A Study on Encoding Natural Language into Distributed Representations

自然言語の分散表現へのエンコードに関する研究

Sho Takase

Graduate School of Information Sciences

Tohoku University

A thesis submitted for the degree of

Doctor of Information Science

January 2017

Acknowledgements

本研究を進めるにあたって，多くの方にご協力をいただきました．ここに，心より感謝の意を表します．

主指導教官である乾健太郎教授には，学部 3 年次での研究室配属から，6 年間の研究活動において，多大なご助言をいただきました．深く感謝いたします．

ご多忙の中，審査委員として本論文を査読してくださいました田中和之教授，ならびに岡谷貴之教授に深く感謝いたします．

岡崎直観准教授には初めての論文執筆から，最初の国際会議への投稿，日本学術振興会特別研究員の申請書の執筆や，博士課程 3 年次での ACL への投稿まで，細部まで面倒をみていただきました．議論の組み立て方や説明の方法を丁寧に指導していただき，心より感謝しています．

NTT コミュニケーション科学基礎研究所の鈴木潤さんにはニューラルネットワークを用いた要約文生成の研究で，インターンシップを受け入れていただきました．短い期間ではありましたが，濃密な経験をさせていただき，トップ会議に採択されるまでに仕上げることができました．感謝しています．次年度より，研究者としてお世話になりますので，よろしくお願いいたします．

乾・岡崎研究室で初めて博士を取った方として，井之上直也助教には研究のみならず，書類や申請書の作成など，手続き面でも大変お世話になりました．本審査当日締め切りの書類を急遽前日に作成することになった際も，遠方，かつ，急な連絡だったにも関わらず，ご助言をいただき，無事に書き終えることができました．ありがとうございました．

最後に，乾・岡崎研究室の皆様からは様々なご助言をいただき，相談にのっていただくとともに，研究生活を暖かく支えていただきました．心より感謝を申し上げます．本当にありがとうございました．

まだしばらくは自然言語処理の研究者としてやっていこうと思いますので，今後ともよろしくお願いいたします．

Abstract

Modeling the meaning of a text is a crucial technique for NLP applications such as relation extraction, textual entailment recognition, paraphrase detection, and so on. Thus, the goal of this thesis is to compute the meanings of words and phrases in order to comprehend the meaning of a text.

To compute the meanings of words, the most basic way is constructing a co-occurrence matrix between words. After the construction, we can regard each row of the matrix as semantic vectors of words and compute the similarity of meanings of words by computing the similarity of semantic vectors based on the distributional hypothesis [Harris, 1954].

In addition to the above distributional representation, recent researches proposed various ways to learn distributed representations, which are dense real-valued vectors encoding syntactic and semantic information, and demonstrated distributed representations are useful to capture the meanings of words.

To expand those meaning representations of words into arbitrary-length phrases, the most naive approach is to regard a phrase as a single unit (word) and to acquire distributional/distributed representations of words as usual. However, this approach might suffer from the datasparseness problem.

To adress the datasparseness problem, we should compute the meaning of a phrase from constituent words. In this thesis, we propose two neural network based methods: modified Recursive Neural Network and Gated Additive Composition. Modified Recursive Neural Network can handle a transition of a polarity by each word such as increase and decrease. Gated Additive Composition can take account of an importance of a word. For example, content words such as verbs are more important than function

words such as prepositions in computing the meaning of a phrase. The latter model is more general than the former model because the latter model can also handle an important dimension for representing polarities.

We have few datasets to evaluate computing the meaning of phrases. Thus, in this thesis, we construct a new evaluation dataset that contains phrase pairs (specifically, relational pattern pairs) with similarity rating annotations. Using the constructed dataset, we show that the proposed method (specifically, Gated Additive Composition) is superior to othe neural network based methods.

In addition to the above evaluation, we evaluate several methods on various datasets to explore the efficient way to compose distributed representations of arbitrary-length phrases from distributed representations of constituent words. Moreover, we indicate that the efficiency of distributed representations of phrases by applying the distributed representations to an application task.

We also propose the method that encodes syntactic and semantic information such as part-of-speech (POS) tags. Through experiments, we show that the proposed method outperforms the method that uses only distributed representations of words. In other words, this thesis also indicates an effect of distributed representations of syntactic and semantic information whom traditional NLP parses.

Contents

List of Figures

Chapter 1

Introduction

Modeling the meaning of a text is one of the main challenges in Natural Language Processing (NLP). Take two sentences "*Tobacco causes lung cancer*" and "*Smoking increases the risk of cancer*" as an example. We can understand that these sentences have almost the same meaning because the meanings of subjects (*Tobacco* and *Smoking*), predicates (*causes* and *increases the risk of*), and objects (*lung cancer* and *cancer*) are similar respectively. The goal of this thesis is to compute the meanings of words and phrases in order to comprehend the meaning of a text as in this example.

In the beginning, we focus on modeling the meanings of words. The most basic way to acquire the meanings of words is constructing a co-occurrence matrix between words. Concretely, we count the co-occurrence frequency between a word and words appearing in its context window (context words). Consider the sentence "*The novelist writes the book.*". For the word *book*, we increment one to the co-occurrence frequency of the pairs *book–writes* and *book–the* if the size of context window is two. After the construction, we can regard each row of the matrix as semantic vectors of words and compute the similarity of meanings of words by computing the similarity of semantic vectors based on the distributional hypothesis [Harris, 1954]. For example, from the co-occurrence matrix in Figure 1.1, we can recognize that the meanings of book and novel are similar to each other because semantic vectors of these words are similar. The semantic vectors obtained from the co-occurrence matrix are also called distributional representations.

Moreover, recent researches proposed various ways to learn distributed representations, which are dense real-valued vectors encoding syntactic and semantic informa-

word \ context word	write	publish	read	walk	run	…
novel	132	97	154	0	0	…
book	178	86	194	0	0	…
dog	0	0	0	102	169	
…	…	….	…			…

Figure 1.1: The co-occurrence matrix of words.

tion, and demonstrated distributed representations are useful to capture the meanings of words [Baroni et al., 2014; Mikolov et al., 2013b; Pennington et al., 2014]. This thesis also employs the distributed representation as the meaning representation.

