NAIST-IS-DD1461008

Doctoral Dissertation

An Exemplar-Based Approach to Word Representation for Information Extraction

Takuo Hamaguchi

February 01, 2018

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of SCIENCE

Takuo Hamaguchi

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Satoshi Nakamura	(Co-Supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Assistant Professor Shindo Hiroyuki	(Co-supervisor)
Assistant Professor Hiroshi Noji	(Co-supervisor)

An Exemplar-Based Approach to Word Representation for Information Extraction*

Takuo Hamaguchi

Abstract

Word representations are the foundation of natural language processing. In particular, *continuous representations* have been widely used in recent years, due to the superb ability to capture semantics of words and compatibility with neural networks.

In continuous representations, embedding vectors are trained and then fixed at runtime. Because of the fixed embedding vectors, they face difficulties in coping with problems that require processing runtime contexts, such as handling out-of-vocabulary (OOV) words and polysemy.

To overcome the problem of fixed embedding vectors, we propose an exemplar-based framework for continuous representations, which computes embedding vectors for words on the fly. It does not fix a representation of a word but exploits objects associated with the word, such as contexts containing the word. A single vector is distilled from these vectors at runtime to suit the task, with a neural network based on an attention mechanism.

Using our framework, we address two information extraction tasks: (i) entity classification and (ii) knowledge base completion. In the former task, the objects associated with words are their contexts in large corpora, and in the latter, they are the neighborhood relations and entities (words) in a knowledge graph. The experimental results show the effectiveness of our proposed framework in both settings.

Keywords:

Distributional Hypothesis, Out-of-Vocabulary, Polysemy, Weakly supervision, Exemplar-based Approach, Word representation

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1461008, February 01, 2018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Continuous Representations and Their Problem

Word representations are the foundation of natural language processing. In particular, *continuous representations* have been widely used in recent years, due to the superb ability to capture semantics of words and compatibility with neural networks.

We can describe continuous representation $f_{\mathbb{R}}$ as

$$f_{\mathbb{R}} : \mathcal{V} \rightarrow \mathbb{R}^d \tag{1.1}$$

where $\mathcal{V}$ is a vocabulary containing words we deal with and $\mathbb{R}^d$ is a d dimensional vector space. A vector $f_{\mathbb{R}}(\mathsf{w})$ is called an *embedding vector* of a word w. Compared with previous representations such as one-hot representation and tf-idf vectors, embedding vectors are trainable. This trainability allows embedding vectors to capture semantics with which the previous representations have difficulties in dealing. For example, *continuous Bag-of-Words* (cBoW) [74] trains embedding vectors to predict a word using surrounding words via optimizing the following function:

$$\begin{aligned} p(\mathsf{w} \mid \text{context}) &= \frac{exp\langle \mathbf{w}, \mathbf{h} \rangle}{\sum_{\mathsf{w}' \in \mathcal{V}} exp\langle \mathbf{w}', \mathbf{h} \rangle} \\ \mathbf{h} &= \frac{1}{N} \sum_{\mathsf{c} \in \text{context}} \mathbf{c} \end{aligned}$$

where $\text{context} = \{\mathsf{c}_1, , , \mathsf{c}_N\}$ is a set of words $\mathsf{c}_i \in \mathcal{V}$ surrounding w. Optimizing this function enables the trained embedding vectors to contain contextual information. In addition to the cBoW, many models of continuous word representations have been proposed [88, 115].

Using continuous representations with neural networks has produced successful results. For example, recurrent neural networks [38, 100] and convolutional neural networks [50, 26] were combined with embedding vectors to tackle parsing [22, 118] and mechine translation [107]. As these studies show, continuous representaitons take on increasing importance in recent natural language processings.

In continuous representations, embedding vectors are trained and then fixed at runtime. We call continuous representations whose embedding vectors are fixed in runtime *fixed representations*. Because of the fixed embedding vectors, these representations face difficulties in coping with problems that require processing runtime context, such as handling *out-of-vocabulary* (OOV) words and polysemy. Fixed representations represent each word by a single or several fixed-length vector. Thus, for polysemy words, they cannot represent only their prototypical senses. Detailed senses contained in different contexts are lost in the training process.

1.2 Distributional Hypothesis and Exemplar-based Approach

To overcome the problem of fixed embedding vectors, we revisit the distributional hypothesis: "A word is characterized by the company it keeps" [35]. This idea is a classical concept rather than a novel research topic studies in these days. Indeed, this hypothesis dates back to studies of Ferdinand de Saussure, who is the pioneer of early linguistics in the mid-19th to early-20th century. He argues that "meaning depends on its relation to other words within the system" [18]. Interestingly, recent continuous representations are inspired by this distributional hypothesis. For example, cBoW attempts to capture contextual information to represent words. However, fixed representations used in these models go against the spirit of the distributional hypothesis, because they are based on the assumption that "a word is characterized by corresponding embedding vectors." This assumption is the cause of problems explained above.

The exemplar-based approach [95] provides a clue to solve the problems of fixed representations. This approach uses a collection of usages of a word as its representation. If we interpret the collection of usages as "the company a word keeps," this approach can be regarded as a straightforward implementation of the distributional hypothesis. The fundamental assumption here is that the usages of a word can characterize the word. The exemplar-based approach is thus conceptually truthful to the

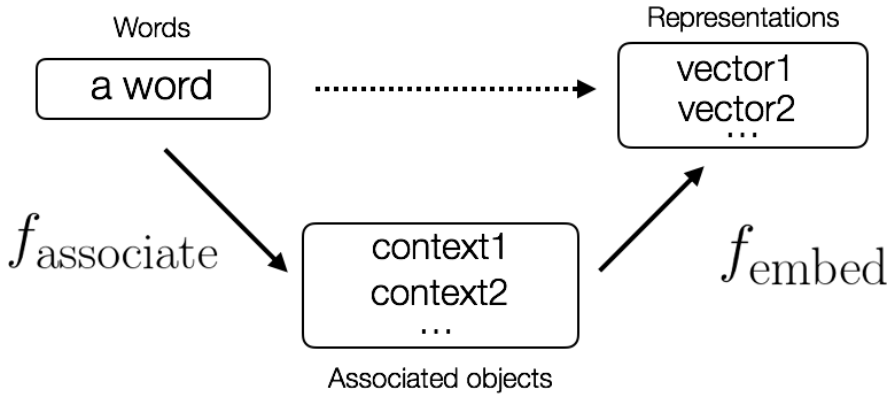


Figure 1.1: Illustration of proposed word representation based on the distributional hypothesis.

distributional hypothesis, it has been not well explored because "[i]ndividual exemplars can be quite noisy and the model can incur high computational overhead" [95]. It was considered difficult to apply the exemplar-based approach to previous representations including n-grams. No attempts were made to apply an exemplar-based approach to continuous representations, either.

1.3 Word Representations based on the Distributional Hypothesis

Inspired by the exemplar-based approach we propose a word representation based on the distributional hypothesis. The proposed word representation consists of the following two functions: association and embedding. To represent a word, the association function collects associated objects of the word and the embedding function converts the associated objects into continuous-valued vectors. We can describe our representation as follows

$$f_{\text{associate}} \quad : \quad \mathsf{w} \in \mathcal{V} \mapsto \{\mathrm{o}_i\}_i \subset \mathcal{O} \tag{1.2}$$

$$f_{\text{embed}} \quad : \quad \{\mathrm{o}_i\} \mapsto \{\mathbf{x}_i\}_i \in 2^{\mathbb{R}^d} \tag{1.3}$$

where $\{\mathrm{o}_i\}_i$ is a set of associated objects. Our representation f is composit function of these two function $f = f_{\text{embed}} \circ f_{\text{associate}}$ and $f_{\text{embed}} \circ f_{\text{associate}}(\mathsf{w})$ is a representation of a word w composed of vectors. This representation is illustrated in the figure 1.1.

In this word representation, we exploit neural networks such as an attention mechanism to mitigate noisy collections, recent progress in GPGPU to overcome the high

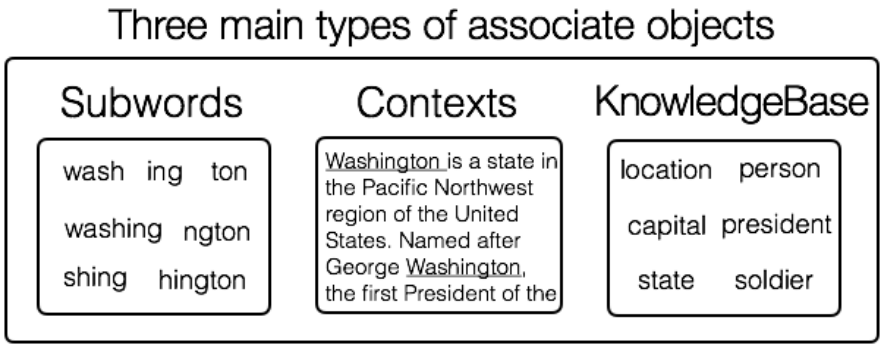


Figure 1.2: Three main types of associated objects in the case of *Washington*.

computational cost. As explained above, continuous representations have superb ability to capture semantics. The ability allows us to evaluate how exemplars are useful and then to mitigate noisiness in collections. Based on the weak supervision scheme, we expand exemplars containing usages of words to other objects related to words, e.g., neighborhoods of words in knowledge bases. The weak supervision (or distant supervision) is an annotation scheme, which associates a target object with weakly related objects. For example, contexts containing an entity are associated with the entity in information extraction tasks. The idea of the weak supervision is that relevant objects are contained in collections if a number of the objects is large enough. Thus, there is no guarantee that all object in collections are "exemplars" because the weak supervision allows collected objects to contains irrelevant objects. We call the object related to words *associated objects*.

1.3.1 Three Types of Associated Objects

There are three main types of associated objects: subwords approach, contextual approach, and relational approach. These associated objects are illustrated in the figure 1.2.

The first approach [52, 66, 17] subdivides the OOV words into literal ingredients, e.g., sequences of characters, subwords, and parts of characters. Thus, previous work taken this approach exploits the subdivided words as associated objects to represent OOV words. This approach is widely studied in previous work because additional costs are small compared with other approaches.

The contextual approach [117, 37, 31] exploits contexts related to OOV words as associated objects. For example, in machine translation, pointing mechanism exploits

representation vectors in source texts to generate target texts by adding the vectors to output word vectors. Furthermore, various textual resources including large-scale corpora (e.g., Wikipedia), descriptions in knowledge bases, and definitions in dictionaries are available in this contextual approach. Benefits of this approach are that we can leverage potential values of large-scale corpora because association procedure does not require additional cost to annotate corpora.

Finally, the relational approach exploits neighborhoods of OOV words in graphs as associated objects. We can apply this approach to the situation we want to represent low-frequency words, specifically, technical terms. Because technical terms do not appear in ordinary corpora, we need to represent OOV words without textual information. In information extraction tasks, this relational approach is useful.

Our proposed representation a generalization of these approaches. The differences between the approaches come from what we use associated objects to represent words. These three approaches share the procedure embedding each associated object into continuous vector space. For example, subdivided words have the base embeddings in subdivision approach, and ordinary words have the base embedding in the contextual approach. This mechanism fits our intuition because we learn meanings of unknown words from contexts related to the words in daily lives. We call this embedding as *base embedding*. These associated objects and base embeddings allow us to calculate embedding vectors of OOV words.

In this thesis, we address contextual approach and relational approach. In particular, we verify the effectiveness of our proposed representation in two settings, with texts and without texts, in entity classification and knowledge base completion respectively. In the entity classification, we exploit large-scale corpora based on the contextual approach to model polysemy of OOV words. In this work, we also explore how to deal with vector set representation using attention mechanism. In contrast to the setting with texts, we exploit neighborhoods of entities in knowledge graph to represent entities based on relational approach. We tackle the problem of out-of-knowledge-base entities, which are OOV words in information extraction.

1.4 Outline of Thesis

The remainder of this thesis is organized as follows.

In Chapter 2, we have proposed an exemplar-based word representation called "Bag-of-Contexts" (BoC). At training time, each context of a word collected from a cor-

pus is converted to a vector, and they are collectively used as a representation (BoC representation) of the word. When a word is given for testing in a specific context, relevant information is distilled from the BoC vectors of the word through the attention mechanism. This approach provides a natural way of dealing with polysemy and out-of-vocabulary words. We have verified the effectiveness of the proposed model in the task of entity classification, which is essentially the task of classifying out-of-vocabulary words.

In Chapter 3, we proposed a new KBC task in which entities unobserved at training time are involved. For this task, we proposed a Graph-NN tailored to KBC with OOKB entities. We conducted two triplet classification tasks to verify the effectiveness of our proposed model. In the OOKB entity problem, our model outperformed the baselines considerably. Our model also showed state-of-the-art performance on WordNet11 in the standard KBC setting.

In Chapter 4, we summarize this thesis. The results contain experiments of contextual approach and relational approach, and future directions of the proposed representation.

Chapter 2

Preliminaries

In this chapter, we describe the background knowledge on word representations and neural networks. The word representations contain standard word representations including one-hot representation, continuous Bag-of-Words, and skip-gram, which represent words using vectors directly. Note that these representations are direct representation which maps words to representation vectors directly,

2.1 Notation

Define $\mathbb{R}^d$ as a d dimensional vector space with and bold symbols (e.g., $\mathbf{x}$) as vectors in a vector space. We denote the i-th value of a vector $\mathbf{x}$ as $\mathbf{x}[i] \in \mathbb{R}$ and a set of vectors as $\{\mathbf{x}_j \mid \mathbf{x}_j in \mathbb{R}^d\}$. The inner-product between two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ is given by $\langle \mathbf{x}, \mathbf{x}' \rangle = \sum_{i=1}^{d} \mathbf{x}[i]\mathbf{x}'[i]$. We also denote a matrix as $\boldsymbol{A}$ and the number of items a_i in a set as $\#a_{i}{}_{i=1}^{n} = n$.

Define $\mathcal{V}$ as a vocabulary containing words and serif symbols (e.g., w) as words. We denote a sequence of words $[\mathsf{w}_1, \mathsf{w}_2, , \mathsf{w}_n]$ as $\mathsf{w}_{1:n} = [\mathsf{w}_1, \mathsf{w}_2, , \mathsf{w}_n]$. For a given sequence $\mathsf{w}_{1:n}$, we describe that a word w is contained in $\mathsf{w}_{1:n}$ as $\mathsf{w} \in \mathsf{w}_{1:n}$. We also denote a sequence (or a text) as $\mathsf{text} = [\mathsf{w}_1, \mathsf{w}_2, , \mathsf{w}_n]$ shortly as long as no ambigulity involved.

2.2 Word Representations

2.2.1 One-hot Representation

One-hot representation or one-hot encoding is a word representation mapping a word $\mathsf{w} \in \mathcal{V}$ to a vector $\mathbf{w}$ filled with 0s except for a 1 at the position indicating the word w:

$$f_{\mathbb{1}} : \mathsf{w} \in \mathcal{V} \mapsto \mathbf{w}_{\mathbb{1}} = \begin{cases} 1 & (\text{if this basis indicates word } \mathsf{w}) \\ 0 & (otherwise) \end{cases}.$$

Note that $\mathbf{w}_{\mathbb{1}} \in \{0, 1\}^d$ where d is a vocabulary size $\#\mathcal{V}$. In general, one constructs the vocabulary $\mathcal{V}$ from a given corpus $\mathcal{C} = \{\mathsf{text}_i\}_i$, i.e, $\mathcal{V} = \bigcup_{\mathsf{text} \in \mathcal{C}} \{\mathsf{w} \mid \mathsf{w} \in \mathsf{text}\}$. We denote this vocabulary as $\mathcal{V}_{\mathcal{C}}$. Beacuse the vocabulary $\mathcal{V}_{\mathcal{C}}$ contains limited number of words, one-hot representation cannot represent out-of-vocabulary words $\mathsf{w} \notin \mathcal{V}_{\mathcal{C}}$.

In addition to one-hot representation, we also introduce Bag-of-Words (BoW) which is a text representation. because BoW is used to represent contexts in Section 3.5. BoW is defined as

$$f_{\text{BoW}}(\mathsf{text}) = \sum_{\mathsf{w} \in \mathsf{text}} f_{\mathbb{1}}(\mathsf{w}).$$

This representation contains frequencies of words only and ignores word orders in the text.

2.2.2 Pointwise Mutual Information

To represent words, we can exploit pointwise mutual information (PMI) [15, 9] which is a measurement between two discrete random variables $x \in X$ and $y \in Y$ defined as

$$\begin{aligned} \text{PMI}(x, y) &= ln \frac{p(x, y)}{p(x)p(y)} && (2.1) \\ &= ln \frac{p(x \mid y)}{p(x)} && (2.2) \\ &= ln \frac{p(y \mid x)}{p(y)} && (2.3) \end{aligned}$$

where p is a probability density function for the variables. In this thesis, we assume X is a corpus $\mathcal{C} = \{\mathsf{text}_i\}_i$, Y is a vocabulary $\mathcal{V} = \{\mathsf{w}_i\}_i$. In additon, we exploit a

collection of pairs $\mathcal{D} = \{(\mathsf{w}, \mathsf{text})_i\}_i$. Based on these notations, there are two way to calculate PMI over the $\mathcal{D}$; use Eq. (2.3) and Eq. (2.3).

