) presented a ranking loss to make the true label score above a positive threshold and the false label score below a negative threshold. W. Liu et al. (2016) proposed a generalized large-margin softmax loss which explicitly encourages intra-class compactness and inter-class separability between learned features. The purpose is to generalize the cross-entropy loss to a more general large-margin loss in terms of angular similarity, leading to potentially larger angular separability between learned features.

A.1.4 Attention Mechanisms for NLP

A recent trend in the design of DNNs is attention mechanisms. Attention in neural networks are loosely based on the visual attention mechanism found in humans. Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to focus on a certain region of an image with "high resolution" while perceiving the surrounding image in "low resolution", and then adjusting the focal point over time.

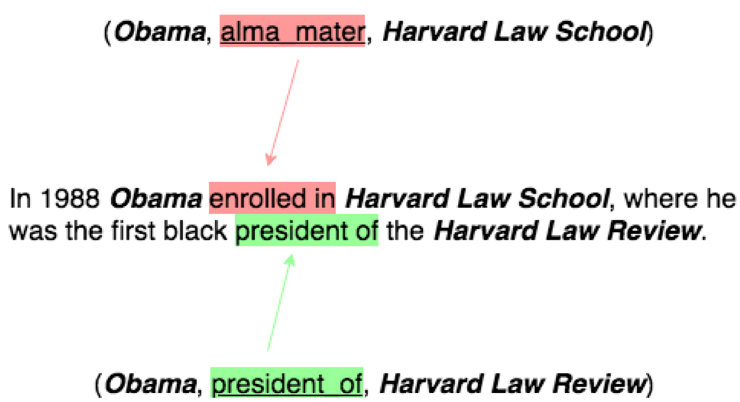

Figure A.3: Attention example for relation extraction task.

Attention in neural networks has a long history, particularly in image recognition (Denil et al. 2012; Larochelle and Hinton 2010). But only recently have attention mechanisms made their way into recurrent neural networks architectures that are typically used in NLP. Conventional information extraction does not distinguish which parts of the given texts are more indicative for the task. Human beings, instead, can figure out which part matches well to the information we're interested in. For example, in relation extraction task shown in Figure A.3, our goal is to extract the proper relations for two different entity pairs given the same context. For the entity pair *(Obama, Harvard Law School)*, we should focus on the phrase *enrolled in*; while for the entity pair *(Obama, Harvard Law Review)*, we should give more attention to the phrase *president of*.

As a result, attention mechanisms are intensively explored recently in the domain of information extraction. For relation extraction, Y. Lin, Shen, et al. (2016) and Luo et al. (2017) built a sentence-level attention over multiple instances to reduce weights of noisy instances. Verga and McCallum (2016) used neural networks with attention to merge similar semantic patterns in universal schema. For fine-grained type classification, Shimaoka et al. (2016) proposed an attentive neural network model that used LSTMs to encode context of an entity mention and used an attention mechanism to allow the model to focus on relevant expressions in such context. Mostly recently, Han, Z. Liu, and Sun (2018) and X. Ji et al. (2018) attempted to incorporate encoded knowledge information to build the attention to facilitate the tasks of information extraction.

A.2 Knowledge Base Embeddings

Representing information about real-world entities and their relations in structured knowledge bases (KBs) enables various applications such as structured search, factual question answering, and intelligent virtual assistants. A major challenge for using discrete representation of knowledge base is the lack of capability of accessing the similarities among different entities and relations. Knowledge Base Embedding (KBE) techniques have been proposed in recent years to deal with this issue. The main idea is to represent the entities and relations in a vector space, and one can use machine learning technique to learn the continuous representation of the knowledge base in the latent space.

For a given knowledge base, let $\mathcal{E}$ be the set of entities with $|\mathcal{E}| = n$, $\mathcal{R}$ be the set of relations with $|R| = m$, and $\mathcal{T}$ be the set of ground truth triples. In general, a knowledge base embedding model can be formulated as a score function $f_r(s, o)$, $s, o \in \mathcal{E}, r \in \mathcal{R}$ which assigns a score to every possible triple in the knowledge base. The estimated likelihood of a triple being correct depends only on its score given by the score function.

Different models formulate their score function based on different designs, and therefore interpret scores differently, which further leads to various training objectives.

A.2.1 Translation-based Models

Translation based models are based on the principle first proposed by Bordes et al. (2013) that if there exists a relationship r between entities s, o then the following relationship between their respective embeddings holds: $\mathbf{e}_s + \mathbf{w}_r \approx \mathbf{e}_o$. The scoring function is thus designed as

$$f_r(s, o) = \|\mathbf{e}_s + \mathbf{w}_r - \mathbf{e}_o\|, \tag{A.10}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^K$ are entity embedding vectors, $\mathbf{w}_r \in \mathbb{R}^K$ is the relation embedding vector and K is the embedding size. `TransE` (Bordes et al. 2013) is the first model to introduce translation-based embedding. Later variants, such as `TransH` (Z. Wang et al. 2014b), `TransR` (Y. Lin, Z. Liu, Sun, et al. 2015), `TransD` (G. Ji et al. 2015) extend `TransE` by projecting the embedding vectors of entities into various spaces. `ManifoldE` (Xiao, M. Huang, and Zhu 2015) embeds a triple as a manifold rather than a point. The objective of training a translation-based model is typically minimizing the following marginal loss:

$$\mathcal{J}_m = \sum_{(s,r,o) \in \mathcal{T}} [\gamma + f_r(s, o) - f_{r'}(s', o')]_+ \tag{A.11}$$

where $[\cdot]_+ = \max(0, \cdot)$ is the hinge loss, γ is the margin (often set to 1), and (s', r', o') is a negative triple generated based on the positive triple.

A.2.2 Latent Factor Models

Latent factor models assume that the probability of the existence of a triple (s, r, o) is given by the logistic link function:

$$P(\mathbf{Y}_{so}^{(r)}) = \sigma(\mathbf{X}_{so}^{(r)}) = \sigma(f_r(s, o)) \tag{A.12}$$

where $\mathbf{X}^{(r)} \in \mathbf{R}^{n \times n}$ is a latent matrix of scores of relation r. Latent factor models try to find a generic structure for $\mathbf{X}^{(r)}$ that leads to a flexible approximation of common relations in real world KBs with matrix factorization. The goal here is to learn representations of the entities and relations capable of predicting probabilities of $\mathbf{Y}_{h't'}^{(r')}$ being true for unobserved triple (h', r', t').

`RESCAL` (Nickel, Tresp, and Kriegel 2011) is one of the earliest studies on embedding based on latent matrix factorization, using a bilinear form as score function. `DistMult` (Yang et al. 2014) simplifies `RESCAL` by only using a diagonal matrix. The score function is defined as follows:

$$f_r(s, o) = \langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle. \tag{A.13}$$

where $\mathbf{w_r} \in \mathbb{R}^K$. However, this model loses much expressiveness due to its simplicity and cannot describe antisymmetric relations accurately. In order to handle these issues, `ComplEx` (Trouillon et al. 2016) transforms the embeddings of `DistMult` from real space to complex space, which defines the score function as:

$$f_r(s, o) = \mathrm{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle), \tag{A.14}$$

where $\mathbf{w_r} \in \mathbb{C}^K$. `HOLE` (Nickel, Rosasco, Poggio, et al. 2016) employs circular correlation to combine the two entities in a triple. `ConvE` (Dettmers et al. 2017) uses a convolutional neural network as the score function.