As a next step, we focus on the way to model the meanings of arbitrary-length phrases. The most naive approach is to regard a phrase as a single unit (word) and to acquire distributional/distributed representations of words as usual. In fact, several studies implemented this approach by mining phrasal expressions with strong collocations from a corpus as a preprocessing step [Mikolov et al., 2013b; Nakashole et al., 2012]. However, this approach might suffer from the datasparseness problem. Concretely, this naive approach has two big problems.

1. The quality of a semantic vector of a phrase may vary because the occurrence frequency of a phrase varies drastically.

2. We cannot compute semantic vectors of out-of-vocabulary phrases.

For example, Figure 1.2 shows the frequency and rank of phrases representing semantic relations between entities (relational patterns) in ukWaC corpus [Baroni et al., 2009]. The graph confirms that the distribution of occurrences of relational patterns follows Zipf's law. We expect that the pattern "*X* cause an increase in *Y*" cannot obtain

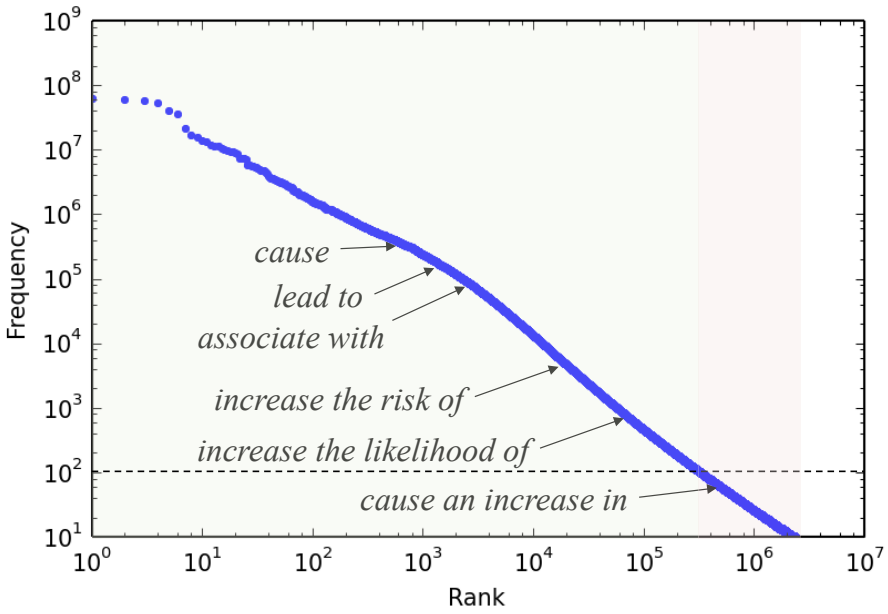


Figure 1.2: The frequency of relational patterns in ukWaC corpus.

sufficiently many co-occurrence statistics because the pattern appears less than 100 times (the first problem). Moreover, as mentioned in the second problem, we have no way to compute the semantic vectors of unseen patterns.

To address these problems, the straightforward way is to compute the meaning of a phrase from constituent words. Thus, the main issue of this thesis is exploring the efficient way to compose distributed representations of arbitrary-length phrases from distributed representations of constituent words.

In addition to compute the distributed representations of phrases, to improve the performance of NLP applications, we can consider to use additional syntactic and semantic information such as part-of-speech (POS) tags. In other words, we expect some enhancement by encoding such syntactic and semantic information into the distributed representation. This thesis also address this issue.

1.1 Research Issues

In this thesis, we address following three research issues:

- **What is the most suitable way to encode an arbitrary-length phrase?** Sev-

eral researches proposed neural network based methods (encoders) to compute distributed representations of phrases from constituent words [Cho et al., 2014; Socher et al., 2011b; Yu and Dredze, 2015] but they paid little attention to compare those encoders. Moreover, we have few datasets to investigate the performance of encoders on arbitrary-length phrases.

- **Are distributed representations of phrases efficient in NLP applications?** We believe that distributed representations of phrases are efficient in downstream tasks because computing the meanings of phrases is probably useful to NLP applications. However, the enhancement of the distributed representations is unclear.

- **Do additional distributed representations enhance the performance of NLP applications?** In addition to the effect of distributed representations of phrases, we are interested in the usefulness of distributed representations of syntactic and semantic information.

1.2 Contributions

This thesis makes following contributions:

- **Dataset Construction**: To evaluate the performance of encoders, we construct a new dataset that contains a pair of phrases (specifically, relational patterns) with five similarity ratings judged by human annotators. The new dataset shows a high inter-annotator agreement, following the annotation guideline of Mitchell and Lapata [2010]. The dataset is publicly available on the Web site[1].

- **Novel Encoders**: To compose distributed representations of arbitrary-length phrases, we propose two novel neural encoders. One is modified Recursive Neural Network to model the verbs that change or inherit the meaning. In other words, the way can identify that each verb is meaningful or meaningless. The other is more general way: the neural encoder can handle the importance of each

[1] `http://github.com/takase/relPatSim`

word in a phrase. For the latter method, the code is publicly available at the Github[1].

- **Exploring the most suitable way to compute distributed representations of phrases**: As mentioned in the previous section, previous studies paid little attention to comparison of phrase composition methods. To reveal pros and cons of each encoder, we compare existing neural encoders and the proposed method on various datasets.

- **Investigating the efficiency of distributed representations of phrases in NLP applications**: To explore the usefulness of distributed representations computed by a neural encoder, we apply the distributed representations as the feature vector on the relation classification task.

- **Exploring the enhancement of additional distributed representations**: We propose the method that is a combination of a sentence encoder and the encoder for syntactic/semantic information, and evaluate the performance on the headline generation task.

1.3 Thesis Overview

The rest of this thesis is structured as follows.

- **Chapter 2: Distributed Representations of Words**. In this chapter, we introduce the way to learn distributed representations of words from an unlabeled corpus. In addition, this chapter explains the relation between distributed representations and distributional representations.

- **Chapter 3: Composing Distributed Representations of Phrases with Modified Recursive Neural Network**. In the beginning of this chapter, we describe Recursive Neural Networks for phrase composition [Socher et al., 2011b]. Then, we modify Recursive Neural Networks to handle the verbs that change or inherit the meaning. We evaluate the performance of the proposed method on the inference relation dataset [Zeichner et al., 2012]. The evaluation brings two

[1] `https://github.com/takase/GAC4relpat`

necessities: data construction and more general way to compose distributed representations.