In the first case, the $p(\mathsf{w})$ and $p(\mathsf{w} \mid \mathsf{text})$ are defined as:

$$p(\mathsf{w}) = \frac{\#\{(\mathsf{w}, \mathsf{text}') \mid (\mathsf{w}, \mathsf{text}') \in \mathcal{D}\}}{\#\mathcal{D}} \tag{2.4}$$

$$p(\mathsf{w} \mid \mathsf{text}) = \frac{\#\{(\mathsf{w}, \mathsf{text}) \mid (\mathsf{w}, \mathsf{text}) \in \mathcal{D}\}}{\#\{(\mathsf{w}', \mathsf{text}) \mid (\mathsf{w}', \mathsf{text}) \in \mathcal{D}\}} \tag{2.5}$$

respectively, where w' is a word in $\mathcal{V}$ and text' in a text in $\mathcal{C}$. We can view the latter equation. (2.5) calculates ratio of words in the given text in the similar manner of BoW because both representations focus on contents in texts.

In the second case, the $p(\mathsf{text})$ and $p(\mathsf{text} \mid \mathsf{w})$ are defined as:

$$p(\mathsf{text}) = \frac{\#\{(\mathsf{w}', \mathsf{text}) \mid (\mathsf{w}', \mathsf{text}) \in \mathcal{D}\}}{\#\mathcal{D}} \tag{2.6}$$

$$p(\mathsf{text} \mid \mathsf{w}) = \frac{\#\{(\mathsf{w}, \mathsf{text}) \mid (\mathsf{w}, \mathsf{text}) \in \mathcal{D}\}}{\#\{(\mathsf{w}, \mathsf{text}') \mid (\mathsf{w}, \mathsf{text}') \in \mathcal{D}\}} \tag{2.7}$$

respectively, where w' and text' are same of the first case. The latter equation. (2.7) in this case represents how the word depends on the texts, e.g., if two words appear in similar texts, probabilities of the words should be closed.

These representations based on PMI have benefits which allows us to compute OOV words. For example, the second case provides word representations composed of coocurrences between words and contexts. However, the representation has drawbacks which the dimensionality of representation vectors growthes with increasing a number of contexts.

2.2.3 Continuous Representations

In this section, we introduce standard continuous representations. Continuous representations [74, 89] are word representations mapping a word $\mathsf{w} \in \mathcal{V}$ with a continuous-valued vector $\mathbf{w}$:

$$f_{\mathbb{R}} : \mathsf{w} \in \mathcal{V} \rightarrow \mathbf{w} \in \mathbb{R}^d. \tag{2.8}$$

The continuous-valued vectors representing words are called *embedding vectors*. The embedding vectors allow us to embed various information, e.g., semantics of words, into the continuous values.

One standard way to exploit the continuous representation Eq .(2.8) is to map words to embedding vectors using a lookup table. We denote a matrix composed of embedding vectors as $\boldsymbol{E}$. Using the matrix $\boldsymbol{E}$ and one-hot representations $f_{\mathbb{1}}$, we can implement the function $f_{\mathbb{R}}$ as lookup table $\boldsymbol{E}f_{\mathbb{1}}(\mathsf{w})$, i.e., $f_{\mathbb{R}}(\mathsf{w}) = \boldsymbol{E}f_{\mathbb{1}}(\mathsf{w})$. Note that this function requires predefined vectors to represent words because only corresponding vectors can represent the words. Moreover, many word repressentations such as modeling polysemy employ the formulation mapping words to correspond representations deterministically.

The differences between continuous representations come from objective functions, optimization methods, and initial values of model parameters. For example, previous work proposes various objective functions and trains embedding vectors through the objective functions to capture different aspects of words. In the next section, we introduce two popular continuous representation models, continuous Bag-of-Words and skip-gram.

Continuous Bag-of-Words and Skip-gram

Continuous Bag-of-Words (cBoW) and *skip-gram* [74, 75] are continuous representation models based on the following objective functions:

cBoW

$$
\begin{aligned}
p(\mathsf{w} \mid \mathsf{text}) &= \frac{\exp\langle \mathbf{w}, \mathbf{h} \rangle}{\sum_{\mathsf{w}' \in \mathcal{V}} \exp\langle \mathbf{w}', \mathbf{h} \rangle} \\
\mathbf{h} &= \frac{1}{n} \sum_{\mathsf{c} \in \mathsf{text}} \mathbf{c}
\end{aligned}
$$

skip-gram

$$
\begin{aligned}
p(\mathsf{text} \mid \mathsf{w}) &= \prod_{\mathsf{c} \in \mathsf{text}} p(\mathsf{c} \mid \mathsf{w}) \\
p(\mathsf{c} \mid \mathsf{w}) &= \frac{\exp\langle \mathbf{c}, \mathbf{w} \rangle}{\sum_{\mathsf{w}' \in V} \exp\langle \mathbf{c}, \mathbf{w}' \rangle}
\end{aligned}
$$

where c_i is a word $\mathsf{c}_i \in \mathcal{V}$ and $\mathsf{text} = [\mathsf{c}_1, , , \mathsf{c}_n]$ is a context of the word w. We train embedding vectors to maximize the probability over a collection of pairs $\mathcal{D} = \{(\mathsf{w}, \mathsf{text})_i\}_i$. In the collection $\mathcal{D}$, a text text is a surrounding context of the word w,

e.g., text $=$ [I, like, such, as, apples] and w $=$ fruits for a given text [I,like,fruits, such,as, apples].

These two models are based on distributional hypothesis, stated in terms "a word is characterized by the company it keeps". Especially, the objective function of cBoW aims to predict a word using surrounding contexts of the word and to characterize embedding vectors using contexts as companions of the words. In addition, the work [62] reports the relationship between skip-gram and PMI: embedding vectors of skip-gram are corresponding to the results of PMI with little modifications.

These continuous representations are heavily benefitical, in fact, these models make a breakthrough not only in natural language processing but also in other domains such as information extraction. In spite of the benefits, there are many drawbacks in continuous representations such as the problem of out-of-vocabulary (OOV) words. For example, cBoW and skip-gram cannot represent words whose embedding vectors are not stored. Interestingly, word representations based on PMI can avoid the problem of OOV words. In PMI, we calculate word representations using contexts and the representations do not depend on predefined embedding vectors. These difference between skip-gram and PMI comes from whether the model represents words using predefined embedding vectors directly or represent words using weakly related objects. The observation implies the problem of OOV words arises from direct representation, which assign words to embedding vectors directly, and inspires *distant representation* proposed in this thesis.

Note that the distant representation provides solutions to dealing with polysemy of words. Both representations based on skip-gram and PMI face difficulties in dealing with polysemy of words. For example, we do not distinguish apple in two contexts [I, ate, an, apple] and[Jobs, founds, apple]. To develop more useful word representations, we need to explore novel semantcs representations and handling methods.

2.3 Modules in Neural Networks

2.3.1 Activation Functions

In this section, we introduce three activation functions, *sigmoid*, *tanh*, and *ReLu* [80], which are used to achieve non-linear transformation in neural networks. The

activation functions are defined as follows:

$$\begin{aligned} \text{sigmoid}(x) &= \frac{1}{1+e^{-x}} \\ \tanh(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\ \text{ReLU}(x) &= max(0,x) \end{aligned}$$

Note that we can transform the tanh function to the sigmoid function using scaling and shifting, i.e., $\text{sigmod}(x) = \frac{1}{2}\tanh(\frac{1}{2}x) + \frac{1}{2}$. In addition to these basic activation functions, many reseaches have proposed different activation functions [72, 54, 29, 48, 1] and studied the theoretical properties of the activation functions [78, 104]

2.3.2 Batch Normalization

Normalization is a linear transformation that convert a vector set $\mathcal{X} = \{\mathbf{x}_i\}_i^n$ into a set of new vectors $\mathcal{X}' = \{\mathbf{x}'_i\}_i^n$.

The normalization consists of two linear transformations called *centering* and *whitening*. Let μ be a mean vector $n^{-1}\sum_{i=1}^n \mathbf{x}_i$ and Σ be a covariance matrix $n^{-1}\sum_{i=1}^n (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$. Using these notations, linear transformations, centering and whitening, are defined as follows:

$$\begin{aligned} \text{centering}: \quad \mathbf{x}_i \mapsto \mathbf{x}_i - \mu &= \mathbf{x}''_i \\ \text{whitening}: \quad \mathbf{x}''_i \mapsto \boldsymbol{W}\mathbf{x}''_i &= \mathbf{x}'_i \end{aligned}$$

where the matrix $\boldsymbol{W}$ is a whitening matrix satisfying the condition $\boldsymbol{W}^T\boldsymbol{W} = \Sigma^{-1}$. The centerized vector set $\mathcal{X}'' = \{\mathbf{x}''_i\}$ has the 0 mean vector and the whitened vector set $\mathcal{X}' = \{\mathbf{x}'_i\}$ has the identity matrix as covariance matrix. The previous work [60] show these transformations improve model performances significantly.

Inspired by the normalization, *batch normalization* [45] provides two linear transformations, dynamic normalization BN_{dynamic} and static transformation BN_{static}, for a given mini-batch $\mathcal{X} = \{\mathbf{x}_i\}_i^n$. Batch normalization BN is a composite function of the two transformations, defined as:

$$\begin{aligned} N_{\text{dynamic}}(\mathbf{x}) &= \sigma_d \odot (\mathbf{x} + \mu_d) \\ N_{\text{static}}(\mathbf{x}) &= \sigma_s \odot (\mathbf{x} + \mu_s) \\ BN &= N_{\text{static}} \circ N_{\text{dynamic}} \end{aligned}$$

where $\odot$ and $\circ$ denote element-wise product and function composition, σ_d and μ_d are an element-wise variance and a mean of the given mini-batch $\mathcal{X}$, and σ_s and μ_s are model parameters. Note that these normalization uses element-wise variance σ_d where $\sigma_d[k] = n^{-1}\sum_{i=1}^{n}(\mathbf{x}_i[k] - \mu[k])^2$, instead of whitening matrices $\boldsymbol{W}$. In usual, calculations of the whitening matrices $\boldsymbol{W}$ for every mini-batches requires much computational cost. There are several modifications and extensitons of the batch-normalization [98, 96, 44, 4].

This batch normalization generalizes activation functions using shifting and scaling operations. For example, we can identify two activation functions, sigmoid and tanh, if we use batch normalizations before and after the activation function, i.e., $BN(f(BN(\mathbf{x})))$. This observation comes from the we can transform the two activation functions into the same form using shifting and scaling operations. Thus, batch normalization allows us to exploit generlized activation functions. In additon, several researches study abilities of neural networks from the viewpoint of mini-batch and batch normalization [21, 51, 12].

2.3.3 Residual Connection

Residual connection [36, 109, 105, 42] is a concept defined on a function $f : \mathcal{R}^d \to \mathcal{R}^d$ basically. Applying redisual connections to function $f(\mathbf{x})$, we can describe the function with residual connections as follows:

$$f(\mathbf{x}) + \mathbf{x}.$$

For example, if $f(\mathbf{x}) = ReLU(\boldsymbol{A}\mathbf{x} + \mathbf{b})$, the function with residual connection becomes $ReLU(\boldsymbol{A}\mathbf{x} + \mathbf{b}) + \mathbf{x}$ where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and $\mathbf{b} \in \mathbb{R}^d$. In the case the domain and region of function are not same, we use a rectangle matrix to convert input vectors, i.e., $f(\mathbf{x}) + \boldsymbol{A}'\mathbf{x}$. This residual connection can mitigate the effect of gradient explosion and gradient vanishing [87]. Moreover, recent work [69, 11] have proposed neraul networks exploiting residual connections in various ways.

2.4 Neural Network Architectures

2.4.1 Fully-Connected Neural Networks

A fully-connected neural network (FCNN) [5, 40] is a basic neural network architecture composed of functions called fully-connected layers. Both domain and range of

fully-connected layers are in vector spaces, i.e., $L_{\text{FCNN}} : \mathbb{R}^{d1} \rightarrow \mathbb{R}^{d2}$. More specifically, for a given input vector $\mathbf{x}$, a fully-connected layer is described as:

$$L_{\text{F}}(\mathbf{x}) = f(\boldsymbol{A}\mathbf{x} + \mathbf{b})$$

where $\boldsymbol{A}$ is a transition matrix $\boldsymbol{A} \in \mathbb{R}^{d1 \times d2}$, $\mathbf{b}$ is a bias vector $\mathbf{b} \in \mathbb{R}^{d2}$, and f is an activation function. In addition to the basic formulation, we can exploit various methods to modify the layers to solve tasks more effectively. For example, if the domain ans region of layers are same (i.e., $d1 = d2$), we can exploit residual connections and batch normalizations:

$$L_{\text{FCNN}}(\mathbf{x}) = BN(f((A)\mathbf{x} + \mathbf{b})) + \mathbf{x}$$

where BN is a batch normalization function. In this thesis, we use fully-connected layers whose domain and region are same, i.e., $d = d1 = d2$.

Using fully-connected layers, we define a whole architecture of FCNN as a composite function of different fully-connected layers:

$$\text{FCNN} = L_{\text{F}}^{T} \circ ... \circ L_{\text{F}}^{1}$$

where $\circ$ indicates function composition and T is a total layer size in this FCNN. We also describe the same architecture as following:

$$\mathbf{x}_{t+1} = L_{\text{F}}^{t+1}(\mathbf{x}_t)$$

where $\mathbf{x}_0$ is an input vector and $\mathbf{x}_t$ is an intermediate vector in layer t. As this formulation shown and definitions of fully-connected layers, only a fixed-dimensional vector is acceptable for FCNNs. This restriction is a crucial obstacle to deal with natural languages because we encounter variable length sequences naturally.

2.4.2 Recurrent Neural Networks

A recurrent neural network (RNN) is an architecture of neural network which allows us to deal with variable length sequences. In this section, we introduce a basic RNN model mapping a sequence of vectors to a new sequce of vectors. Let $\mathbb{X} = [\mathbf{x}_1, , \mathbf{x}_n]$ be a sequence of vectors and $\mathbb{X}_{i:j}$ be a sub-sequence $[\mathbf{x}_i, , \mathbf{x}_j] \in \mathbb{X}$. Using these notations, a layer in our recurrent neural network is defined as follows:

$$\begin{aligned} \mathbf{L}_R(\mathbb{X}) &= [\mathbf{x}'_1, , \mathbf{x}'_n] \\ \mathbf{x}'_{i+1} &= L_F(\mathbf{x}'_i, \mathbf{x}_{i+1}) \end{aligned}$$

where $L_F(\mathbf{x}'_i, \mathbf{x}_{i+1})$ is a transition function $L_F(\mathbf{x}'_i, \mathbf{x}_{i+1}) = f({}_1\mathbf{x}'_i +_2 \mathbf{x}_{i+1} + \mathbf{b})$ and $\mathbf{x}'_0$ is a model parameter vector. Because this function does not depend on the sequence lengths, many studies in natural language processing exploit recurrent neural networks to deal with long sequences such as whole sentences. Popular extension of recurrent neural networks are long short-term memory [38], bidirectional recurrent neural networks [100], and gated recurrent units [14].

2.4.3 Convolutional Neural Networks

A convolutional neural network (CNN) [58, 25] is a neural network architecture used in various domains including computer vision [102], speech signal processing [85], and natural language processing [26]. Due to the usefulness, many extension and modification to X have been proposed to resolve domain specific aims. Especially, we focus on CNNs mapping a sequence of vectors to a new sequence of vectors in this thesis.

A function called *convolution* or filter plays an important role in the CNN architecture. Let $\mathbb{X} = [\mathbf{x}_1, , \mathbf{x}_n]$ be a sequence of vectors and $\mathbb{X}_{i:j}$ be a sub-sequence $[\mathbf{x}_i, , \mathbf{x}_j] \in \mathbb{X}$. For a given sequence of vectors $\mathbb{X}$, we introduce a convolution layer composed of $2k$ matrices as follows:

$$L_{\mathrm{C}}(\mathbb{X}_{i-k:i+k}) = f(\sum_{j=-k}^{k} \boldsymbol{A}_j \mathbf{x}_{i+j} + \mathbf{b}) \tag{2.9}$$

where f is an activation function and k is window size, and $\boldsymbol{A}_i$ and $\mathbf{b}$ are same as fully-connected neural networks, i.e., $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$. Applying this convolution to the whole vector sequence $\mathbb{X} = [\mathbf{x}_1, , \mathbf{x}_n]$, we obtain outputs $\mathbb{X}' = [\mathbf{x}'_1, , \mathbf{x}'_n]$ where $\mathbf{x}'_i = L_{\mathrm{C}}(\mathbb{X}_{i-k:i+k})$. Note that out of range embedding vectors such as $\mathbf{x}_{-1}$ and $\mathbf{x}_{n+1}$ are model parameters. We can view this function works as convolution in the functional analysis due to the analogy between Eq. (2.9) and convolution on two functions f and g, $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$.

A whole structure of our CNNs is a composite function of filter functions:

$$\mathrm{CNN} = L_{\mathrm{C}}^T \circ ... \circ L_{\mathrm{C}}^1$$

where L_{C}^t maps a sequence of vectors $\mathbb{X} \in \mathbb{R}^{d \times n}$ to a new sequence of vectors $\mathbb{X}' \in \mathbb{R}^{d \times n}$. As one of the properties of convolution layers, the CNN architecture does not depend on the length n and then allows us to deal with variable length inputs. This prop-

erty is beneficial for natural language processing because variable length sequences such as sentences appear naturally.

From the viewpoint of n-gram, we can view the convolution $f(\sum_{j=-k}^{k} \boldsymbol{A}_j \mathbf{x}_{i+j} + \mathbf{b})$ attempt to represent n-gram feature in sequences. Suppose we have a sequence of embedding vectors $\mathbb{X} = [\mathbf{I}, \mathbf{like}, \mathbf{fruits}, \mathbf{including}, \mathbf{apples}]$ and obtain a new sequence of vectors $\mathbb{X}'$ applying a convolution L_{C}^{1} with $k = 1$. The $\mathbb{X}'$ contains vectors of trigram in the text $[\mathsf{I}, \mathsf{like}, \mathsf{fruits}, \mathsf{such}, \mathsf{as}, \mathsf{apples}]$, e.g., $\mathbf{I}, \mathbf{like}, \mathbf{fruits}$ where $\mathbf{I}, \mathbf{like}, \mathbf{fruits} = f(\boldsymbol{A}_{-1}\mathbf{I} + \boldsymbol{A}_0\mathbf{like} + \boldsymbol{A}_{+1}\mathbf{fruits})$. We can view first convolution L_{C}^{1} aims to capture trigram information and stacked convolutions aim to capture high-order semantics in given texts. In addition, the convolution layers have a characteristic point called *shift invariance*: sub-phrases have same meanings themselves even if they are contained in any texts viewed from natural language processing. This characteristic follows our intuition when we understand meanings of texts from words to whole texts.

2.5 Attention Mechanism

2.5.1 Softmax Function

Softmax function [79, 32, 128] is a categorical distribution which describes distribution over one-hot representation of k-possible discrete elements. Various multi-class classification tasks use the softmax function, e.g., to predict words for generating sequences. For a given input vector $\mathbf{x}$ and a vector set $\Theta = \{\theta_i\}_i$, the probability to predict the random variable (or the label) k is defined as

$$\begin{aligned} \mathrm{softmax}(\theta_k \mid \mathbf{x}, \Theta) &= \frac{exp\langle \theta_k, \mathbf{x} \rangle}{Z} \\ Z &= \sum_{\theta_i \in \Theta} exp\langle \theta_i, \mathbf{x} \rangle \end{aligned}$$

where θ_k and θ_i represent to corresponding random variable k and l respectively. We call Θ as *output vectors*. To predict labels with high accuracies, the $\mathbf{x}$ need to contain information about objects we want to classify and the θ_k should represent general concepts of the label k.

To generate words using neural networks, softmax function plays a crucial role predicting new words. Especially, we exploit output vectors to represent words we want to predict. Note that this vector set works as a lookup table in vanilla softmax function,

due to the same reason we have limited number of embedding vectors in continuous representation. The difference between embedding vectors and output vectors comes from their work. Embedding vectors aim to represent words computationally and work as vectors depicted arrows. In contrast, output vectors aim to represent regions we want to assign corresponds labels. The regions of output vectors depend on other output vectors because we need to compare the outcomes to decide the maximum outcome in softmax function.

2.5.2 Attention Mechanism

Attention mechanism [71, 126, 113] is a function calculating a vector $\mathbf{x_q}$ from a vector set $\mathcal{X} = \{\mathbf{x}_i\}_i^n$ depending on a query $\mathbf{q} \in \mathbb{R}^d$. We can describe the function as follows:

$$\begin{aligned} \text{attention}(\mathcal{X}, \mathbf{q}) &= \sum_{\mathbf{x} \in \mathcal{X}} \alpha_{\mathbf{x}} \mathbf{x} \\ \alpha_{\mathbf{x}} &= \frac{\exp f_\alpha(\mathbf{x}, \mathbf{q})}{\sum_{\mathbf{x}' \in \mathcal{X}} \exp f_\alpha(\mathbf{x}', \mathbf{q})} \end{aligned}$$

where f_α is a function calculating relatedness between vectors in $\mathcal{X}$ and the query vector II, i.e., $f_\alpha : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. There are two kinds of f_α, additive function $\langle \theta_x, \mathbf{x} \rangle + \langle \theta_q, \mathbf{q} \rangle$ and multiplicative function $\langle \mathbf{x}, \mathbf{q} \rangle$. The additive function is used in work, and the multiplicative function is also used in various work. We can view attention mechanism is a weighted sum of the given vector set $\mathcal{X}$ using softmax function to calculate the weight.

Attention mechanism is a kind of pooling function P which maps a set of vectors to a vector, i.e., $P : 2^{\mathbb{R}^d} \to \mathbb{R}^d$. In addition to attention mechanism, there are three pooling functions, defined as

$$\begin{aligned} P_{\text{max}}(\mathcal{X})[i] &= \max_{1 \leq k \leq n} \mathbf{x}_k[i] \\ P_{\text{average}}(\mathcal{X}) &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \end{aligned}$$

where $\mathbf{x}[i]$ indicates the i-th value in the vector $\mathbf{x}$. Note that attention mechanism is a parametrized average function because the differences between the two functions come from how to calculate weight. These pooling functions are useful in natural language

processing because standard loss functions including softmax function require the input is represented as a fixed-dimensional vector $\mathbf{x} \in \mathbb{R}^d$. To handle different length sequences, we apply pooling functions to map a vector set to a vector and then use fully-connected neural networks for the output of pooling functions.

Amongst the pooling functions, attention mechanism has an important characteristic point: we can select required information on demands. For example, we want to classify whether the given relation triplet $(\mathsf{apple}, \mathsf{is\text{-}a}, \mathsf{fruit})$ is correct or not, using the given text $[\mathsf{I}, \mathsf{like}, \mathsf{fruits}, \mathsf{such}, \mathsf{as}, \mathsf{apples}]$. In the text, the sub-sequence $[\mathsf{fruits}, \mathsf{such}, \mathsf{as}, \mathsf{apples}]$ helps to classify the triplet is correct, but $[\mathsf{I}, \mathsf{like}, \mathsf{fruits}]$ is not. Thus, for the classification, we want to focus on the former sub-sequence and ignore the latter one. Attention mechanism can implement this intuition by weighting corresponding embedding vectors.

Recently, a variant of attention mechanism, called multi-dimensional attention (MDA), has been proposed in the work [101]. Although domain of the MDA is a vector set, we also describe the multi-dimensional attention using query vectors $\mathbf{q}$ as follows:

$$\begin{aligned} \text{attention}_{multi}(\mathcal{X}, \mathbf{q}) &= \sum_{\mathbf{x} \in \mathcal{X}} \alpha_{\mathbf{x}} \odot \mathbf{x} \\ \alpha_{\mathbf{x}}[k] &= \frac{\exp f_{\alpha,k}(\mathbf{x}, \mathbf{q})}{\sum_{\mathbf{x}' \in \mathcal{X}} \exp f_{\alpha,k}(\mathbf{x}', \mathbf{q})} \end{aligned}$$

where $\alpha_{\mathbf{x}} \in \mathbb{R}^d$ and $f_{\alpha,k}$ is a score function specialized to the index k. Note that we can describe max function as following equation:

$$\max_{x \in X}(x) = \lim_{\tau \to \infty} \sum_{x \in X} \frac{\exp \tau x}{\sum_{x' \in X} \exp \tau x'} x$$

where $x_i \in \mathbb{R}$ and $X = \{x_i\}_i$. If the score function is $f_{\alpha,k}(\mathbf{x}, \mathbf{q}) = \tau \mathbf{x}[k]$, the attention_{multi} can approximate max pooling function, thus,

$$P_{\max}(\mathcal{X}) = \lim_{\tau \to \infty} \text{attention}_{multi}(\mathcal{X}, \mathbf{q}).$$

In addition, if the score function is $f_{\alpha,k}(\mathbf{x}, \mathbf{q}) = 0$, MDA become average pooling function because all weights are $\frac{1}{n}$ These observations implies that MDA is a parametrized max pooling function and average pooling function.

2.5.3 Pointing Mechanism

Generating sequences of words using softmax function suffers from the problem of out-of-vocabulary (OOV) words: there is no corresponding output vectors representing

OOV words.

Pointing mechanism [117, 37, 31] or copying mechanism aims to overcome the problem by calculating output vectors from related texts on the fly. For example, neural machine translation models face the situation source sentences contain rare words such as person's name and locations. Using vanilla softmax function described in Eq. (2.10), we cannot generate the rare words whose output vectors do not exist. Pointing mechanism take an approach to calculate word representation vectors using source sentences to represent the rare words.

For example, we have a source sentence $\mathsf{text} = [\mathsf{Jobs}, \mathsf{founds}, \mathsf{apple}, .]$ and a sequence of vectors $\mathbb{X} = [\mathbf{UNK}, \mathbf{founds}, \mathbf{apple}, .]$. Note that there is no embedding vector of Jobs and the word is mapped to $\mathbf{UNK}$ representing OOV words. Here, we apply CNNs to $\mathbb{X}$ and obtain a new sequence of vectors $\mathbb{X}' = [\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4,]$ where $\mathbf{w}_1$ does not represent OOV words any more. Pointing mechanism treats this representation $\mathbf{w}_1$ as the original word Jobs substitutively.

For a text text and a sequence of vectors $\mathcal{W}_{\text{text}}$ generated from the text, we can formulate this pointing mechanism as following:

$$\begin{aligned} \text{softmax}(\theta_k \mid \mathbf{x}, \Theta \cup \mathcal{W}_{\text{text}}) &= \frac{exp\langle \theta_k, \mathbf{x} \rangle}{Z} \\ Z &= \sum_{\theta_i \in \Theta \cup \mathcal{W}_{\text{text}}} exp\langle \theta_i, \mathbf{x} \rangle \end{aligned}$$

where $\theta_k \in \Theta \cup \mathcal{W}_{\text{text}}$ represents corresponding words. Although all word vectors $\mathcal{W}_{\text{text}}$ are used in this formulation, we can exploit word vectors of OOV words only. This approach inspires us to develop word representations based on the distributional hypothesis thoroughly: calculate representation vector of a word using surrounding context of the word.

Chapter 3

Entity Classification

3.1 Introduction

In recent years, continuous word representation models (e.g., [74, 88]) have proved their effectiveness in many natural language applications.

These models associate words with vectors in a continuous space. The vectors are trained on a corpus, but once the training is over, they are fixed. Using fixed vectors makes dealing with polysemy difficult in downstream applications. For a polysemous word, all its senses might be stored in the learned vector, but this is an optimistic expectation as the training procedure is not designed to take the presence of multiple senses into account. Moreover, even if all the necessary information is indeed present in the vector, a question remains as to how an appropriate sense can be extracted from the vector when a specific context of the word is given at runtime.

To address the problem of polysemy, some of the recent word representation models [95, 82, 41] learn multiple "prototype" vectors for a word. The idea is to have each prototype vector represent one of the senses the word carries. However, vectors are fixed after training as in single-vector word representation models. These models thus only partially solve the problem of polysemy, because they still do not provide a way to disambiguate a word according to a given context. Determining the number of senses (or prototype vectors) a word also poses a problem.

Another problem with fixed representations (including both single- and multiple-vector representations) is how to deal with *out-of-vocabulary* (OOV) words, which do not occur in the training corpus but are encountered at runtime. Because they are not known at the time of training, their representation vectors are missing. The problem is how to obtain representations for these words.

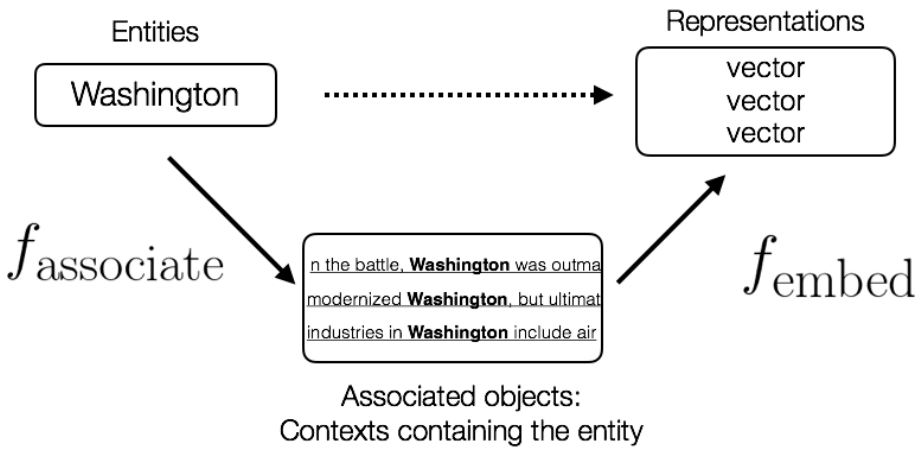


Figure 3.1: Scheme of Bag-of-Contexts representation

Several researchers have tackled the problem of OOV words in representation learning (e.g., [31, 30]; see also Section 3.2). Their methods calculate word representations from the context given at runtime. Another conceivable way is to retrieve relevant contexts of the OOV words from external resources on the fly. Using multiple contexts should bolster more accurate representations to be learned, but the previous methods assume a single context given at runtime, and do not provide a way to handle multiple contexts. To utilize contexts collected on the fly, the system must also cope with noise, because the collection might include the usage of the words that are irrelevant to the way they are used in the runtime context. This could be a problem especially when the words are polysemous.

To summarize, we observe two issues with existing word representation models: (i) The sense of a word is determined by a given context at runtime, but this aspect is not considered in the existing models. (ii) These models are unable to sift relevant representation out of multiple, possibly noisy, contexts observed/collected at runtime.

On the basis of these observations, we propose a word representation architecture called "Bag-of-Contexts" (BoC). We call it an architecture as opposed to representation because the proposal consists of not only multiple representations (*BoC representations*) but also a neural network (*aggregating network*) to adapt them to run-time task or context. It also involves training another neural network (*embedding network*) to convert raw context into a representation vector, to deal with OOV words. These networks and BoC representation are illustrated in Figure 3.1 and 3.2 schematically. Contexts of a target word are collected from a corpus, and each is converted to a vec-

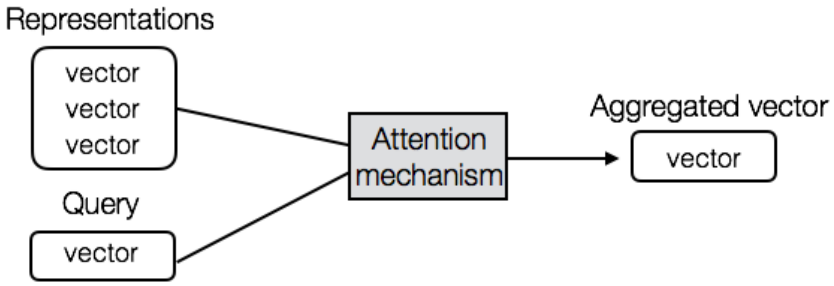


Figure 3.2: Disambiguatation scheme by Bag-of-Contexts representation

tor (context vector). This conversion is done by a neural network (which is trained alongside the context vectors), and it can be retained for later use to compute vectors for OOV words. The set of converted vectors constitutes a representation of the target word (BoC representation). When a query (i.e., a vector representing the runtime context/task) is given, an attention mechanism distills a suitable representation from the BoC representation.

To be more precise, BoC representations are generated in two steps: collecting contexts containing a word from corpora, and generating a vector for each collected context using the embedding network. The set of these vectors makes the representation of the word. Because a vector is associated with a context, there is no need to strictly determine the number of prototypes or senses; one only needs to collect a sufficient number of exemplar contexts to cover all the senses of a word. Multiple vectors associated with a word facilitate dynamic calculation of the representation for word in context. Specifically, this calculation is based on a neural attention mechanism [70] in the aggregating network: the runtime context or task that determines the sense of a word is represented as a vector ("query vector"), and on the basis of this vector, the aggregating network extracts relevant information or a sense from the multiple-vectors associated with a word.

For evaluation of the BoC architecture, a new dataset for an entity classification task is created and made publicly available (Section 3.4). Using this dataset, we show the effectiveness of the proposed model in Section 3.5.

3.2 Related work

We review past efforts on modeling polysemy, word sense disambiguation, and handling of OOV words in continuous word representation models. Work on weakly supervised learning is also explained as it bears similarity to our proposed architecture.

Modeling Polysemy Although studies on polysemy have a long history [19], we focus here on work that models polysemous senses by word representations.

The *prototype-based* approach [95, 3] addresses polysemous senses by representing typical senses of a word by multiple vectors called prototypes. This approach was further combined with a skip-gram model [82], a topic model [68], and other extensions [110, 41]. Its iterative application was proposed in [119]. There exists work [115, 3] that models senses as multivariate Gaussian distributions. Although these models use multiple vectors to represent a word, they are fixed after training; they do not address how to determine the sense that a word represents in a given context.

Another approach to polysemy is the *exemplar-based* approach, which is similar to our proposed method in that individual contexts of a word are kept without being merged into a smaller number of prototypes or predetermined senses. However, this approach was not well explored, because "[i]ndividual exemplars can be quite noisy and the model can incur high computational overhead" [95]. Moreover, work in this approach [112, 23, 94, 93] represented each context as "co-occurrence in context," not by vectors in continuous space. We can interpret our proposal as an exemplar-based model that exploits an attention mechanism to mitigate noisy collections, and recent progress in GPGPU to overcome high computational cost.