- **Chapter 4: Data Construction for Modeling Semantic Similarity**. To evaluate the performance of each encoder, we construct the dataset that contains a phrase pair with semantic similarity judgements by crowdsourcing workers.
- **Chapter 5: Composing Distributed Representations of Phrases with Gated Additive Composition**. We propose a more general encoder than modified Recursive Neural Network: Gated Additive Composition. This chapter shows the comparison among neural encoders including the proposed method. This chapter also indicate that the distributed representations of phrases enhance the performance on the relation classification task, and the dataset constructed in the previous chapter is useful.
- **Chapter 6: Encoding Semantic and Syntactic Features**. This chapter presents the method that is a combination of a sentence encoder and the encoder for syntactic/semantic information. We evaluate the enhancement by the distributed representations of syntactic/semantic information on the headline generation task.
- **Chapter 7: Conclusions**. We summarize our discussion, and present our future direction.

Chapter 2

Distributed Representations of Words

We explain the way to learn distributed representations of words before discussion about distributed representations of phrases. Concretely, in this chapter, we introduce representative methods to obtain distributed representations of words from unlabeled corpus: skip-gram with negative sampling, continuous bag of words, and GloVe. In addition, this chapter mentions the relation between distributed representations and distributional representations.

2.1 Methods to Obtain Distributed Representations

2.1.1 Skip-gram with Negative Sampling

Mikolov et al. [2013a] introduced the Skip-gram model that trains distributed representations of words to predict surrounding words. Let $\mathcal{D}$ denote a corpus consisting of a sequence of words $w_1, w_2, ..., w_T$, and V the set of words occurring in the corpus. The Skip-gram model minimizes the following objective function,

$$J = -\sum_{w \in \mathcal{D}} \sum_{c \in C_w} \log p(c|w). \tag{2.1}$$

Here, C_w is the set of context words for word w. $C_w = \{w_{-h}, ..., w_{-1}, w_{+1}, ..., w_{+h}\}$ (h is a parameter that adjusts the width of contexts), where w_{-p} and w_{+p} represent the word appearing p words before and after, respectively, the centered word w. The conditional probability $p(c|w)$ for predicting context word c from word w, is formalized

by a log-bilinear model,

$$p(c|w) = \frac{\exp\left(\boldsymbol{v}_w \cdot \tilde{\boldsymbol{v}}_c\right)}{\sum_{c' \in V} \exp\left(\boldsymbol{v}_w \cdot \tilde{\boldsymbol{v}}_{c'}\right)}. \tag{2.2}$$

Here, $\boldsymbol{v}_w \in \mathbb{R}^d$ is the vector for word w, and $\tilde{\boldsymbol{v}}_c \in \mathbb{R}^d$ is the vector for context c. Training the log-bilinear model yields two kinds of vectors $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}$, but we use only $\boldsymbol{v}$ as semantic vectors of words (word vectors). Because computing the denominator in Equation 2.2, the sum of the dot products for all the words in the corpus, is intractable, Mikolov et al. [2013b] proposed the negative sampling method based on noise contrastive estimation [Gutmann and Hyvärinen, 2012]. The negative sampling method trains logistic regression models to be able to discriminate an observed context word c from k noise samples (pseudo-negative words z).

$$\log p(c|w) \approx \log \sigma(\boldsymbol{v}_w \cdot \tilde{\boldsymbol{v}}_c) + k\, \mathbb{E}_{z \sim P_n}\left[\log \sigma(-\boldsymbol{v}_w \cdot \tilde{\boldsymbol{v}}_z)\right] \tag{2.3}$$

Here, P_n is the probability distribution for sampling noise words. Usually, we used the probability distribution of unigrams raised to the 3/4 power [Mikolov et al., 2013b].

2.1.2 Continuous Bag of Words

Mikolov et al. [2013a] also introduced the Continuous bag of words model (CBOW). On the contrary to the Skip-gram model, the CBOW trains distributed representations of words to predict the center word from surrounding words. In the same manner as the explanation of the Skip-gram model, we use $\mathcal{D}$ as a corpus containing a sequence of words $w_1, w_2, ..., w_T$, and V as the set of words occurring in the corpus. The CBOW minimizes the following objective function:

$$J = -\sum_{w \in \mathcal{D}} \log p(w|C_w). \tag{2.4}$$

Here, C_w is the set of context words, and the conditional probability $p(w|C_w)$ is the same as Equation 2.2 on condition that we define the vector of C_w as follows:

$$\boldsymbol{v}_{C_w} = \frac{1}{|C_w|} \sum_{c \in C_w} \boldsymbol{v}_c. \tag{2.5}$$

Here, $|C_w|$ is the number of words in C_w. In brief, the CBOW makes the vector of the center word to similar to the mean of the context word vectors.

2.1.3 GloVe

Pennington et al. [2014] claimed that the Skip-gram model and CBOW poorly take advantage of the statistical information from corpus such as co-occurrence matrix because they focus on only local co-occurrence in training. To enhance the quality of distributed representations, they proposed the GloVe that trains distributed representations of words to model co-occurrence matrix. Concretely, Glove minimizes the following equation:

$$J = \sum_{i,j}^{V} f(X_{ij})(v_i \cdot \tilde{v}_j + b_i + \tilde{b}_j - \log X_{ij})^2. \tag{2.6}$$

Here, V is the set of words (vocabulary), i and j are the ith word and jth word in vocabulary respectively, v_i is the distributed representation of i, $\tilde{v}_i$ is the distributed representation when i is the context word, and X_{ij} is co-occurrence frequency between i and j. Briefly, Equation 2.6 is a least squared loss between of co-occurrence frequencies and dot products of distributed representations. In addition, $f(X_{ij})$ returns a value from 0 to 1 depending of X_{ij}. This function makes low co-occurrence frequency to be light weight in objective function.

2.2 Relation between Distributional Representations and Distributed Representations

Previous studies proposed several methods to learn distributed representations from corpus. In addition, we can obtain distributional representations by constructing a co-occurrence matrix. These cause a simple question; which method is the best to obtain high quality meaning representations?

Baroni et al. [2014] demonstrated that distributed representations learned by the CBOW are superior than vector representations acquired by factorization of a co-occurrence matrix. On the other hand, Pennington et al. [2014] indicated that GloVe,

that models a co-occurrence matrix, achieved the significant improvement over the Skip-gram model and CBOW.