Word Sense Disambiguation Word sense disambiguation (WSD) is a long-studied field in natural language processing [61]. Earlier work on WSD represented words not as continuous vectors but as co-occurrence vectors [6, 27] or by a topic model [63, 120].

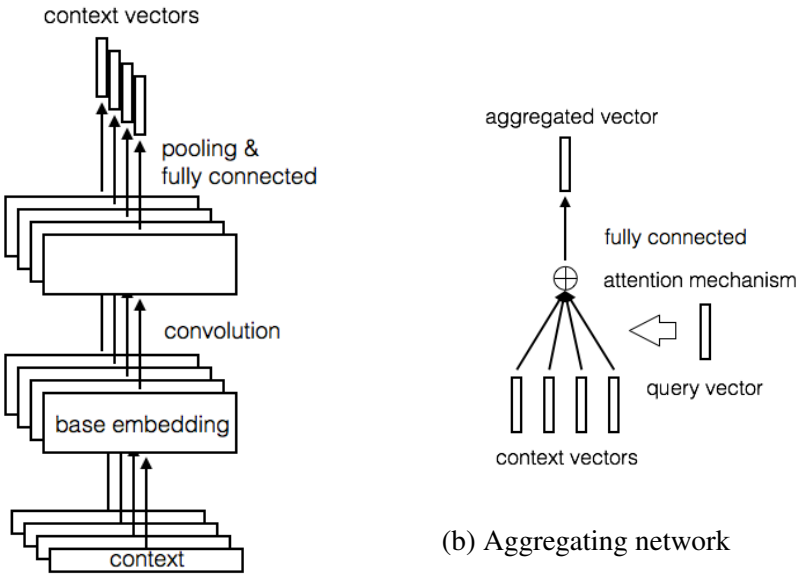
Following the recent successful results of neural network models in other areas, several researchers [130, 56, 92] have proposed WSD models that utilize continuous vectors. For example, [92] proposed recurrent neural network models and [13] used knowledge bases to tackle how to represent polysemous senses and disambiguate the polysemy at the same time.

Handling Out-of-Vocabulary Words In continuous word representation models, representation vectors need to be trained. If words are unavailable at the time of training, their representations are non-existent when they appear in downstream applications. This problem of out-of-vocabulary (OOV) words can be addressed in two ways: (i) using character-level information, and (ii) using sentences containing the OOV words. For the latter, the context can be either provided at runtime (if the task is WSD or similar tasks), or collecting contexts from an external corpus in an on-demand manner. In this thesis, we take the latter approach (but note that two are not mutually exclusive).

Models that explore sentences that contain target words have been proposed for question answering [37] and for general problems which demand to select one component of input sequence at each time step [117]. The same or similar idea has been used in various domains: machine translation [31], language models [30, 73], machine reading [55, 49], and question answering [37, 59, 111]. The idea shared in these models is to calculate word representations from additional texts such as source texts in machine translation and paragraphs related to answers in question answering. This procedure allows us to calculate representation vectors of OOV words. However, these methods do not provide how to deal with many contexts containing target words. Work in language modeling [30, 73] can deal with multiple occurrences of a word, but the score of the word is defined merely as the sum of the scores of individual occurrences.

Weak/Distant Supervision Distant supervision [16, 77] is a strategy to leverage large unlabeled corpora for relation extraction. Given a pair of entities, sentences (or passages) that jointly mention two entities are collected from a corpus. Although some (or most) of these sentences are likely to be irrelevant or merely noise, the assumption is that if a certain relation holds between two entities, it should be described in at least one of the collected sentences. Research in distant supervision includes treating collected sentences as an instance of multi-instance learning [10, 106], mitigating noisiness [108, 97], and extending distant supervision itself [90, 2, 34].

Weak supervision [39, 57, 43] is a broad term that covers various attempts to leverage large unlabeled corpora for relation extraction [16, 77, 106]. Along this line of research, work in image processing explores "weakly labeled" image data in the sense that each image is labeled with (possibly multiple) objects pictured in the image, but the exact segment of the labeled objects is not provided [114, 86, 91]. In our proposed architecture, the contexts for a (polysemous) word are collected, but the sense of the



(a) Embedding network

(b) Aggregating network

Figure 3.3: Illustration of embedding networks and aggregating networks used in experiments.

word in each context is not provided. It thus can be regarded as a form of learning from weakly labeled examples.

3.3 The Bag-of-Context Architecture

In this section, we illustrte the proposed BoC architecture in figure 3.3. The BoC architecture consists of two neural networks, *embedding network* and *aggregating network*. The embedding network converts each input context of a word to a "context" vector. The collection of the produced context vectors for a word is then regarded as its representation (BoC word representation). At runtime, the aggregating network takes two inputs, a "query" and the BoC representation of a word, and outputs a single vector that represents a suitable meaning of the word according to the query. The query is a vector representation of runtime context or task, which is not necessarily the same type of objects produced by the embedding network. Indeed, in the entity classification experiment in Section 3.5, we use (the vector encoding of) an entity type (*location*, *person*, etc.) as the query, which is a distinct object from the context vectors.

3.3.1 Embedding Networks

The embedding network computes the vector representations $\{\mathbf{c}_i\}_i$ of input contexts $\{\mathsf{c}_i\}_i, \mathsf{c} = [\mathsf{w}_1, \mathsf{w}_2, , , \mathsf{w}_n]$.

Suppose $\mathbf{c}$ consists of n words. First, each word w_j ($j = 1, \ldots, n$) in $\mathbf{c}$ is converted to a word embedding $\mathbf{w}_j$. Notice that this word embedding is different from the BoC representation of the target word that we are trying to compute; word embeddings $\{\mathbf{w}_j\}$ are the representation of words occurring in input contexts, and they are used only inside the embedding network to compute a context vector. They are initialized randomly and trained alongside the embedding network and the aggregating network. We call these embeddings *base embeddings*. Using base embeddings, we map a context to a list of vectors as

$$\text{base embed} : [\mathsf{w}_1, \mathsf{w}_2, , , \mathsf{w}_n] \mapsto [\mathbf{w}_1, \mathbf{w}_2, , , \mathbf{w}_n] \tag{3.1}$$

where $[\mathsf{w}_1, \mathsf{w}_2, , , \mathsf{w}_n]$ is a context explained above and $\mathbb{W} = [\mathbf{w}_1, \mathbf{w}_2, , , \mathbf{w}_n]$ is a list of mapped vectors. Although this mapping deals with OOV words as unknown word, a list of mapped vectors contains useful information composed of words contained in the vocabulary of base embeddings.

The embedding network can be further broken down into three neural network modules, as shown in Figure 3.3a. Firstly, we apply convolutional neural networks (1) explained in Section 2.4.3 to a list of vectors $\mathbb{X}$ generated from a context c:

$$\text{CNN} : \mathbb{X} \mapsto \mathbb{X}'. \tag{3.2}$$

where the CNN consists of convolution layers which calculate a output vector $\mathbf{x}'_i$ as $f(\sum_{j=-1}^{1} \boldsymbol{A}_j \mathbf{x}_{i+j} + \mathbf{b})$. This mapping aims to capture semantic information in the input context. Secondly, we convert a list of vectors $\mathbb{X} = [\mathbf{x}_1, \mathbf{x}_2, , , \mathbf{x}_n]$ to a vector $\mathbf{x}'$ using average pooling function (2) i.e., $\mathbf{x}' = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$. Finally, the output of average pooling function goes through fully-connected neural networks (3). After each of the contexts for a target word is converted to a context vector by the embedding network, we treat the resulting set of context vectors as the BoC representation of the target word.

Note that various neural network models can be used for individual modules in this architecture. For example, an elementwise max function can be used in place of the average pooling function. We explain the details of our architecture in Section 3.5.

3.3.2 Aggregating Networks

For a polysemous word, it is the surrounding context, not the word alone, that determines the sense of the word. For example, we know that the word *Washington* is used as a person if it is preceded by "*Jefferson met*." The aggregating network in our model is designed to carry out this process, and extracts suitable information from the bag of context vectors of a word.

The aggregating network takes two types of inputs: a BoC representation (of a target word), and a "query" vector representing runtime context/task. We call the surrounding context given at runtime *query* to clarify distinction from the context vectors in BoC representations. The aggregating network, after receiving a BoC representation and a query vector, outputs a single fixed-dimensional vector. The idea here is to train this network (and the embedding network as well) in a way that it can filter/distill from a collection of vectors (in the input BoC representation) a aggregated representation of the word that matches the runtime context/task represented by the query vector.

To this end, we use a neural attention mechanism [70]. Let $\{\mathbf{c}_i\}_{i=1}^{b}$ be the BoC representation of a word, consisting of b context vectors, and let $\mathbf{q}$ be a query vector. We assume context vectors $\mathbf{c}_i$ and query vector $\mathbf{q}$ have the same dimensionality. Using the query vectors, the attention mechanism calculates a vector by $\mathbf{x}_q = \sum_{i=0}^{b} \alpha_i \mathbf{c}_i$, where $\alpha_i = \exp\langle \mathbf{c}_i, \mathbf{q} \rangle / \sum_{j=0}^{b} \exp\langle \mathbf{c}_j, \mathbf{q} \rangle$. As depicted in Figure 3.3b, the vector $\mathbf{x}_q$ is further processed by a fully-connected NN to produce the final output vector of the aggregating network, which we call the *aggregated vector* of the word with respect to query $\mathbf{q}$.

Conversion to a aggregated vector, which is a vector of a fixed dimension, absorbs the difference in the number of context vectors in the BoC representations; note that words may have a different number of contexts in their BoC representations. A fixed-dimension vector also makes it easier to apply standard classifiers, such as logistic regression, in the downstream application.

3.4 Entity Classification Task

3.4.1 Task Settings

Entity classification is an information extraction task introduced in [67] and studied in [81, 127]. In this task, given an entity e and a type t, the system is required to classify whether entity e has type t or not, e.g., entity *Washington* has type *location*

Table 3.1: Statistic of the FRAGMENT datasets containing three different sizes: small, middle, and large.

	Small	Middle	Large
#texts	998,526	5,000,825	24,994,619
avg	13.2	47.9	185.3
#train	234,863	325,797	351,583
#dev	13,345	18,538	20,000
#test	13,314	18,576	20,000

or not. This is a task of multi-label binary classification; i.e., each entity may have multiple types. For example, entity *Metropolitan Museum* has five types including *building* and *organization* according to the FIGER entity classification dataset [67].

If we regard types as senses, the entity classification becomes a task well suited for evaluating how well word representation models can handle polysemy. It has two characteristics that makes it further suitable for evaluating word representation models. First, the classification of entity is expected to be solved by collecting contexts. Second, this task essentially demands classification of OOV words (entities), i.e., words that do not appear in the training data; in entity classification, the set of test entities that must be classified are disjoint from the set of words that occur in the training data. To classify OOV words, the contexts in which they occur are also provided (or collected from the corpus on the fly) at test time. These characteristics make the task suitable for evaluating our approach.

In this thesis, we use a plain text corpus to collect contexts of words. Previous work on entity classification is tightly coupled with a specific corpus or data resource, e.g., calculating an entity representation from the Wikipedia article for the entity [81], or exploiting the entity description in knowledge bases [125]. Although these resources are valuable, depending on them limits the applicability; if an entity does not have a Wikipedia article or if it is not present in a knowledge base, these methods are not applicable. We thus opt to use a plain textual corpus to collect contextual passages for words/entities.

Table 3.2: Comparison with baselines.

	Method	MicroF1	MacroF1	Perfect
Baseline	BoW	74.7	59.6	43.1
	Avg WE	70.6	49.7	39.6
Proposal	Max agg	81.1	57.0	55.5
	avg agg	81.5	53.2	55.6
	attention	**88.8**	**76.0**	**70.1**

3.4.2 Datasets for Entity Classification

Entity classification needs two resources, type database and text corpus. The type database provides gold entity-type pairs, and the text corpus provides the textual contexts of the entities. For the type database, there exist publicly available databases such as FIGER [67] and FIGMENT [127]. However, to the best of our knowledge, the textual corpora to be used with them are not standardized, which makes the comparison of entity classification systems/word representation models difficult on the same ground.

To improve the situation, we release a new free and open dataset called FRAGMENT. FRAGMENT contains not only a type database but also a collection of the textual contexts for entities. It provides three collections of different scales, small, middle, and large. The construction of FRAGMENT is described in Supplementary Material, and the statistics of the dataset are shown in Table 3.1. *#texts* and *avg* indicate numbers of texts and averaged numbers of texts per entities, and *#train*, *#dev*, *#test* indicate numbers of entities contained in. It can be downloaded from http://url.hidden.for.blind.review.net.

3.5 Experiments

Using the FRAGMENT dataset, we evaluate our proposed BoC architecture in entity classification.

3.5.1 Task Setting and Baseline

As explained in Section 3.4, previous methods for entity classification are tightly coupled with certain types of resources. The two previous work [81] and [125] assumes the existence of Wikipedia pages or descriptive sentences for entities in Free-

base. Since the main goal of our experiment is the evaluation of word/entity representations, we exclude these methods whose focus is to exploit existing resources and improve task performance. Moreover, it is not clear how to deal with cases when entities lack Wikipedia article or are not in Freebase.

As for word representation models, prototype-based models and existing models that handles out-of-vocabulary words mentioned in Section 3.2 are not applicable for entity classification, either; The former cannot handle OOV words because prototype vectors are fixed after training, and the latter cannot deal with multiple vectors.

Thus, we use two simple models as our baseline: Bag-of-Words (BoW) and averaged word embeddings (avg WE). The BoW model calculates feature vectors composed of frequencies of words appearing in contexts. The avgWE model represents an entity as an averaged vector of base embeddings in contexts. In this experiment, we use the top $10,000$ most frequency words for both BoW and avg WE, and 100 dimension size in avg WE.

3.5.2 Our Model

Architecture and Optimizations We use a neural network model shown in Figure 3.3 and described in Section 3.3. In the embedding network, we use a 5-layer convolutional neural network (CNN) in the first layers and a 3-layer fully-connected neural network (FCNN) in the second layers. In the aggregating network, we use a 3-layer FCNN to a aggregated vector. Every convolution and fully-connected network uses batch-normalization [45], ReLU activation function [28], and residual connection [36]. CNN in the word-level module has 1 stride filters mapping $(3, dim) \rightarrow (dim)$ in each layer. The dimension of all the vectors (depicted by circles in Figure 3.3) is fixed at 100. We train a base embedding for each of the $10,000$ words that has the highest frequency in the large-scale corpus, and one special embedding to represent all the remaining lower frequency words. To optimize all models in our experiments, we use Adam [53] with learning rate 0.001 and total training epoch 50. We also evaluate two variants of our model, using elementwise max-pooling and average-pooling in place of attention in the aggregating network

Sampling Contexts and Window Sizes In the FRAGMENT dataset we use for experiment, some words have a large number of contexts, and some of the contexts are quite long. Due to computational limitations, we reduce the number of contexts for a

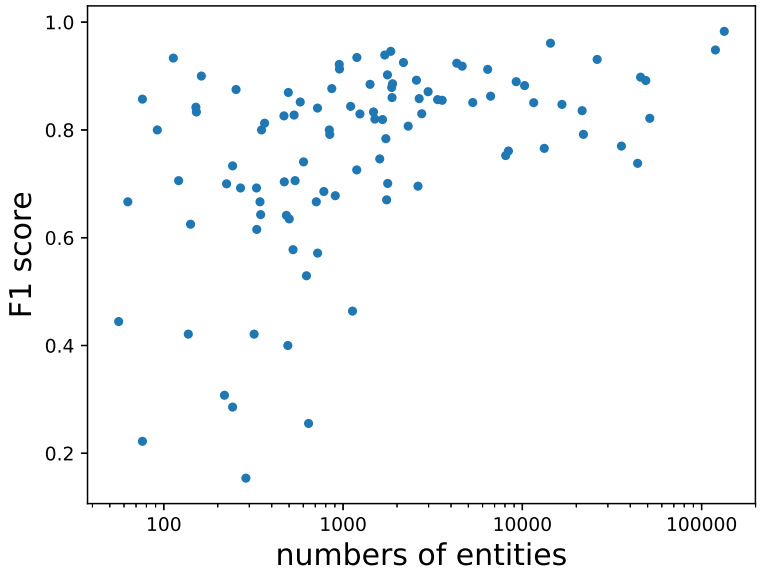


Figure 3.4: Relations between F1 scores of types and numbers of entities having the types.

word (bag size) by sampling. We also trim all contexts within a window size around the target words. We set the bag size and the window size randomly at the time of training. Let $\mathcal{U}(a, b)$ denote a discrete uniform distribution over {a, a+1, ..., b}. We sample the bag size from $\mathcal{U}(10, 30)$ and the window size from $\mathcal{U}(10, 20)$ in training, but fix the bag size to 100 and the window size to 20 in test. If the randomly selected bag size exceeds the number of contexts available, we use all the contexts. Likewise, if the sampled window size exceeds the length of a context, the entire context is used.