However, Levy and Goldberg [2014a] proved that the Skip-gram model is interpreted as factorizing co-occurrence matrix implicitly. In addition, Suzuki and Nagata [2015] showed that the Skip-gram with negative sampling and GloVe can be represented by a unified form. These researches suggest that the Skip-gram model, GloVe, and distributional representations obtained from a co-occurrence matrix are essentially identical. In fact, Levy et al. [2015] demonstrated that the Skip-gram model, GloVe, CBOW, and vector representations acquired by factorizing a co-occurrence matrix achieved similar performance on the various tasks.

In this thesis, we utilize the Skip-gram model with negative sampling to train distributed representations because the method does not require a co-occurrence matrix and it is easy to implement the method.

Chapter 3

Composing Distributed Representations of Phrases with Modified Recursive Neural Network

In this chapter, we pick a *relational pattern* that is a linguistic pattern connecting entities as a target phrase. Identifying the meaning of a relational pattern is essential in relation extraction that is the task of extracting semantic relations between entities from corpora. Based on the distributional hypothesis [Harris, 1954], most previous studies construct a co-occurrence matrix between relational patterns (e.g., "*X* cause *Y*") and entity pairs (e.g., "*X*: smoking, *Y*: cancer"), and then they recognize relational patterns sharing the same meaning regarding the co-occurrence distribution as a semantic vector [Min et al., 2012; Mohamed et al., 2011; Nakashole et al., 2012]. For example, we can find that the patterns "*X* cause *Y*" and "*X* increase the risk of *Y*" have the similar meaning because the patterns share many entity pairs (e.g., "*X*: smoking, *Y*: cancer"). Using semantic vectors, we can map a relational pattern such as "*X* cause *Y*" into a predefined semantic relation such as CAUSALITY only if we can compute the similarity between the semantic vector of the relational pattern and the prototype vector for the relation. In addition, we can discover relation types by clustering relational patterns based on semantic vectors.

However, this approach suffers from the data sparseness problem due to regarding a pattern as a 'word'. A natural approach to these problems is to compute the meaning of

a relational pattern based on semantic compositionality, e.g., computing the vector for "*X* increase the risk of *Y*" from the constituent words (e.g, 'increase' and 'risk'). This treatment can be expected to improve the quality of semantic vectors, incorporating information of the constituent words into the semantic vectors of relational patterns. For example, we can infer that the relational pattern "*X* increase the risk of *Y*" has a meaning similar to that of "*X* increase the danger of *Y*" only if we know that the word 'risk' is similar to 'danger'.

Recently, there has been much progress in the methods for learning distributed representations of words [Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013b]. Among these methods, the Skip-gram model [Mikolov et al., 2013b] received a fair amount of attention from the NLP community, because the model exhibits the additive compositionality exemplified by the famous example, $\boldsymbol{v}_{\mathrm{king}} - \boldsymbol{v}_{\mathrm{man}} + \boldsymbol{v}_{\mathrm{woman}} \approx \boldsymbol{v}_{\mathrm{queen}}$. Although we found a number of positive reports regarding additive composition, a linear combination of vectors is inadequate in some cases. For example, "*X* prevent the growth of *Y*" is dissimilar to "*X* grow *Y*" because 'prevent' negates the meaning of 'grow', but additive composition cannot handle the transformation. On the other hand, since "*X* have access to *Y*" has almost the same meaning as "*X* access *Y*", we should not add the meaning of 'have' to that of 'access'. For handling the verbs changing or inheriting the meaning, it is appropriate to apply a matrix because a matrix can transform (or inherit) a vector. In fact, Socher et al. [2012] proposed the recursive neural network (Recursive NN) method that can handle a word changing the meaning by using matrices, but the method requires a certain amount of labeled data.

In this chapter, we propose a novel method for modeling the meanings of relational patterns based on compositionality. More specifically, in addition to additive composition, we model the verbs that change or inherit the meaning by using Recursive NN. We extend the Skip-gram model so that it can learn parameters for Recursive NNs and distributed representations of words from unlabeled data. In addition, we introduce l_1-regularization for training parameters of Recursive NN to obtain a simpler model for semantic composition.

We conduct four kinds of experiments on the existing datasets, pattern similarity, relation extraction, and word similarity. The experimental results show that the proposed method can successfully model semantic compositions of relational patterns, outperforming strong baselines such as additive composition. The experiments also

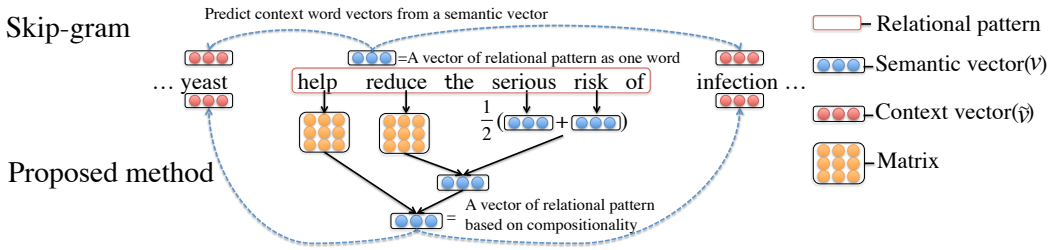


Figure 3.1: Overview of the proposed method. The original Skip-gram model is illustrated on the upper level.

demonstrate the contribution of this work to the task of relation extraction. We confirm that the proposed method improves not only the quality of distributed representations for relational patterns but also that for words.

3.1 Proposed method

The proposed method bases on the Skip-gram model described in Section 2.1.1 and Recursive NN. Therefore, we first review the Recursive NN in Section 3.1.1 followed by the proposed method.

3.1.1 Recursive Neural Network

Recursive neural network computes the semantic vectors of phrases based on compositionality [Socher et al., 2011b]. Using a weight matrix $M \in \mathbb{R}^{d \times 2d}$ and an activation function g (e.g., $\tanh$), Recursive NN computes the semantic vector of the phrase consisting of two words w_a and w_b,

$$g\left(M \begin{bmatrix} \boldsymbol{v}_{w_a} \\ \boldsymbol{v}_{w_b} \end{bmatrix}\right). \tag{3.1}$$

The vector computed by Equation 3.1 is expected to represent the meaning of the phrase based on semantic compositionality. Socher et al. [2011b] apply this function recursively inside a binarized parse tree, and compose the semantic vectors of phrases and sentences. Although the study modeled only one compositional function with a single matrix M, Socher et al. [2012] extended Recursive NN to matrix-vector recur-

sive neural network (MV-RNN) in order to configure a compositional function for each word, assigning a word with both a vector and a matrix.