Result and Error Analysis The results are shown in Table 3.2. The figures below *Perfect* indicate the percentage of entities for which all gold types are classified correctly. Bold and underlined figures are respectively the best and second best scores for each evaluation. The evaluation metric *Perfect* indicates the percentage of entities that received perfect prediction; i.e., all the types an entity has are correctly identified. This metric has been used in previous work on entity classification. Our model using attention outperforms all other models considerably. Max and average models show better scores than the baselines but perform poorer than the proposed model using attention.

Let us analyze the relations between F1 score for individual types and the number of entities having the types. In the scatter plot of Figure 3.4, each point represents a type, with y-axis representing the F1 value for the type, and x-axis representing the number of entities having the type. The figure shows a tendency that the F1 value increases with the number entities that have the type. For example, the type *livingthing-animal* appears in 287 entities, and has a 15.3 F1 value. In contrast, the type *person* appears

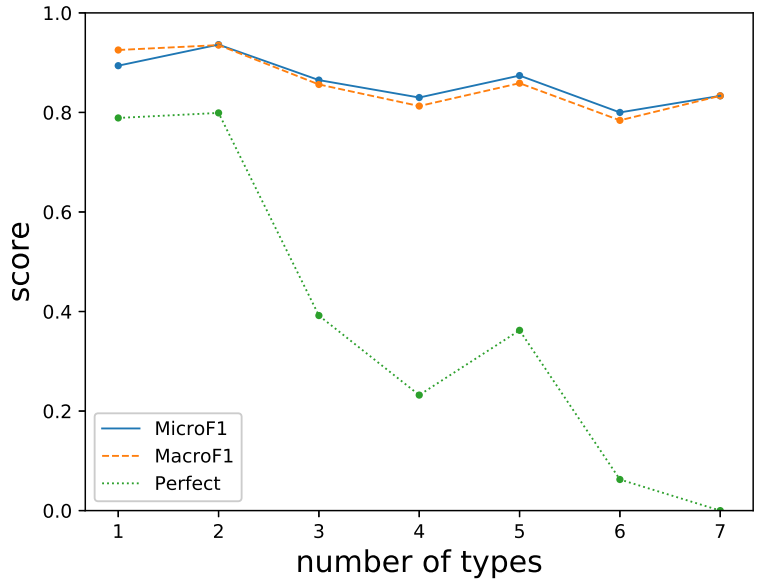


Figure 3.5: MicroF1, MacroF1, and Perfect scores with different type numbers are illustrated by solid, dash, dotted lines respectively.

in 133, 797 entities and its F1 is 98.3.

We also analyze the prediction accuracy in terms of the number of types an entity has. The result is shown in Figure 3.5. Minimum, maximum and averaged type numbers are 1, 7, and 1.9 in this dataset. In the FRAGMENT dataset, the minimum number of types an entity has is 1 and the maximum is 7. As the figure shows, micro F1 and macro F1 scores remain high even for entities with a large number of types. The "perfect" index drops with the number of types, but this is expected because perfect prediction (for all types an entity has) becomes more difficult with the increased number of types.

3.5.3 Construction of Bag-of-Contexts

In this section, we investigate how properties of Bag-of-Contexts affect model performance. In particular, we carry out two experiments with focusing on the number of contexts in the dataset, window size and bag size.

Number of Contexts in Datasets

Using datasets with different sizes, we evaluate how the number of contexts contained in the datasets affects model performance. The results are shown in Table 3.3. The *avg contexts* indicates the averaged number of contexts per entity. The table clearly

Table 3.3: Comparison among datasets of different scales.

Datasets	MicroF1	MacroF1	Perfect	Avg contexts
Small	79.3	59.2	53.4	13.2
Middle	85.1	68.6	63.4	47.9
Large	88.7	73.9	69.9	185.3

shows that increasing the number of contexts improves the model performances. These results are favorable for applications because collecting contexts is relatively easy compared with retraining.

Window Size and Bag Size

Using the middle-scale dataset, we now examine how the window size and the bag size affect the performance of the model. The setting is as follows: In training time, a total of nine different combinations of window and bag sizes are tried. To be precise, we vary the window size sampled from $\mathcal{U}(5, k)$ among $k \in \{5, 10, 20\}$, combined with the bag size sampled from $\mathcal{U}(5, m)$ with $m \in \{5, 10, 20\}$. In Table 3.4, the maximum value of the sampling domain (k and m in the above formulas) are shown in the *window* and *bag* columns. During testing, we fixed the window size to the k value above used for sampling the window sizes in training. The bag size for test examples is set to 100 regardless of the parameter used in training.

All results are shown in Table 3.4. We see that increasing the window size improves the classification scores. The improvement is greater when the window sizes is increased from 5 and 10 than from 10 to 20, which is expected as the relevant information must be in the vicinity of the target word. In contrast to the window size, the bag size does not affect the model performance.

3.6 Construction of FRAGMENT datasets

In this appendix, we explain how to create the FRAGMENT datasets. Our FRAGMENT datasets use two large-scale open datasets in natural language processing, Wikipedia https://dumps.wikimedia.org/ and Freebase https://developers.google.com/freebase/, for textual and type resources respectively.

Table 3.4: Comparison between different window sizes and bag sizes. Each value indicates maximum values of window size or bag size in training time.

window	bag	MicroF1	MacroF1	Perfect
5	5	81.7	61.9	57.4
5	10	81.4	62.0	57.1
5	20	80.9	60.4	55.3
10	5	84.4	68.4	62.3
10	10	84.5	65.0	61.6
10	20	84.8	66.9	61.7
20	5	85.6	67.8	64.3
20	10	85.2	66.2	62.4
20	20	85.4	67.9	64.0

3.6.1 Creating textual resources

we extract texts composed of natural languages from raw Wikipedia structured as HTML. We use scripts provided from MIT media-lab http://medialab.di.unipi.it/wiki/Wikipedia_Extractor in this procedure.

We annotate entities in extracted texts using DBpedia-Spotlight https://github.com/dbpedia-spotlight/dbpedia-spotlight. Parameters of DBpedia-SpotLight are "confidence": 0.5 and "support":10. Because each entity is labeled as DOI in this annotations, we collect alignment from DOI to MID, which used in Freebase. In particular, we exloit sites http://dbpedia.org/property/example containing corresponding MID of DOI *example*. We discard annotations if the correspondences are not found. We apply Spcay https://spacy.io/ to segemente the extracted texts and filter out the texts contain broken annotations.

3.6.2 Combining Type Resources

As type resources, we use knowledge relations contained in Freebase. Fortunately, previous work provide several type resourcees derived from Freebase, FIGER https://github.com/xiaoling/figer in [67] and FIGMENT https://github.com/yyaghoobzadeh/figment in [127]. In particular, we exploit type resources in FIGMENT dataset composed of four type resources, train, dev, test, and non-obj. We merge all entity-type pairs in the files in a file and then filter out entities not contained in the textural resources and texts in textual resource not containing any entity. After the filtering, we

split the selected entities into train, dev, and test randomly. In particular, dev and test contain $20,000$ entities and train contains remains entities.

3.6.3 Small, Middle, and Large Scale Datasets

We call the dataset explained above as *large-scale* dataset. In addition, we also create small and middle-scale datasets sampled from the large-scale dataset. In particular, we randomly choose a one-fifth part of texts in the large dataset for middle datasets and a one-fifth part of texts in the middle dataset for small datasets. Because this sampling reduces texts, we filter out entities not contained in sampled texts again.

3.7 Conclusion

In this chapter, we have proposed an exemplar-based word representation called "Bag-of-Contexts" (BoC). At training time, each context of a word collected from a corpus is converted to a vector, and they are collectively used as a representation (BoC representation) of the word. When a word is given for testing in a specific context, relevant information is distilled from the BoC vectors of the word through the attention mechanism. This approach provides a natural way of dealing with polysemy and out-of-vocabulary words. We have verified the effectiveness of the proposed model in the task of entity classification, which is essentially the task of classifying out-of-vocabulary words.

Chapter 4

Knowledge Base Completion

4.1 Introduction

Knowledge bases such as WordNet [76] and Freebase [7] are used for many applications including information extraction, question answering, and text understanding. These knowledge bases can be viewed as a set of *relation triplets*, i.e., triplets of the form (h, r, t) with an entity h called the *head entity*, a relation r, and an entity t called the *tail entity* [8, 83]. Some examples of relation triplets are (*Philip-K.-Dick*, *write*, *Do-Androids-Dream-of-Electric-Sheep?*) and (*Do-Androids-Dream-of-Electric-Sheep?*, *is-a*, *Science-fiction*). Although a knowledge base contains millions of such triplets, it is known to suffer from incompleteness [84]. Knowledge base completion (KBC) thus aims to predict the information missing in knowledge bases.

In recent years, *embedding-based* KBC models have been successfully applied to large-scale knowledge bases [8, 83, 33]. These models build the distributed representations (or, *vector embeddings*) of entities and relations observed in the training data to predict missing relation triplets. This continuous valued representation allows us to deal with information we cannot represent as discrete variables [46, 124], and use various vector operations over the embeddings [33, 83]. Although continuous valued representation has these benefits, it is known to suffer from several problems.

In this thesis, we address the *out-of-knowledge-base* (OOKB) entity problem in embedding-based KBC. To keep the value of knowledge bases, we need to update knowledge bases by adding not only lacked triplets about known entities but also triplets about new entities not stored in the current knowledge bases. We call the newly encountered entities as OOKB entities. The problem we address arises when OOKB entities occur in the triplets that are given to the system *after* training. As these entities

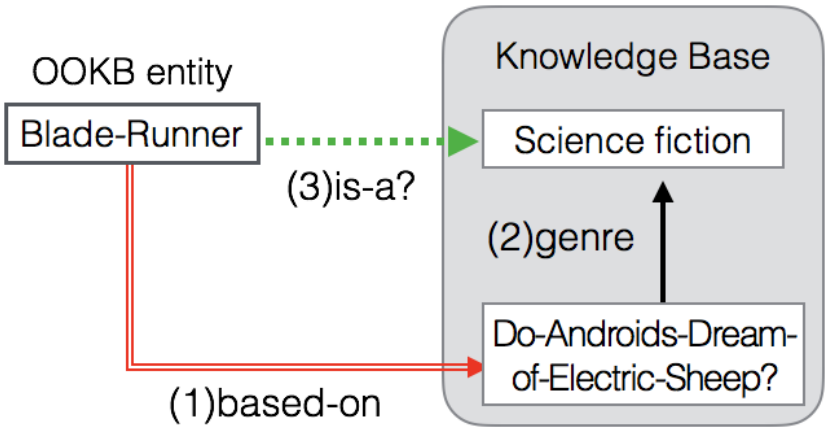


Figure 4.1: Illustration of OOKB entity problem.

were unknown to the system at training time, the system does not have their embeddings, and hence does not have a means to predict the relations for these entities. The OOKB entity problem thus asks how to perform KBC involving such OOKB entities. To solve KBC with OOKB entities, we need to calculate representation vectors of the OOKB entities.

Previous work for the OOKB entity problem takes an approach using texts to calculate representation vectors. For example, the work [121, 24] uses corpus related with OOKB entities such as Wikipedia, and the work [125] uses description about entities contained in Freebase to obtain representation vectors of related OOKB entities. Although this approach is benefitical, the textual resources have limitations to solve the OOKB entity problem; standard corpora do not contain any entity including highly specialized entities, and descriptions in knowledge bases are costly and unavailable for new-born entities. For these reason, we tackle the OOKB entity problem without textual resources.

We explain concrete problem setting using figure 4.1 illustrating an example of KBC with OOKB entity schematically. Suppose we find an OOKB entity "*Blade-Runner*" and want to complete relation triplets about this entity. For example, we want to answer the question, (Blade-Runner, is-a, Science-fiction), i.e., whether (3) the green dashed arrow in the figure should be drawn. To estimate answer of this question, we assume a new auxiliary triplet (1) depicted as a red double arrow is given in test time. This triplet is (*Blade-Runner*, *based-on*, *Do-Androids-Dream-of-Electric-Sheep?*), which connect the OOKB entity *Blade-Runner* and the entity *Do-Androids-Dream-of-Electric-Sheep?* stored in the knowledge graph (shown as the shaded box). The task is to tell whether

any other relations hold between *Blade Runner* and the entities in the knowledge graph, by using (1) the new triplet and (2) the existing triplet, depicted by a black arrow. Previous work cannot answer the question about OOKB entities in this setting, because there is no textual resources.

To solve the OOKB entity problem, we apply *graph neural networks* (GNNs) [99, 64] to a *knowledge graph*, which is a graph obtained by regarding entities as nodes and triplets as edges. A GNN is a neural network architecture defined on a graph structure and composed of two models called the *propagation model* and *output model*. The propagation model manages how information propagates between nodes in the graph. In the propagation model, we first obtain the embedding vectors of the neighborhood of a given node (entity) e and then convert these vectors into a representation vector of e using a pooling function such as the average. In other words, each node is embedded as a vector in continuous space, which is also used to calculate the vectors of the neighborhood nodes. This mechanism enables the vector for an OOKB entity to be composed from its neighborhood vectors at test time. The output model defines the task-oriented objective function over node vectors. This allows us to use an existing embedding-based KBC model as the output model. In this thesis, we use TransE [8] as the output model, but we can adopt other embedding-based KBC methods.

Our main contributions can be summarized as follows:

- We propose a new formulation of the problem of OOKB entities in KBC.
- We propose a GNN suitable for the KBC task with OOKB entities.
- We verify the effectiveness of our models in both the standard and the "OOKB" entity settings.

4.2 Related work

Recently embedding based models are widely studied in knowledge base completion. For example, TransE [8] train embedding vectors of head entity $\mathbf{v}_h$, relation $\mathbf{v}_r$, and tail entity $\mathbf{v}_t$ to hold the equation $\mathbf{v}_h + \mathbf{v}_r = \mathbf{v}_h$ for the correct triplet (h, r, t). There are various embedding based models such as restricting embedding space to capture relations [122], representing relations as matrices [65], using kernel methods [123], and combining generative method [124], and so on.

There are few work tackling OOKB entity problem. The work [121, 125, 24] uses language models including skip-gram to train word representation vectors we

encounter as OOKB entities potentially. The work [125] exploits descriptions about entities contained in Freebase to calculate representation vectors dynamically. Note that the common approach of these work is to use textual resources for OOKB entities.

4.3 Knowledge Base Completion with Out-of-Knowledge-Base Entity

4.3.1 Knowledge Graph

Let $\mathcal{E}$ be a set of *entities* and $\mathcal{R}$ be a set of *relations*. Define the *fact*, or *relation triplet*, to be a triplet of form (h, r, t) where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$.

Let $\mathcal{G}_{\text{gold}} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ be the set of *gold* facts, i.e., the set of all relation triplets that hold for pairs of entities in $\mathcal{E}$ and relations in $\mathcal{R}$. If a triplet is in $\mathcal{G}_{\text{gold}}$, we say it is a *positive triplet*; otherwise, it is a *negative triplet*. The goal of knowledge completion is to identify $\mathcal{G}_{\text{gold}}$, when only its proper subset, or an "incomplete" *knowledge base*, $\mathcal{G} \subset \mathcal{G}_{\text{gold}}$ is accessible.

A knowledge base $\mathcal{G}$ is often called *knowledge graph* because each triplet in $\mathcal{G}$ can be regarded as a (labeled) edge in a graph; i.e., the entities in a triplet correspond to the end nodes and the relation gives the label of the edge.

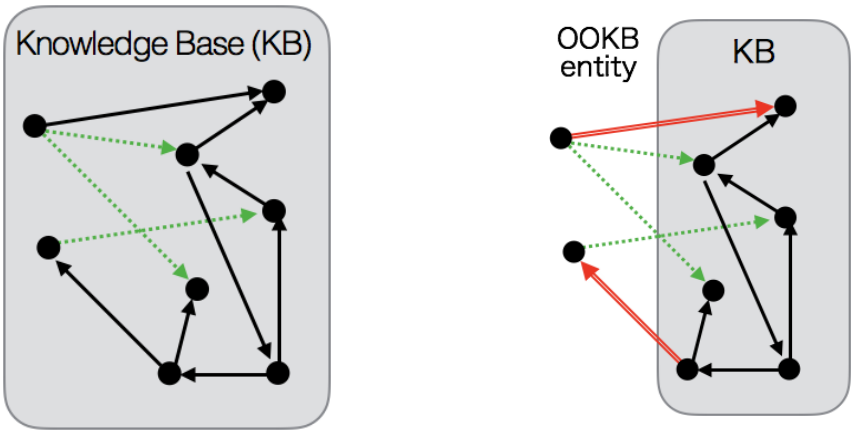
4.3.2 KBC: Triplet Classification

Triplet classification is a typical KBC task introduced by [103] [1] and has since been a standard benchmark for KBC methods [122, 46, 123, 129, 83].

In this task, existing knowledge base $\mathcal{G}$ is assumed to be incomplete, in the sense that some of triplets that must be present in $\mathcal{G}$ are missing; i.e., $\mathcal{G} \neq \mathcal{G}_{\text{gold}}$.

Let $\mathcal{H} = (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \backslash \mathcal{G}$ be the set of triplets not present in $\mathcal{G}$. Because $\mathcal{G}$ is incomplete, two cases are possible for each triplet $x \in \mathcal{H}$; either x is a positive triplet (i.e., $x \in \mathcal{G}_{\text{gold}}$), or x is a negative triple (i.e., $x \notin \mathcal{G}_{\text{gold}}$). For the former case, x is not in $\mathcal{G}$ only because of the incompleteness, and the knowledge base must be updated to contain x. We thus encounter the problem of determining which of the above two possible cases each triplet not present in $\mathcal{G}$ falls into. This problem is called *triplet classification*.

[1] A similar task had been around for general graphs under the name of *link prediction*.



(a) Standard KBC

(b) KBC with OOKB entity

Figure 4.2: Illustration of standard KBC (left) and KBC with OOKB entity (right).

Viewed as a machine learning problem, triplet classification is a classifier induction task in which $\mathcal{E}$ and $\mathcal{R}$ are given, and knowledge base $\mathcal{G}$ forms the training set (with only positive examples), with $\mathcal{H}$ being the test set. The set $\mathcal{H}$ can be divided into the set of positive test examples $\mathcal{H} \cap \mathcal{G}_{\text{gold}} = \mathcal{G}_{\text{gold}} \backslash \mathcal{G}$ and the set of negative test examples $\mathcal{H} \backslash \mathcal{G}_{\text{gold}}$.

In the standard triplet classification, $\mathcal{E}$ and $\mathcal{R}$ are limited to the entities and relations that appear in $\mathcal{G}$. That is, $\mathcal{E} = \mathcal{E}(\mathcal{G})$ and $\mathcal{R} = \mathcal{R}(\mathcal{G})$, where $\mathcal{E}(\mathcal{G}) = \{h \mid (h, r, t) \in \mathcal{G}\} \cup \{t \mid (h, r, t) \in \mathcal{G}\}$ and $\mathcal{R}(\mathcal{G}) = \{r \mid (h, r, t) \in \mathcal{G}\}$ denote the entities and relations appearing in $\mathcal{G}$, respectively.

4.3.3 OOKB entity problem without texts

We now introduce a new task in KBC involving OOKB entities. As we explained in Section 4.1, we want to solve knolwedge base completion involving OOKB entities without texts. This setting is a kind of triplet classification, because we do not use textual resources. The differences between standard KBC and our KBC is existance of new entities and related auxiliary knowledge. We clarify the difference bewteen standard KBC setting and KBC with OOKB entities using figure 4.2a and figure 4.2b respectively. We explain problem setting below using each images.

In addition to the knowledge base $\mathcal{G}$ observed at training time, we assume new triplets $\mathcal{G}_{\text{aux}}$ are given at test time. The $\mathcal{G}$ and triplets we want to classify in test time

are depicted as black arrows and green dashed arrows respectively in the both of figure 4.2a and figure 4.2b. $\mathcal{G}_{\text{aux}}$ contains new entities $\mathcal{E}_{\text{OOKB}} = \mathcal{E}(\mathcal{G}_{\text{aux}}) \backslash \mathcal{E}(\mathcal{G})$, but no new relations are involved $\mathcal{R}(\mathcal{G}_{\text{aux}}) \subseteq \mathcal{R}(\mathcal{G})$. We call new entities $\mathcal{E}_{\text{OOKB}}$ as out-of-knowledge-base entity (OOKB entity) and $\mathcal{G}_{\text{aux}}$ as auxiliary triplets. In the figure 4.2b, OOKB entities are depicted as black circles not contained in the kwoledge base shown as the shaded box. It is assumed that every triplet in $\mathcal{G}_{\text{aux}}$ contains exactly one OOKB entity from $\mathcal{E}_{\text{OOKB}}$ and one entity from $\mathcal{E}(\mathcal{G})$; that is, the additional triplets $\mathcal{G}_{\text{aux}}$ represent edges bridging $\mathcal{E}(\mathcal{G})$ and $\mathcal{E}_{\text{OOKB}}$ in the combined knowledge graph $\mathcal{G} \cup \mathcal{G}_{\text{aux}}$. In the figure 4.2b, $\mathcal{G}_{\text{aux}}$ is depicted as double red arrows connecting $\mathcal{E}_{\text{OOKB}}$ and known entity $\mathcal{E}(\mathcal{G})$.

In this setting, $\mathcal{E} = \mathcal{E}(\mathcal{G}) \cup \mathcal{E}_{\text{OOKB}} \neq \mathcal{E}(\mathcal{G})$, and the task is to correctly identify missing relation triplets that involve the OOKB entities $\mathcal{E}_{\text{OOKB}}$. Because the embeddings for these entities are missing, they must be computed from those for entities in $\mathcal{G}$. In other words, we want to design a model by which the information we already have in $\mathcal{G}$ can be transferred to OOKB entities $\mathcal{E}_{\text{OOKB}}$, with the help of the added knowledge $\mathcal{G}_{\text{aux}}$. This requirement for calculating representation vectors on demand is the difference between standard KBC and KBC with OOKB entity, although available datum in training time are same in both settings.

4.4 Proposed Model

4.4.1 GNNs

Graph neural networks (GNNs) are neural networks defined on a graph structure. We use GNNs proposed in [99, 64], which consists of the propagation model and the output model. Although there exist GNNs based on convolution operation [20], we use GNNs composed of two models because convolution based GNNs cannot deal with large scale graph such as knowledge bases.

To solve OOKB entity problem, we need to calculate representation vectors of OOKB entities and inference using the obtained vectors. The motivation to use GNNs composed of two models is the propagation model and the output model are applicable for the calculation and the inference respectively. In particular, the propagation model allows us to calculate representation vectors of an entity only using representation vectors of neighborhood entities. Based on the calculated representation vectors, the output model aims to calculate score values to classify whether given triplets are correct

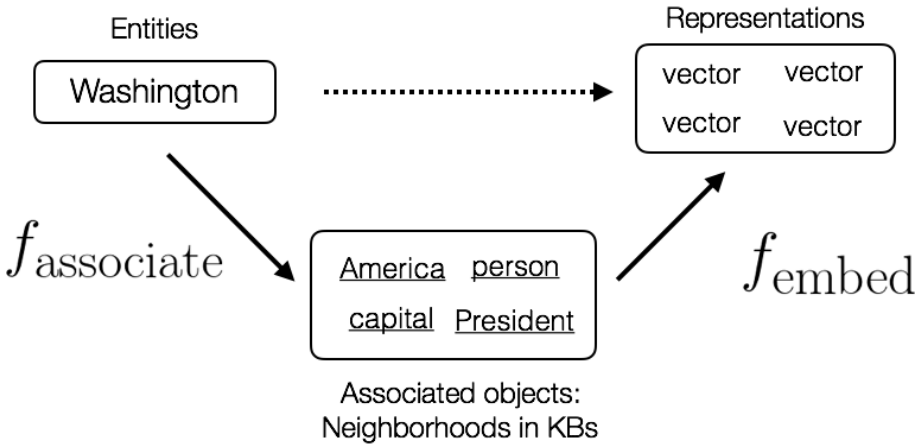


Figure 4.3: Scheme of relational approach of distant representation using graph neural network

or not. To the best of our knowledge, there is no work applying the GNNs to knowledge base completion task with OOKB entities. Thus, we need to investigate how to use and/or modify the GNNs. As explained below, we propose GNNs composed of generalized propagation model and modified output model to solve knowledge base completion more effectively.

4.4.2 Propagation Model on a Knowledge Graph

Let $\mathcal{G}$ be a knowledge graph, $e \in \mathcal{E}(\mathcal{G})$ be an entity, and $\mathbf{v}_e \in \mathbb{R}^d$ be the d-dimensional representation vector of e. Li et al. define the propagation model by the following equation [64]:

$$\mathbf{v}_e = \sum_{(h,r,e)\in\mathcal{N}_{\text{head}}(e)} T_{\text{head}}(\mathbf{v}_h; h, r, e) + \sum_{(e,r,t)\in\mathcal{N}_{\text{tail}}(e)} T_{\text{tail}}(\mathbf{v}_t; e, r, t), \tag{4.1}$$

where head neighborhood $\mathcal{N}_{\text{head}}$ and tail neighborhood $\mathcal{N}_{\text{tail}}$ are $\mathcal{N}_{\text{head}}(e) = \{(h, r, e) \mid (h, r, e) \in \mathcal{G}\}$ and $\mathcal{N}_{\text{tail}}(e) = \{(e, r, t) \mid (e, r, t) \in \mathcal{G}\}$ in a knowledge graph $\mathcal{G}$, respectively. In addition, $T_{\text{head}}, T_{\text{tail}} : \mathbb{R}^d \times \mathcal{E}(\mathcal{G}) \times \mathcal{R}(\mathcal{G}) \times \mathcal{E}(\mathcal{G}) \to \mathbb{R}^d$ are called *transition functions* [64] and used to transform the vector of a neighbor node for its incorporation in the current vector $\mathbf{v}_e$, depending on the property of the edge between them.

We generalize the propagation function using the following *pooling function* P:

$$S_{\text{head}}(e) = \{T_{\text{head}}(\mathbf{v}_h; h, r, e) \mid (h, r, e) \in \mathcal{N}_h(e)\}, \tag{4.2}$$

$$S_{\text{tail}}(e) = \{T_{\text{tail}}(\mathbf{v}_t; e, r, t) \mid (e, r, t) \in \mathcal{N}_t(e)\}, \tag{4.3}$$

$$\mathbf{v}_e = P(S_{\text{head}}(e) \cup S_{\text{tail}}(e)), \tag{4.4}$$

Here, $S_{\text{head}}(e)$ contains the representation vectors of neighborhood $\mathcal{N}_{\text{head}}(e)$, and $S_{\text{tail}}(e)$ contains those for $\mathcal{N}_{\text{tail}}(e)$. The difference between Eq. (4.1) and ours (Eqs. (4.2)–(4.4)) is the use of the pooling function in place of the summation. The candidates for functions T_{head}, T_{tail}, and P are described below.

Transition Function The aim of transition function T (including both T_{head} and T_{tail}) is to modify the vector of a neighbor node to reflect the relations between the current node and the neighbor. The examples of the transition function are listed here:

$$T(\mathbf{v}) = \mathbf{v}, \qquad \text{(identity)}$$

$$T(\mathbf{v}) = \tanh(\boldsymbol{A}\mathbf{v}), \qquad \text{(single } \tanh \text{ layer)}$$

$$T(\mathbf{v}) = \mathrm{ReLU}(\boldsymbol{A}\mathbf{v}), \qquad \text{(single } \mathrm{ReLU} \text{ layer)}$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ is a matrix of model parameters and $\tanh$ and ReLU are elementwise hyperbolic tangent and rectified linear unit functions. In addition, we can use other neural network techniques, such as batch-normalization [45], residual connection [36], and long short term memory [64].

We can also make the transition function dependent on the relation between the current node (entity) and the neighbor, such as in the following:

$$T_{\text{head}}(\mathbf{v}_h; h, r, e) = \tanh(\boldsymbol{A}^{\text{head}}_{(h,r,e)}\mathbf{v}_h),$$
$$T_{\text{tail}}(\mathbf{v}_t; e, r, t) = \tanh(\boldsymbol{A}^{\text{tail}}_{(e,r,t)}\mathbf{v}_t).$$

Note that the parameter matrices are now defined individually for each combination of node e, the current neighbors (h or t), and the relation r between them.

In the experiments of Section 4.7 and 4.6, we use the following transition functions:

$$T_{\text{head}}(\mathbf{v}_h; h, r, e) = \mathrm{ReLU}(\mathrm{BN}(\boldsymbol{A}^{\text{head}}_r\mathbf{v}_h)), \tag{4.5}$$

$$T_{\text{tail}}(\mathbf{v}_t; e, r, t) = \mathrm{ReLU}(\mathrm{BN}(\boldsymbol{A}^{\text{tail}}_r\mathbf{v}_t)), \tag{4.6}$$

where BN indicates batch normalization [45].

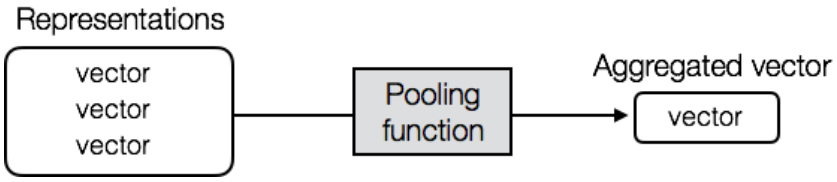


Figure 4.4: Scheme of pooling function in relational approach

Pooling Function Pooling function P is a function that maps a set of vectors to a vector, i.e., $P : 2^{\mathbb{R}^d} \to \mathbb{R}^d$. Its objective is to extract shared aspects from a set of vectors. For $S = \{\mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^N$, some simple pooling functions are as follows:

$$P(S) = \sum_{i=1}^{N} \mathbf{x}_i, \qquad \text{(sum pooling)}$$

$$P(S) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i, \qquad \text{(average pooling)}$$

$$P(S) = \max(\{\mathbf{x}_i\}_{i=1}^N), \qquad \text{(max pooling)}$$

where $\max$ is the elementwise max function. Sum pooling was used in [99, 64]; see also Eq. (4.1).

Stacking and Unrolling Graph Neural Networks As explained above, the propagation models decide how to propagate information from a node to its neighborhood. Applying this propagation model repeatedly, we can broadcast information of a node to farther nodes, i.e., each node can receive further information. Broadcasting can be implemented in one of two ways: stacking or unrolling.

The unrolled GNN is discussed in [99, 64]. In the unrolled GNN, the propagation model uses the same model parameters in every propagation. The propagation procedures are the same as described in Eqs. (4.2)–(4.4).

The stacked GNN is constructed in a similar manner to the well-known stacking technique [116]. In particular, the propagation process in the stacked GNN uses different model parameters depending on time step n. The transition function at each time step n, indicated by superscript n, is as follows.

$$\mathbf{v}_e^{(n)} = \begin{cases} \mathbf{v}_e, & \text{if } n = 0, \\ P(S_{\text{head}}^{(n-1)}(e) \cup S_{\text{tail}}^{(n-1)}(e)), & \text{otherwise,} \end{cases}$$

where

$$S^{(n)}_{\text{head}}(e) = \{T^{(n)}_{\text{head}}(\mathbf{v}^{(n)}_h; h, r, e) \mid (h, r, e) \in \mathcal{N}_{\text{head}}(e)\}$$
$$S^{(n)}_{\text{tail}}(e) = \{T^{(n)}_{\text{tail}}(\mathbf{v}^{(n)}_t; e, r, t) \mid (e, r, t) \in \mathcal{N}_{\text{tail}}(e)\}.$$

where $T^{(n)}_{\text{head}}$ and $T^{(n)}_{\text{tail}}$ are transition functions depending on head/tail and time.

4.4.3 Output Model: Score and Objective Functions

We use a TransE-based objective function as the output model. TransE [8] is one of the basic embedding-based models for KBC, and we use it for its simplicity and ease of training. Notice however that our architecture is not limited to TransE, and adopting other embedding-based models for the output model is equally straightforward.

Below, we explain the score function of TransE and its commonly used *pairwise-margin* objective functions. We then describe the modified objective function we used in our experiments, called the *absolute-margin* objective.

Score Function The (implausibility) score function f evaluates the implausibility of a triplet (h, r, t); smaller scores indicate that the triplet is more likely to hold. In TransE, the score function is defined by

$$f(h, r, t) = \|\mathbf{v}_h + \mathbf{v}_r - \mathbf{v}_t\| \tag{4.7}$$

where $\mathbf{v}_h$, $\mathbf{v}_r$, and $\mathbf{v}_t$ are the embedding vectors of the head, relation, and tail, respectively. This score function states that for a positive triplet (h, r, t), the sum of the head and relation vectors $\mathbf{v}_h + \mathbf{v}_r$ must be close to the tail vector $\mathbf{v}_t$, i.e., $\mathbf{v}_h + \mathbf{v}_r \sim \mathbf{v}_t$. This score function is modified and extended in [122, 65, 46, 124]. As mentioned earlier, all these models can be used as our output model.

Relative-Margin Objective Function The objective (loss) function defines the quantity to be minimized through optimization. The following *relative-margin* objective function is commonly used with KBC methods including TransE [8, 124]:

$$\mathcal{L} = \sum_{i=1}^{N} [\tau + f(h_i, r_i, t_i) - f(h'_i, r_i, t'_i)]_+ \tag{4.8}$$

Table 4.1: Specifications of the triplet classification datasets.

	WordNet11	Freebase13
Relations	11	13
Entities	38,696	75,043
Training triplets	112,581	316,232
Validation triplets	5,218	11,816
Test triplets	21,088	47,466

where $[x]_+$ is the hinge function $[x]_+ = \max(0, x)$ and scalar $\tau \in \mathbb{R}$ is a threshold (called *margin*), with (h_i, r_i, t_i) denoting a positive triplet and (h'_i, r_i, t'_i) denoting a negative triplet. This objective function requires score $f(h'_i, r_i, t'_i)$ to be greater than score $f(h_i, r_i, t_i)$ by at least τ. If the difference is smaller than τ, then the optimization changes the parameters to meet the requirement. In contrast, if the difference is greater than τ, the parameters are not updated.

The relative-margin objective thus pays attention to the difference in scores between positive-negative triplet pairs.