3.1.2 Semantic composition for relational patterns

We extend the Skip-gram model to enable it to take into account the semantic composition for relational patterns. We provide an overview of the proposed method using the example in Figure 3.1. Here, we have a sequence of lemmatized words "yeast help reduce the serious risk of infection". As explained in Section 1, it is inefficient to regard the relational pattern "*X* help reduce the serious risk of *Y*" as a single 'word' (upper). Instead, we compute the semantic vector from the constituent words of the relational pattern, e.g., 'help', 'reduce', 'serious', and 'risk'. Simultaneously, we would like to handle cases in which words have a major influence on changing the meaning of the entire phrase.

Inspired by Socher et al. [2012], we represent the words inheriting or changing the meaning with matrices in Recursive NN. In this paper, we assume that verbs appearing frequently in relational patterns may inherit or change the meaning computed by other constituent words. We call these verbs *transformational verbs*[1]. In the example in Figure 3.1, we may think that 'reduce' changes the meaning of 'risk' and 'help' inherits the meaning of "reduce the serious risk of"; and the change and inheritance are represented by matrices.

To compute the distributed representation of the relational pattern "*X* help reduce the serious risk of *Y*", the proposed method first computes the distributed representation of "the serious risk of". In this study, we assume additive compositionality for words except for transformational verbs. For this reason, the proposed method obtains the semantic vector for "the serious risk of" by computing the mean of the distributed representations of 'serious' and 'risk'. Next, the proposed method multiplies the semantic vector for "the serious risk of" and the matrix for 'reduce', and then multiplies the computed vector and the matrix for 'help'. To learn the parameters in matrices and vectors, we incorporate the Recursive NN framework into the Skip-gram model. This is not only because the Skip-gram model achieved successes in training high-quality

[1]Transformational verbs are similar to *light verbs* and *catenative verbs*, but it is hard to give a formal definition.

word vectors from a large corpus, but also because the online training algorithm (word-by-word) is suitable for incorporating matrix-vector compositions used in Recursive NN.

Meanwhile, giving a formal definition of transformational verbs is arguable. In this study, we make three assumptions for identifying transformational verbs.

1. Verbs can behave as transformational verbs.
2. Whether a verb is a transformational verb or not is determined by the statistics of its occurrences in relational patterns.
3. Other words, e.g, nouns, adjectives, adverbs, and verbs not qualified to be transformational verbs express meanings of their own. We call these words *content words*.

Although these assumptions are rather provisional, we would like to explore the possibility of semantic compositoinality for relational patterns.

We assume that the relational pattern P is composed of transformational verbs $p_1, ..., p_n$, followed by content words $p_{n+1}, ..., p_m$. For example, the lemmatized relational pattern "help reduce the serious risk of" is composed of the transformational verbs 'help' and 'reduce' as well as 'the', 'serious', 'risk', 'of', as shown in Figure 3.1. Removing non-content words such as determiners and prepositions, we obtain content words 'serious' and 'risk'. Accordingly, the relational pattern "help reduce the serious risk of" is represented by $P = (p_1, p_2, p_3, p_4) = (\text{help}, \text{reduce}, \text{serious}, \text{risk})$. The total number of words in P is $m = 4$, and the boundary between the transformational verbs and content words is $n = 2$. Although formal definitions of relational pattern, transformational verbs, and content words are open questions, we mine them from the corpus (refer to Section 3.2.1).

As previously mentioned, in this study, we assume additive compositionality for content words in a relational pattern. That is to say, the meaning of content words $p_{n+1}, ..., p_m$ is computed from the mean of the semantic vectors corresponding to the content words,

$$\frac{\boldsymbol{v}_{p_{(n+1)}} + \boldsymbol{v}_{p_{(n+2)}} + ...\boldsymbol{v}_{p_m}}{m - n}. \tag{3.2}$$

In contrast, we assume that each transformational verb p_i inherits or transforms a given semantic vector using the mapping function, $f_{p_i} : \mathbb{R}^d \rightarrow \mathbb{R}^d$. Hence, the semantic

vector $\boldsymbol{v}_P$ for the relational pattern P is computed as,

$$\boldsymbol{v}_P = f_{p_1}\Big(f_{p_2}\Big(...f_{p_n}\Big(\tanh\Big(\frac{\boldsymbol{v}_{p_{(n+1)}}+\boldsymbol{v}_{p_{(n+2)}}+...\boldsymbol{v}_{pm}}{m-n}\Big)\Big)\Big)\Big). \tag{3.3}$$

We design the mapping function f_{p_i} using Recursive NN [Socher et al., 2011b]. More specifically, the mapping function for the transformational verb p_i is modeled using a matrix $W_i \in \mathbb{R}^{d\times d}$ and an activation function.

$$f_{p_i}(\boldsymbol{v}) = \tanh(W_{p_i}\boldsymbol{v}) \tag{3.4}$$

In short, the proposed method computes the meaning of content words of a relational pattern as the vector mean, and inherits/transforms the meaning using matrix-vector products of Recursive NN.

3.1.3 Training

The proposed method is identical to the Skip-gram model when a context window involves no relational pattern. In other words, we train $\boldsymbol{v}_w$ and $\tilde{\boldsymbol{v}}_c$ in the same manner as the original Skip-gram model with negative sampling. We summarize the differences from the original Skip-gram model.

1. We treat a relational pattern as a 'word', but its semantic vector is computed using Formula 3.3. We update the vectors for content words and matrices for transformational verbs to enable the composed vector of the relational pattern P to predict context words c well.

2. In addition to word vectors $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}$, we train semantic matrices W for the transformational verbs. We use backpropagation for updating vectors and matrices.

3. We do not use Formula 3.3 for computing a context vector of a relational pattern. In other words, when the negative sampling picks a relational pattern for a centered word, we use a context vector $\tilde{\boldsymbol{v}}$ assigned for the pattern.