Absolute-Margin Objective Function Instead of the relative-margin objective, in this thesis, we employ the following objective function, which we call the *absolute-margin* objective.

$$\mathcal{L} = \sum_{i=1}^{N} f(h_i, r_i, t_i) + [\tau - f(h'_i, r_i, t'_i)]_+ \tag{4.9}$$

where τ is a hyperparameter, again called the *margin*. This objective function considers positive and negative triplets separately in the first and the second terms, not jointly as in the relative-margin objective. The scores for the positive triplets will be optimized towards zero, whereas the scores of the negative triplets are going to be at least τ. This objective function not only is easy to optimize, but also obtained good results in our preliminary experiments. We thus used this objective function for the experiments in Section 4.7 and 4.6.

Table 4.2: Result of the standard KBC experiment (without OOKB entities). The figures represent accuracy.

Method	WordNet11	Freebase13
NTN [103]	70.4	87.1
TransE [8]	75.9	81.5
TransH [122]	78.8	83.3
TransR [65]	85.9	82.5
TransD [46]	86.4	**89.1**
TransE-COMP [33]	80.3	87.6
TranSparse [47]	86.8	88.2
ManifoldE [123]	<u>87.5</u>	87.3
TransG [124]	87.4	87.3
lppTransD [129]	86.2	<u>88.6</u>
TransE-NMM [83]	86.8	<u>88.6</u>
NMM [83]	86.8	<u>88.6</u>
Proposed method	**87.8**	81.6

4.5 Implementation, Hyperparameters, and Datasets

4.5.1 Implementation and Optimization

We implemented our models using the neural network library Chainer (http://chainer.org/). The code and dataset are available at https://github.com/takuo-h/GNN-for-OOKB. All networks were trained by stochastic gradient descent with backpropagation; specifically, we used the Adam optimization method [53]. The step size of Adam was $\alpha_1/(\alpha_2 \cdot k + 1.0)$, where k indicates the number of epochs performed, $\alpha_1 = 0.01$, and $\alpha_2 = 0.0001$. The mini-batch size was $5,000$ and the number of training epochs was 300 in every experiment. Moreover, the dimension of the embedding space was 200 in the standard triplet classification and 100 in other settings.

4.5.2 Architecture

In the preliminary experiments, we tried several activation functions and pooling functions, and found the following hyperparameter settings on account of both computational time and performance. We used Eqs. (4.5)–(4.6) as transision functions in both the standard and OOKB settings. As the pooling function, we used the max pooling function in the standard triplet classification, and tried three pooling functions, max, sum, average, in the OOKB setting. The results of the preliminary experiments were

reflected in our selection of the absolute-margin objective function over the relative-margin objective function as well as the margin value $\tau = 300$ in the absolute-margin objective function (Eq. (4.9)). The absolute-margin objective function converged faster than the relative-margin objective function. Because the task is a binary classification of triplets into positive (i.e., the relations that must be present in the knowledge base) and negative triplets (those that must not), we determined the threshold value for output scores between these classes using the validation data.

4.5.3 Datasets

We used WordNet11 and Freebase13 [103] for evaluation. The data files were downloaded from http://cs.stanford.edu/people/danqi/. These datasets are subsets of two popular knowledge graphs, WordNet [76] and Freebase [7]. The specifications on these datasets are shown in Table 4.1. Half of the validation and test sets are negative triplets, and these are included in the numbers of validation triplets and test triplets. Detailed procedures to construct these datasets are explained in Section 4.8.

4.5.4 Construction for Corrupted Triplets

Both datasets contain training, validation, and test sets. The validation and test sets include positive and negative triplets. In contrast, the training set does not contain negative triplets. As usual with the case in which negative triplets are not available, *corrupted* triplets are generated from positive triplets are used as a substitute for negative triplets. From a positive triplet (h, r, t) in knowledge base $\mathcal{G}$, a corrupted triplet is generated by substituting a random entity sampled from $\mathcal{E}(\mathcal{G})$ for h or t. Specifically, to generate corrupted triplets, we used the "Bernoulli" trick, a technique also used in [122, 65, 46, 47].

4.5.5 Sampling Neighborhoods

To deal with the limited available computational resources (e.g., GPU memory), we sampled the neighbor entities randomly when an entity has too many of them. Indeed, there were some entities that appeared in a large number of the triplets, and thus had many neighbors; when the neighborhood size exceeded 64, we randomly chose 64 entities from the neighbors.

Table 4.3: Results of the OOKB experiment: accuracy of the simple baseline and proposed models.

		Head			Tail			Both		
Method	P	1,000	3,000	5,000	1,000	3,000	5,000	1,000	3,000	5,000
Baseline	sum	54.6	52.5	52.0	53.7	53.0	52.8	54.0	52.7	53.2
	max	58.1	56.3	56.4	55.2	54.2	55.3	56.8	56.8	56.4
	avg	63.0	60.2	61.1	63.8	63.9	63.0	65.3	63.9	64.8
Proposed	sum	70.2	62.6	59.6	64.6	56.5	55.0	59.5	55.2	54.2
	max	80.3	75.4	72.7	74.8	63.1	58.7	68.0	59.5	56.5
	avg	**87.3**	**84.3**	**83.3**	**84.0**	**75.2**	**69.2**	**83.0**	**73.3**	**68.2**

4.6 OOKB Entity Experiments without Texts

Datasets As we explained in Section 4.1, we tackle KBC tasks with OOKB entities and without texts. Although we can exploit various datasets for standard KBC, there is no dataset for this KBC setting. Therefore we processed the WordNet11 and WordNet11 datasets to construct several datasets for our OOKB entity experiment. In total, nine datasets were constructed with different numbers and positions of OOKB entities sampled from the test set. Although detailed procedures are described in Section 4.8, we explain aims of these datasets. Firstly, we address three different settings of numbers of OOKB entities, $1,000$, $3,000$, and $5,000$. to investigate how the numbers of OOKB entities effects model performance. Secondly, we change locations of OOKB entities in triplets. Error analysis in standard KBC implies that head entity and tail entity do not have same properties. For example, head entity and tail entity must be a parson and a nation in *nationality* relation. To investigate how these differences effects model performance, we address three different settings, $Head$, $Tail$, and $Both$, in which OOKB entities are selected in head entities, tail entities, and both of entities. Finally, we construct nine different datasets. The details of the generated OOKB datasets are shown in Table 4.8. We denote each of the nine datasets by $\{\text{Head}, \text{Tail}, \text{Both}\}$-$\{1{,}000, 3{,}000, 5{,}000\}$, respectively, where the first part represents the position of OOKB entities and the second part represents the number of triplets used for generating the OOKB entities

Baselines and Proposal We explain baselines and proposal in this experiment. We use TransE as baselines to train representation vectors in training time. Proposals are same in standard KBC tasks basically. As below, we explain how to use the baselines

Table 4.4: Detailed accuracies and statistics in the results of WordNet11.

Relation	Accuracy	Total Number	$\#\mathcal{N}_h$	$\#\mathcal{N}_t$
member_holonym	95.3	9146	1.98	1.02
member_meronym	94.8	9146	1.02	1.97
part_of	91.5	6600	1.14	2.13
subordinate_instance_of	91.3	3778	1.20	6.96
has_part	90.6	6139	2.29	1.11
domain_region	87.1	4227	12.11	1.03
synset_domain_topic	86.7	3976	1.03	11.7
has_instance	84.5	36178	3.56	1.16
type_of	84.1	30556	1.09	3.25
domain_topic	80.2	1099	8.14	1.05
similar_to	71.8	1659	1.42	1.38

Table 4.5: Detailed accuracies and statistics in the results of Freebase13.

Relation	Accuracy	Total Number	$\#\mathcal{N}_h$	$\#\mathcal{N}_t$
gender	84.0	59,423	1.00	29711.5
nationality	82.2	47,581	1.06	255.81
profession	81.6	45,931	1.38	100.95
religion	80.7	8,112	1.11	75.81
ethnicity	76.5	4,698	1.10	22.37
cause_of_death	75.3	10,816	1.15	63.62
institution	74.6	15,566	1.26	21.44

and proposals in training and evaluation times.

1. *Training Representation Vectors*

   As baselines, we train representation vectors using standard TransE, i.e., represent entities using corresponding representation vectors themselves. In contrast, we calculate representation vectors of entities using transision functions Eqs. (4.5)–(4.6) and pooling functions. Note that the calculation does not use corresponding representation vectors, although we can provide corresponding vectors for the entities. The differences between baselines and proposals come from whether corresponding vectors are used to represent entities.

2. *Calculating Representation Vectors of OOKB entities*

   In KBC with OOKB entities, we recieve auxiliary knowledge connecting known entities and OOKB entities in test time. We exploit the auxiliary knowledge to calculate representations of OOKB entities in both of baselines and proposals. The differences appear in how to calculate representations. Baselines apply

Table 4.6: Accuracy of stacked and unrolled GNNs.

Depth	Stacking	Unrolling
1	87.8	87.8
2	87.5	87.2
3	87.1	86.7
4	87.0	87.0

pooling functions to representation vectors of neighborhood directly. In contrast, proposals exploit pooling functions after applied transision functions Eqs. (4.5)–(4.6).

Let $\mathrm{Aux}(e)$ be auxiliary knowledge of an entity e containing triplets $\mathrm{Aux}_{\text{head}}(e) = \{(h', r', e)\}$ and $\mathrm{Aux}_{\text{tail}}(e) = \{(e, r', t')\}$, i.e., $\mathrm{Aux}(e) = \mathrm{Aux}_{\text{head}}(e) \cup \mathrm{Aux}_{\text{tail}}(e)$. If average function is used as pooling function, we can describe calculations of the baseline and the proposal as follows:

$$\begin{aligned}
\text{baseline}: \quad \mathbf{v}_e = \; & \frac{1}{n}\Big(\sum_{(h',r',e)\in \mathrm{Aux}_{\text{head}}(e)} \mathbf{v}_{h'} + \sum_{(e,r',t')\in \mathrm{Aux}_{\text{tail}}(e)} \mathbf{v}_{t'} \Big) \\
\text{proposal}: \quad \mathbf{v}_e = \; & \frac{1}{n}\Big(\sum_{(h',r',e)\in \mathrm{Aux}_{\text{head}}(e)} T_{\text{head}}(\mathbf{v}_{h'}; h', r', e) \\
& + \sum_{(e,r',t')\in \mathrm{Aux}_{\text{tail}}(e)} T_{\text{tail}}(\mathbf{v}_{t'}; e, r', t') \Big)
\end{aligned}$$

where n is a number of triplets in auxiliary knowledge $n = \#\mathrm{Aux}(e)$.

Result Using the nine datasets generated from WordNet11, we verified the effectiveness of our proposed model.

We used the following simple method as the baseline in this experiment. Given an OOKB entity u, we first obtained the embedding vectors of the neighborhood (determined by the triplets in the auxiliary knowledge) using TransE, and then converted these vectors into the representation vector of u using a pooling function: sum, max, or average. Note that because all the neighborhood entities of u are in the training knowledge base, their vectors can be computed using standard KBC methods. We followed the original paper [8] of TransE for the hyperparameter and other settings.

The results are shown in Table 4.3. Bold and underlined figures are respectively the best and second best scores for each dataset. The column labeled "pooling" indicates which pooling function was used. As the table shows, our model outperforms the

baselines considerably. In particular, GNN with average pooling outperforms the other methods on all datasets. GNN with max pooling also shows good accuracy, but in several settings, such as Tail-3000, it was outperformed by the baseline model with average pooling.

In additon to WordNet11, we also carried out experiments using Freebase13. The Freebase13 is generated from Freebase which is an other standard knowledge base in KBC tasks. The results are show in Table 4.7. Following the results in WordNet11 experiments, we used average function as pooling functions only in both baseline and proposal. As the table shows, our model outperforms the baselines in Head and Both settings. However, our molde does not show better results than basesline's results in Tail.

4.7 Standard Triplet Classification

Aims, Results, and Error Analysis In this section, we carried out standard knowledge base completion in which OOKB entities appear. The experiments aim to investigate the different results between WordNet11 and Freebase13 in OOKB entity experiments. The results are shown in Table 4.2. Except for the proposed method, they are obtained from the respective papers. Bold and underlined figures are the best and second best scores for each dataset, respectively. Our model showed state-of-the-art performance on the WordNet11 dataset. On the Freebase13 dataset, it did not perform as well as the state-of-the-art KBC methods, although it was slightly better than TransE on which our model was built on.

To analyse these results in detail, we show accuracies, total number of triplets containing the relations, and averaged numbers of head entities and tail entities the relations have. The Table 4.4 shows numbers of WordNet11 and the Table 4.5 shows numbers of Freebase13. In these tables, $\#\mathcal{N}_h$ and $\#\mathcal{N}_t$ indicates averaged numbers of entities e containd in triplets (h, r, e) and (e, r, t) respectively, for a given relation r. We can find out Wordnet11 contains symmetric relations such as *part-of* and it has-part basically. These relations allow us to propagate information transductively, and to achieve high accuracies. On the other hand, Freebase13 are quite different from WordNet11 from the viewpoint of knowledge graphs. For example, relations (e.g., *cause-of-death*) are hard to infer and relations (e.g., *gender*) have quite unbalanced appearances of head and tail entities in the Freebas13. This observation implies that these relations make propagations difficult and then we cannot improve predictions in

Table 4.7: Results of KBC with OOKB entitis using Freebase13.

		Head			Tail			Both		
Methods	Pooling	1,000	3,000	5,000	1,000	3,000	5,000	1,000	3,000	5,000
Baseline	avg	**79.1**	75.6	74.6	62.4	**71.2**	**71.5**	60.0	61.0	60.0
Proposal	avg	77.9	**76.8**	**75.9**	**69.0**	66.8	63.3	**70.8**	**67.7**	**65.1**

Freebase13.

Effects of Iterative Propagation Stacking and unrolling are important techniques because they enable us to broadcast node information to distant nodes. In this experiment, we illustrate the effect of stacking and unrolling in the standard triplet classification task. The dataset is WordNet11, used for standard triplet classification.

Table 4.6 shows the performance of the stacked and unrolled GNNs. The parameter "depth" indicates how many times the propagation model is iteratively applied. Note that when depth $= 1$, the two models reduce to the vanilla GNN, for which the result was 87.8%, as also shown in Table 4.2. These results imply that the stacking and unrolling techniques do not improve performance. We believe this is because of the power of the embedding models, i.e., we can embed information about distant nodes into a continuous space.

4.8 Creating Datasets

We processed the WordNet11 dataset to construct several datasets for our OOKB entity experiment.

In total, nine datasets were constructed with different numbers and positions of OOKB entities sampled from the test set. The process consists of two steps: choosing OOKB entities and filtering and splitting triplets.

1. *Choosing OOKB entities.* To choose the OOKB entities, we first selected $N = 1,000$, $3,000$, and $5,000$ triplets from the WordNet11 test file.

   For each of these three sets, we chose the initial candidates for the OOKB entities (denoted by $\mathcal{I}$) in three different ways (thereby yielding nine datasets in total); these settings are called Head, Tail, and Both. In the Head setting, all head entities in the N triplets are regarded as candidate OOKB entities. The Tail

setting is similar, but with the tail entities regarded as candidates. In the Both setting, all entities appearing as either a head or tail are the candidates.

The final OOKB entities are the entities $e \in \mathcal{I}$ that appear in a triplet (e, r, e') or (e', r, e) in the WordNet11 training set, with $e' \notin \mathcal{I}$. Note that the entities $h, t \notin \mathcal{I}$ are contained in the knowledge bases we already have and not in the OOKB entities. This last process filters out candidate OOKB entities that do not have any connection with the training entities.

2. *Filtering and splitting triplets.* Using the selected OOKB entities, the original training dataset was split into the training dataset and the auxiliary datasets for the OOKB entity problem. That is, triplets that did not contain the OOKB entities were placed in the OOKB training set, and triplets containing one OOKB entity and one non-OOKB entity were placed in the auxiliary set. Triplets that contained two OOKB entities were discarded.

   For the test triplets, we used the same first N triplets in the WordNet11 test file that we used in Step 1, with the exception that the triplets that did not contain any OOKB entities were removed. For the validation triplets, we simply removed the triplets containing OOKB entities from the WordNet11 validation set.

The details of the generated OOKB datasets are shown in Table 4.8. We denote each of the nine datasets by $\{\text{Head}, \text{Tail}, \text{Both}\}$-$\{1{,}000, 3{,}000, 5{,}000\}$, respectively, where the first part represents the position of OOKB entities and the second part represents the number of triplets used for generating the OOKB entities.

4.9 Conclusion

In this thesis, we proposed a new KBC task in which entities unobserved at training time are involved. For this task, we proposed a Graph-NN tailored to KBC with OOKB entities. We conducted two triplet classification tasks to verify the effectiveness of our proposed model. In the OOKB entity problem, our model outperformed the baselines considerably. Our model also showed state-of-the-art performance on WordNet11 in the standard KBC setting.

Table 4.8: Number of entities and triplets in the the OOKB datasets.