4. We apply the activation function $\tanh$ even for word vectors $\boldsymbol{v}$. This keeps the value range of semantic vectors consistent between composed vectors and word

vectors. Each dimension of a semantic vector of a relational pattern is bound to the range of $(-1, 1)$, because Formula 3.3 uses $\tanh$ as an activation function.

Meanwhile, some transformational verbs (e.g., light verbs) may not contribute to meanings. For example, the word 'take' in the pattern "take care of" does not have a strong influence on the meaning of the pattern. Thus, we explore the use of l_1-regularization to encourage diagonal matrices. We modify the objective function (Equation 2.4) into:

$$J' = -\sum_{w \in \mathcal{D}} \sum_{c \in C_w} \log p(c|w) + \lambda \sum_{W \in \mathbb{W}} r(W). \tag{3.5}$$

Here, $\mathbb{W}$ represents the set of all matrices for the transformational verbs. The function $r(W)$ computes the l_1-norm from off-diagonal elements of W,

$$r(W) = \sum_{i \neq j} |W_{i,j}|. \tag{3.6}$$

3.2 Experiments

3.2.1 Corpora and training settings

We used ukWaC[1] as the corpus for training the semantic vectors and matrices. This corpus includes the text of Web pages crawled from the .uk domain, and contains 2 billion words. This corpus also includes parts-of-speech tags and lemmas annotated by the TreeTagger[2]. In our experiment, we lowercased words and used the lemmas except for past participle forms of verbs (we used their surface forms)[3]. Furthermore, tokens consisting of a single character (e.g., 'a' and 'b'), determiners (e.g., 'the'), interrogative words (e.g., 'what'), and prepositions were removed as stop words.

We applied Reverb [Fader et al., 2011] to the ukWaC corpus to extract relational pattern candidates. To remove unuseful relational patterns, we followed the filtering rules that are compatible with the ones used in the publicly available extraction result[4]:

[1] `http://wacky.sslmit.unibo.it/doku.php?id=corpora`
[2] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`
[3] We use past particle forms to express the passive/active voice.
[4] `http://reverb.cs.washington.edu/`

the confidence score of a pattern must be no less than 0.9 at least once in the corpus, a relational pattern must not contain a temporal expression (e.g., 'yesterday' and 'tonight'), and the frequency of occurrence of a pattern must be no less than 5. Additionally, throughout the experiments, we removed relational patterns that appear in the evaluation data in order to examine the performance of the proposed method in composing semantic vectors of unseen relational patterns. After the above preprocessing, we obtained $55,885$ relational patterns.

Verbs appearing in five or more kinds of relational patterns were identified as transformational verbs. We removed the verb 'be' in this experiment. Using the criterion, we identified 697 verbs as transformational verbs in relational patterns. If a relational pattern consists only of transformational verbs, we regarded the last word as a content word. While there may be some room for consideration regarding the definition of transformational verbs and content words, we used these criteria in this experiment.

When training the proposed method, we removed words and relational patterns appearing less than 10 times. As a result, we obtained approximately 0.7 million words (including relational patterns) as targets for training semantic vectors. When a transformational verb appears outside of a relational pattern (e.g., 'reduce'), we update a vector for the word similarly as for an ordinary word.

For comparing with the existing methods, we used the same hyper-parameters as the ones presented in the papers [Mikolov et al., 2013b; Socher et al., 2012]. We set the number of dimensions $d = 50$, following Socher et al. [2012]. For the width of context window h, number of negative samples k, and subsampling parameter in the Skip-gram, we used the same hyper-parameters as in Mikolov et al. [2013b]: $h = 5$, $k = 5$, and subsampling with 10^{-5}. We initialize word vectors $\boldsymbol{v}$ and context vectors $\tilde{\boldsymbol{v}}$ using the result from the original Skip-gram model [Mikolov et al., 2013b]. Elements in semantic matrices W are initialized with random values sampled from a Gaussian distribution with mean 0 and variance 0.1. We learn parameters ($\boldsymbol{v}$, $\tilde{\boldsymbol{v}}$, and W) by the backpropagation with the stochastic gradient descent (SGD) method. We control the learning rate α for an instance by using the formula implemented in word2vec[1]:

$$\alpha = \alpha_0 * (1 - \frac{\text{the number of processed sentences}}{\text{the number of total sentences} + 1}). \tag{3.7}$$

[1] `https://code.google.com/p/word2vec/`

In Equation 3.7, α_0 represents the initial learning rate (0.025 in this experiments). Equation 3.7 decreases the learning rate steadily according to the number of processed sentences.

3.2.2 Evaluation datasets

We conducted three experiments, pattern similarity, relation classification, and word similarity.

Pattern similarity We would like to examine whether our proposed method can successfully compose semantic vectors of relational patterns. The performance of a method can be measured by the correlation between similarity judgments of humans for relational patterns and the similarities of the corresponding semantic vectors computed by the method. However, unfortunately, no existing dataset provides similarity judgements between relational patterns. Instead, we adapted the dataset developed for semantic inferences between relational patterns [Zeichner et al., 2012]. Using relational patterns extracted by Reverb, this dataset labels whether a pair of relational patterns (e.g., '*X* prevent *Y*' and "*X* reduce the risk of *Y*") is meaningful[1] or not. A meaningful pair is annotated with a label indicating whether the pair has an inference relation (entailment). The dataset consists of $6,567$ pairs overall.

After discarding pairs labeled meaningless and cases where the set of arguments is reversed between paired patterns such as "*X* contain embedded *Y*" and "*Y* be embedded within *X*", we extracted $5,409$ pairs for evaluation. The evaluation dataset includes $2,447$ pairs with inference relation (similar), $2,962$ pairs without inference relation (dissimilar). This dataset includes only binary decisions (similar or dissimilar) for relational patterns, whereas similarity values computed by a method range in [0, 1.0]. Thus, we regard pattern pairs having similarity values greater than a threshold as 'similar', and the rest as 'dissimilar'. In this way, we can measure the precision and recall of a method for detecting similar relational patterns with the given threshold. By

[1]When an annotator judges a pair, the slots of relational patterns are filled with the same subject and object. If the annotator can easily understand the both of expressions, the pair is meaningful. Take the pair "*X* belong to *Y*" and "*X* be property of *Y*" as an example. If the pair is filled with "Such people" and "the left", it is unrealistic to understand the meaning of "Such people be property of the left". In this case, the pair is annotated with meaningless.

changing the threshold from 0.0 to 1.0, we can draw a precision-recall curve for each method.

Relation classification To examine the contribution of this work to the relation classification task, we used the SemEval-2010 Task 8 dataset [Hendrickx et al., 2010]. The task is to identify the relationship of a given entity pair. The dataset consists of $10,717$ relation instances ($8,000$ training and $2,717$ test instances), each of which is annotated with a relation label. The data set has 19 candidate relation labels, nine directed relationships (e.g.,CAUSE-EFFECT) and one undirected relationship OTHER. For example, the entity pair 'burst' and 'pressure' in the sentence "The burst has been caused by water hammer pressure" is labeled as CAUSE-EFFECT(e_1, e_2).

Word similarity We also evaluated the word vectors to verify that the proposed method does not degrade the quality of word vectors. We used a variety of word similarity datasets: WordSim-353 [Finkelstein et al., 2001], MC [Miller and Charles, 1991], RG [Rubenstein and Goodenough, 1965], and SCWS [Huang et al., 2012]. For each dataset, the numbers of word pairs are 353, 30, 65, and 2,003. For evaluation, we used all word pairs included in the datasets after lowercasing and lemmatizing similarly to the training procedure. We calculate Spearman's rank correlation coefficients between human judgments and cosine similarity values of the semantic vectors computed by each method.

3.2.3 Results

Pattern similarity Figure 3.2 shows precision-recall curves of the proposed method and baseline methods on the pattern-similarity task. The red locus shows the performance of the proposed method. In this figure, we set the parameter for l_1-regularization to 10^6, because the parameter achieved the best performance (Table 3.1). The blue locus corresponds to the Skip-gram model, in which relational patterns are regarded as single words. This treatment is identical to the procedure for training phrase vectors in Mikolov et al. [2013b]. The green locus shows the performance of additive compositions of word vectors trained by the Skip-gram model. In this method, we trained word vectors as usual (without considering relational patterns), and computed vectors

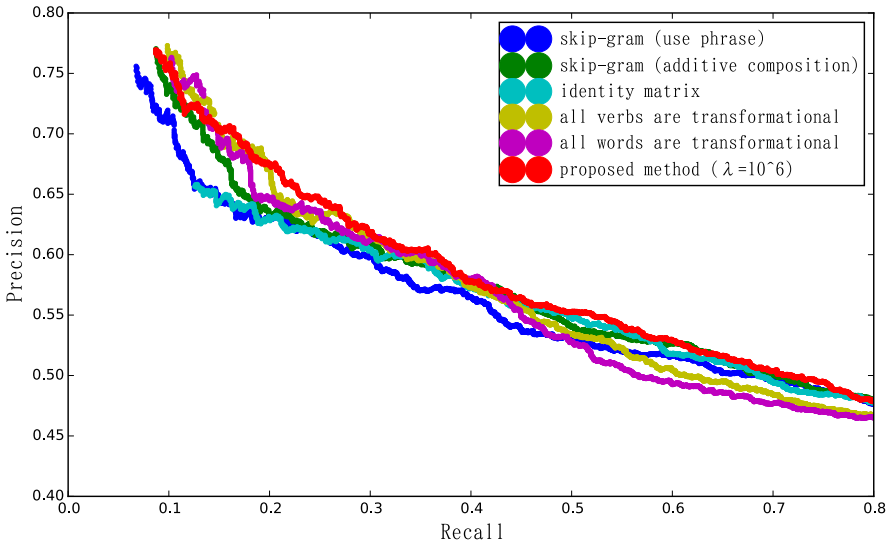


Figure 3.2: Precision-recall curve of each method on the pattern-similarity task

of relational patterns as the mean of vectors of constituent words. This treatment is the popular and strong baseline method to compute a phrase vector from its constituent words [Muraoka et al., 2014]. The light blue locus reports the performance when we fix matrices for transformational verbs as identity matrices: this corresponds to ignoring transformational verbs in a relational pattern. The yellow and purple loci correspond to assigning every verb (yellow) and every word (purple), respectively, with a matrix (rather than a vector). In these settings, we use a vector representation for a content word that are located at the end of relational patterns. In other words, these settings are more flexible than the recommended setting (red) for composing semantic vectors, having more free parameters to train the models.

Figure 3.2 shows that the proposed method performed better than all baseline methods. It is noteworthy that the proposed method performed better than the Skip-gram model with additive composition in green, which has been regarded as a strong baseline for semantic composition. This result indicates that representing transformational verbs with matrices in Recursive NN is more suitable than additive composition for computing the semantic vectors of relational patterns.

The proposed method outperformed the setting with transformational verbs ignored (light blue). This result indicates that the treatment for transformational verbs is important for composing semantic vectors of relational patterns. In fact, the proposed

Method	AUC	Sparsity
Skip-gram (phrase)	0.557	—
Skip-gram (additive)	0.568	—
Identity matrix	0.552	—
Every verb has a matrix	0.566	—
Every word has a matrix	0.561	—
Proposed ($\lambda = 0$)	0.570	0.0%
Proposed ($\lambda = 1$)	0.570	0.0%
Proposed ($\lambda = 10$)	0.570	0.7%
Proposed ($\lambda = 10^2$)	0.573	14.4%
Proposed ($\lambda = 10^3$)	0.574*	54.4%
Proposed ($\lambda = 10^4$)	0.575*	88.2%
Proposed ($\lambda = 10^5$)	**0.576***	96.6%
Proposed ($\lambda = 10^6$)	**0.576***	97.8%
Proposed ($\lambda = 10^7$)	0.575*	98.0%

Table 3.1: Area under the curve (AUC) of each method on the pattern-similarity task. This table also reports the sparsity (the ratio of zero-elements) of matrices with different parameters for l_1-regularization. If a method outperforms the Skip-gram (additive) with 95% statistical significance ($p < 0.05$), we put * on the value of AUC.

method successfully computes semantic vectors of relational patterns with inhibitory verbs: for example, the proposed method could predict the similarity between "prevent the growth of" and 'inhibit' while the use of identity matrices cannot.