	Head		
	1,000	3,000	5,000
Training triplets	108,197	99,963	92,309
Validation triplets	4,613	4,184	3,845
OOKB entities	348	1,034	1,744
Test triplets	994	2,969	4,919
Auxiliary entities	2,474	6,791	10,784
Auxiliary triplets	4,352	12,376	19,625

	Tail		
	1,000	3,000	5,000
Training triplets	96,968	78,763	67,774
Validation triplets	3,999	3,122	2,601
OOKB entities	942	2,627	4,011
Test triplets	986	2,880	4,603
Auxiliary entities	8,191	16,193	20,345
Auxiliary triplets	15,277	31,770	40,584

	Both		
	1,000	3,000	5,000
Training triplets	93,364	71,097	57,601
Validation triplets	3,799	2,759	2,166
OOKB entities	1,238	3,319	4,963
Test triplets	960	2,708	4,196
Auxiliary entities	9,899	19,218	23,792
Auxiliary triplets	18,638	38,285	48,425

Chapter 5

Conclusion

In this thesis, we have proposed a general word representation based on the distributional hypothesis. This representation consists of two functions, association and embedding. To represent a word, the association function collects objects related to the word, and the embedding function converts the objects into continuous-valued vectors. Using these functions, our representation calculates word representations from associated objects on the fly. Our representation allows us to handle two difficult types of words: out-of-vocabulary words and polysemous words. Previous word representations face problems arisen from the words, i.e., the problem of out-of-vocabulary words whose embedding vectors are not trained and polysemous words whose senses are hard to capture using a fixed number of vectors. The underlying cause of these problems is representing words as embedding vectors directly. This direct representation requires pre-trained embedding vectors to represent words and loses rich information contained in polysemous words. To overcome these problems, our representation calculates word representations from associated objects dynamically. The representation depending on associated objects is a general solution to handle out-of-vocabulary words and polysemous words. For example, previous work addressing the problem of out-of-vocabulary words exploit characters and subwords of the words as features of word representations. We can view these work takes an approach deal with characters and subwords as associated objects. To represent polysemous words, the exemplar-based approach exploiting usages of a word has been proposed. In this approach, associated objects are collections of usages.

Using our representation, we addressed two information extraction tasks: (i) entity classification and (ii) knowledge base completion. In the former task, the objects associated with words are their contexts in large corpora, and in the latter, they are the

neighborhood relations and entities (words) in a knowledge graph. To tackle the entity classification, we exploited contexts in Wikipedia as associated objects, i.e., represent an entity using a vector set generated from the contexts. This work aims not only to solve entity classification, but also to evaluate how word representations based on our framework can model polysemy based on contextual approach. Our model outperformed baselines significantly. Thus, our framework can model polysemous words, even if the words are out-of-vocabulary. To tackle the knowledge base completion task, we exploited neighborhoods of entities in knowledge graphs as associated objects. This work aimed not only to complete knowledge bases but also to deal with OOKB entities without texts based on relational approach. The relational approach allows us to calculate low-frequency words, specifically, technical terms. The experimental results have shown the effectiveness of our proposed model in the OOKB setting. Additionally, in the standard KBC setting in which OOKB entities are not involved, our model achieved state-of-the-art performance on the WordNet dataset. Word representations are the foundation of natural language processing. At the same time, they can be the foundation of information extraction. This fact points out a future direction of our representation, which describes natural language processing and information extraction seamlessly. For example, our representation can be applied to question answering tasks. Assume that a given question is a text and a candidate of answers are represented as a collection of contexts. Our representation implies that a question can be regarded as an associated object of a correct answer because the question characterizes the answer sufficiently. Therefore, our representation allows us to rephrase question answering tasks as classification tasks: which vector sets a vector should be contained in, where the vector sets and the vector represent answers and a question respectively.

Bibliography

[1] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. In *CoRR*, Vol. abs/1412.6830, 2014.

[2] G. Angeli, J. Tibshirani, J. Wu, and C. D. Manning. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1556–1567, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] B. Athiwaratkun and A. Wilson. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1645–1656, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[4] L. J. Ba, R. Kiros, and G. E. Hinton. Layer normalization. In *CoRR*, Vol. abs/1607.06450, 2016.

[5] G. Bebis and M. Georgiopoulos. Feed-forward neural networks. 13(4):27–31, 1994.

[6] G. Boleda, S. Schulte im Walde, and T. Badia. Modelling polysemy in adjective classes by multi-label classification. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 171–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250, 2008.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, pp. 2787–2795, 2013.

[9] G. Bouma. Normalized (pointwise) mutual information in collocation extraction. pp. 31–40, 2009.

[10] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 576–583, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[11] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. abs/1709.03698, 2017.

[12] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. 2016.

[13] X. Chen, Z. Liu, and M. Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1025–1035, Doha, Qatar, October 2014. Association for Computational Linguistics.

[14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[15] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. 16(1):22–29, 1990.

[16] M. Craven and J. Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pp. 77–86. AAAI Press, 1999.

[17] F. Dai and Z. Cai. Glyph-aware embedding of chinese characters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pp. 64–69, 2017.

[18] F. De Saussure and W. Baskin. Course in general linguistics (1915). 1959.

[19] P. D. Deane. Polysemy and cognition. 75(4):325–361, 1988.

[20] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pp. 3844–3852, 2016.

[21] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In D. Precup and Y. W. Teh eds., *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[22] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 199–209. Association for Computational Linguistics, 2016.

[23] K. Erk and S. Pado. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pp. 92–97, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[24] W. Fang, J. Zhang, D. Wang, Z. Chen, and M. Li. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 260–269, 2016.

[25] X. Gastaldi. Shake-shake regularization. abs/1705.07485, 2017.

[26] J. Gehring, M. Auli, D. Grangier, and Y. Dauphin. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 123–135, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[27] A. Gliozzo, C. Giuliano, and C. Strapparava. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 403–410, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[28] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudk eds., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Vol. 15 of *Proceedings of Machine Learning Research*, pp. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[29] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, pp. 1319–1327, 2013.

[30] E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017.

[31] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 140–149, Berlin, Germany, August 2016. Association for Computational Linguistics.

[32] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

[33] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 318–327, 2015.

[34] X. Han and L. Sun. Global distant supervision for relation extraction, 2016.

[35] Z. Harris. Distributional structure. 10(23):146–162, 1954.

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[37] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett eds., *Advances in Neural Information Processing Systems 28*, pp. 1693–1701. Curran Associates, Inc., 2015.

[38] S. Hochreiter and J. Schmidhuber. Long short-term memory. 9(8):1735–1780, 1997.

[39] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld. 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*, pp. 541–550, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[40] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. 2(5):359–366, 1989.

[41] E. Huang, R. Socher, C. Manning, and A. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 873–882, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[42] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. abs/1608.06993, 2016.

[43] Y. Huang and W. Y. Wang. Deep residual learning for weakly-supervised relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1803–1807, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[44] S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *CoRR*, Vol. abs/1702.03275, 2017.

[45] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

[46] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. 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*, Vol. 1: Long Papers, pp. 687–696, 2015.

[47] G. Ji, K. Liu, S. He, and J. Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[48] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. Deep learning with s-shaped rectified linear activation units. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 1737–1743, 2016.

[49] R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 908–918, Berlin, Germany, August 2016. Association for Computational Linguistics.

[50] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. June 2014.

[51] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 2016.

[52] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2741–2749, 2016.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. abs/1412.6980, 2014.

[54] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett eds., *Advances in Neural Information Processing Systems 30*, pp. 972–981. Curran Associates, Inc., 2017.

[55] S. Kobayashi, R. Tian, N. Okazaki, and K. Inui. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 850–855, San Diego, California, June 2016. Association for Computational Linguistics.

[56] M. Kågebäck and H. Salomonsson. Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pp. 51–56, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[57] J. Krishnamurthy and T. Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 754–765, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger eds., *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

[59] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In M. F. Balcan and K. Q. Weinberger eds., *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48 of *Proceedings of Machine Learning Research*, pp. 1378–1387, New York, New York, USA, 20–22 Jun 2016. PMLR.

[60] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 1998.

[61] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pp. 24–26. ACM, 1986.

[62] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q.

Weinberger eds., *Advances in Neural Information Processing Systems 27*, pp. 2177–2185. Curran Associates, Inc., 2014.

[63] J. Li and D. Jurafsky. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1722–1732, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[64] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. abs/1511.05493, 2015.

[65] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[66] W. Ling, I. Trancoso, C. Dyer, and A. W. Black. Character-based neural machine translation. abs/1511.04586, 2015.

[67] X. Ling and D. Weld. Fine-grained entity recognition, 2012.

[68] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun. Topical word embeddings, 2015.

[69] Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. abs/1710.10121, 2017.

[70] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[71] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421. Association for Computational Linguistics, 2015.

[72] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, Vol. 30, 2013.

[73] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[74] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. 2013.

[75] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger eds., *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

[76] G. A. Miller. WordNet: A lexical database for English. 38(11):39–41, 1995.

[77] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th InternationalJoint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[78] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger eds., *Advances in Neural Information Processing Systems 27*, pp. 2924–2932. Curran Associates, Inc., 2014.

[79] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. AISTATS ' 05. Citeseer, 2005.

[80] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[81] A. Neelakantan and M.-W. Chang. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 515–525, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[82] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1059–1069, Doha, Qatar, October 2014. Association for Computational Linguistics.

[83] D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. Neighborhood mixture model for knowledge base completion. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 40–50, 2016.

[84] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. 104:11–33, 2016.

[85] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. abs/1711.10433, 2017.

[86] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[87] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In S. Dasgupta and D. McAllester eds., *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[88] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[89] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[90] M. Pershina, B. Min, W. Xu, and R. Grishman. Infusion of labeled data into distant supervision for relation extraction. In *Proceedings of the 52nd Annual*

Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 732–738, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[91] J. Peyre, I. Laptev, C. Schmid, and J. Sivic. Weakly-supervised learning of visual relations. In *ICCV 2017-International Conference on Computer Vision 2017*, 2017.

[92] A. Raganato, C. Delli Bovi, and R. Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1156–1167, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[93] S. Reddy, I. Klapaftis, D. McCarthy, and S. Manandhar. Dynamic and static prototype vectors for semantic composition. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pp. 705–713, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[94] S. Reddy, D. McCarthy, S. Manandhar, and S. Gella. Exemplar-based word-space model for compositionality detection: Shared task system description. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pp. 54–60, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[95] J. Reisinger and R. J. Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 109–117, Los Angeles, California, June 2010. Association for Computational Linguistics.

[96] M. Ren, R. Liao, R. Urtasun, F. H. Sinz, and R. S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *International Conference on Learning Representations*, 2016.

[97] R. Roller, E. Agirre, A. Soroa, and M. Stevenson. Improving distant supervision using inference learning. 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)*, pp. 273–278, Beijing, China, July 2015. Association for Computational Linguistics.

[98] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett eds., *Advances in Neural Information Processing Systems 29*, pp. 901–909. Curran Associates, Inc., 2016.

[99] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. 20:61–80, 2009.

[100] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. 45(11):2673–2681, 1997.

[101] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang. Disan: Directional self-attention network for RNN/CNN-free language understanding. abs/1709.04696, 2017.

[102] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. abs/1409.1556, 2014.

[103] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26*, pp. 926–934, 2013.

[104] S. Sonoda and N. Murata. Neural network with unbounded activation functions is universal approximator. In *Applied and Computational Harmonic Analysis*. Elsevier, 2015.

[105] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. abs/1505.00387, 2015.

[106] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. 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*, pp. 455–465, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[107] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger eds., *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

[108] S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 721–729, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[109] S. Targ, D. Almeida, and K. Lyman. Resnet in resnet: Generalizing residual architectures. 2016.

[110] F. Tian, H. Dai, J. Bian, B. Gao, R. Zhang, E. Chen, and T.-Y. Liu. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 151–160, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[111] A. Trischler, Z. Ye, X. Yuan, P. Bachman, A. Sordoni, and K. Suleman. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 128–137, Austin, Texas, November 2016. Association for Computational Linguistics.

[112] B. Vandekerckhove, D. Sandra, and W. Daelemans. A robust and extensible exemplar-based model of thematic fit. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 826–834, Athens, Greece, March 2009. Association for Computational Linguistics.

[113] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett eds., *Advances in Neural Information Processing Systems 30*, pp. 6000–6010. Curran Associates, Inc., 2017.

[114] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised semantic segmentation with a multi-image model. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 643–650. IEEE, 2011.

[115] L. Vilnis and A. McCallum. Word representations via gaussian embedding. In *International Conference on Learning Representations*, 2015.

[116] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. 11:3371–3408, 2010.

[117] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett eds., *Advances in Neural Information Processing Systems 28*, pp. 2692–2700. Curran Associates, Inc., 2015.

[118] O. Vinyals, L. u. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett eds., *Advances in Neural Information Processing Systems 28*, pp. 2773–2781. Curran Associates, Inc., 2015.

[119] T. Vu and D. S. Parker. k-embeddings: Learning conceptual embeddings for words using context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1262–1267, San Diego, California, June 2016. Association for Computational Linguistics.

[120] J. Wang, M. Bansal, K. Gimpel, B. Ziebart, and C. Yu. A sense-topic model for word sense induction with unsupervised data enrichment. 3:59–71, 2015.

[121] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1591–1601, 2014.

[122] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[123] H. Xiao, M. Huang, and X. Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 1315–1321, 2016.

[124] H. Xiao, M. Huang, and X. Zhu. TransG: A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1: Long Papers, pp. 2316–2325, 2016.

[125] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions, 2016.

[126] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

[127] Y. Yaghoobzadeh and H. Schütze. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 715–725, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[128] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen. Breaking the softmax bottleneck: a high-rank RNN language model. *arXiv preprint arXiv:1711.03953*, abs/1706.00286, 2017.

[129] H.-G. Yoon, H.-J. Song, S.-B. Park, and S.-Y. Park. A translation-based knowledge graph embedding preserving logical property of relations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 907–916, 2016.

[130] D. Yuan, J. Richardson, R. Doherty, C. Evans, and E. Altendorf. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1374–1385, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

List of Publications

Journal Papers

1. 濱口拓男, 大岩 秀和, 新保仁, 松本裕治, 未知エンティティを伴う知識ベース補完: グラフニューラルネットワークを用いたアプローチ, 人工知能学会論文誌 2018 Vol.33, No.2, 2018 to appear.

International Conference

1. Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, Yuji Matsumoto, Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach, Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, page 1802–1808, 2017

List of Other Publications

1. 濱口拓男, 新保仁, 松本裕治, “条件付きロジスティック分布を用いた重み付き多タスク学習” 情報処理学会研究報告 第 216 回自然言語処理研究会 第 101 回音声言語情報処理研究会 合同研究会, 東京, 2014 年 5 月.
2. 濱口拓男, 新保　仁, 松本裕治, “文脈自由文法に基づいた復元的ニューラルネット”, 第 17 回情報論的学習理論ワークショップ (IBIS2014), 愛知, Nov. 2014.
3. 濱口拓男, 大岩 秀和, 新保仁, 松本裕治, “Graph Neural Network を用いた未知 Entity の表現獲得について” 情報処理学会研究報告 第 231 回自然言語処理研究会・第 116 回音声言語情報処理研究会 合同研究会, 大阪, 2017 年 5 月.
4. 濱口拓男, 大岩 秀和, 新保　仁, 松本裕治, “周辺文脈の集合による単語表現の獲得と関係抽出への応用”, 第 12 回 NLP 若手の会 (YANS) 第 12 回シンポジウム (YANS2017), 沖縄, 2017 年 9 月

Awards

1. 奨励賞, 第 12 回 NLP 若手の会 (YANS) 第 12 回シンポジウム (YANS2017), 沖縄, 2017 年 9 月

Acknowledgments

主指導教員の松本裕治教授には，研究指導だけでなく研究生活や進路に関して様々な助言・指導をしていただきました．また何より，研究室に受け入れていただいたおかげで計量的に自然言語を扱う面白さや情報抽出の意義を知ることができました．そうして機会を与えて頂けたからこそ，こうして博士論文を書けるまで至るようにまでなれたと思います．深く感謝いたします．中村哲教授には，お忙しい中審査委員を引き受けて頂き，公聴会・最終審査にて研究に関する助言をいただきました．深く感謝致します．新保仁准教授には，日頃から多くのご助言をいただきました．特に論文の書き方や研究の姿勢に関して，入学してから今まで様々な形でご指導していただきました．右も左も分からないような状態から丁寧に接していただいき，深く感謝致します．進藤裕之助教授，能地宏助教授には日々の細かな研究の相談に乗っていただいきました．特に幾つかの着想や研究は，お二方との研究に関する話の中で得られたものがあります．深く感謝しております．松本研の北川祐子秘書には，研究生活において多くの支援をしていただきました．その支えがなければ難しかったことも多々あり，深く感謝しております．リクルートでのインターン時のメンターの大岩秀和氏には，本論文を書くための重要な示唆を多く与えていただきました．また研究の要点の掴み方を教えていただくなど，恐らく大岩氏がいなければここまで辿り着くことはなかったと思います．あの時声をかけていただいたいた事，とても感謝しております．ありがとうございます．学部時代にお世話になった金子孝吉教授には，自然言語の世界を知る機会を与えていただきました．あの時，授業で教えていただいた知識が今に至るきっかけになったと言って過言ではないと思います．深く感謝致します．松本研に所属する，また所属していた学生の皆さんには，定例の研究会での有益なコメントを多くいただきました．まだまとまりきってない段階の研究に関する相談を聞いていただくなど，様々な形での議論が今に至る研究には欠かせないものになっていました．ありがとうございます．最後に，日々の生活に限らずよく迷走してしまう私ですが，やりたいことをやりなさいと支えてくれた父と母に感謝いたします．