The comparison among the proposed method (red), representing every verb with a matrix (yellow), and representing every word with a matrix (purple) demonstrates the effectiveness of identifying transformational verbs in advance. This result suggests that transformational verbs (verbs appearing frequently in relational patterns) can inherit/change the meaning and that it is important to incorporate their behaviors in composing semantic vectors of relational patterns.

Training semantic vectors of relational patterns by regarding a relation pattern as a single word (blue) performed worse than most of other methods in this experiment. This suggests the difficulty in learning vector representations of relational patterns only with the distributional hypothesis. The incorporation of the distributional hypothesis with the semantic compositionality is the key to success in modeling semantic vectors of relational patterns.

Table 3.1 shows the area under the curve (AUC) of each method appearing in Fig-

ure 3.2. In addition to AUC values, we report the sparsity (the percentage of zero elements in matrices), changing the parameter for l_1-regularization from 0 to 10^7 in powers of ten[1]. Again, we can reconfirm from this table that the proposed method outperforms the baseline methods. Moreover, the methods with $\lambda = 10^3, 10^4, 10^5, 10^6, \text{and } 10^7$ outperform the strong baseline, Skip-gram (additive) with 95% statistical significance ($p < 0.05$) measured by paired bootstrap resampling [Koehn, 2004]. The proposed method obtained the best performance (0.576) with over 95% sparsity ($\lambda = 10^5$ and 10^6). The results indicate that the use of l_1-regularization for off-diagonal elements of matrices improves the performance even though the obtained model becomes compact. However, the model with $\lambda = 10^7$ was too sparse to achieve the best performance.

Table 3.1 also shows that representing every verb with a matrix or representing every word with a matrix performed worse than Skip-gram (additive). This also suggests that it is essential to distinguish transformational verbs from content words. In addition, representing content words with matrices gave too much flexibility for this task.

Relation classification Table 3.2 shows the performance of each method on the relation classification task. The top 4 rows represent the results of the baseline method and improvements by using semantic vectors computed by the proposed method. To predict whether a given entity pair has a specific relation, we built one-versus-one classifiers modeled by SVM with radial basis function (RBF) kernel. We defined basic features for the classifiers: parts-of-speech tags, surface forms, and lemmas of words appearing between an entity pair, and lemmas of the words in the entity pair. In addition, we included the value of each dimension of the semantic vectors of a relational pattern and entity pairs as features in order to examine the effect of the semantic vectors obtained by the proposed method with ($\lambda = 10^6$). Moreover, we employed named entity information and WordNet super sense classes predicted by a super sense tagger [Ciaramita and Altun, 2006]. We used LIBSVM[2] for training SVM models. For hyper-parameters, we determined $C = 8.0$ and $\gamma = 0.03125$ based on 5-fold cross-validation.

Table 3.2 shows that the use of semantic vectors of the proposed method boosted the performance from 76.0 to 79.0 F1 scores. Moreover, even with the external knowl-

[1]We stopped increasing λ to 10^7 because the AUC decreased when we changed λ from 10^6 to 10^7.
[2]`https://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Method	Features	F1
SVM	basic features	76.0
(Use semantic vectors	basic features, semantic vectors	79.0
obtained by the proposed method)	basic features, WordNet, NE	79.9
	basic features, semantic vectors, WordNet, NE	82.1
SVM (Best in SemEval 2010) [Rink and Harabagiu, 2010]	POS, prefixes, morphological, WordNet, dependency parse, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
Recursive NN	-	74.8
[Socher et al., 2011b]	WordNet, NE	77.6
MV-RNN	-	79.1
[Socher et al., 2012]	WordNet, NE	82.4
CNN [Zeng et al., 2014]	WordNet	82.7
FCM	-	80.6
[Yu et al., 2014]	dependency parse, NE	83.0
CR-CNN [dos Santos et al., 2015]	-	84.1
RelEmb	-	82.8
[Hashimoto et al., 2015]	dependency parse, WordNet, NE	83.5
depLCNN+NS	-	84.0
[Xu et al., 2015]	WordNet	85.6

Table 3.2: Comparison of using the proposed method with previously published results.

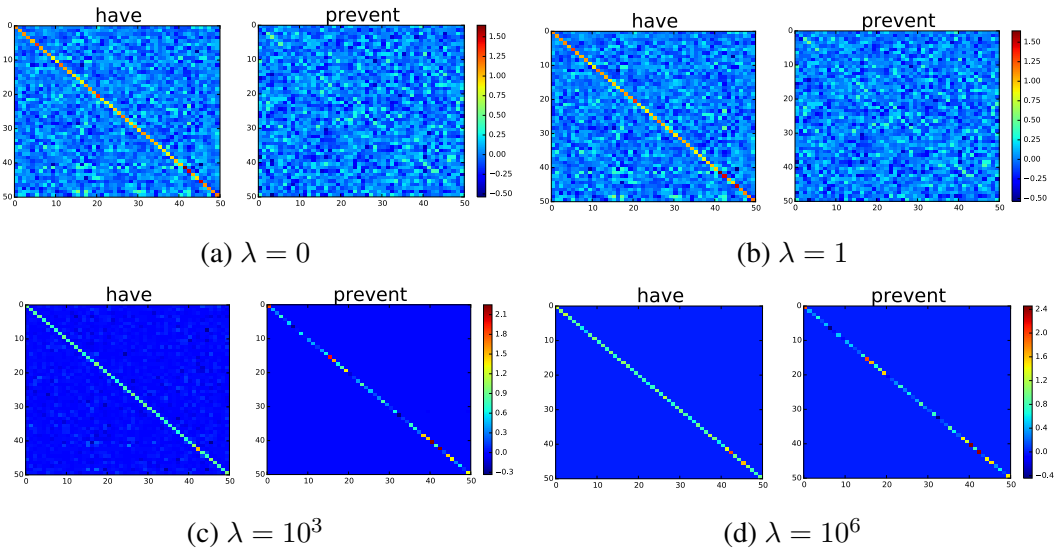
edge (WordNet super sense), semantic vectors computed by the proposed method improved the performance from 79.9 to 82.1. This demonstrates the usefulness of the semantic vectors computed by the proposed method for the task of relation classification.

For comparison, Table 3.2 includes the performance reported in the previous work. Table 3.2 shows that using our semantic vectors exhibited performance closed to the best method in the SemEval-2010 task 8 competition [Rink and Harabagiu, 2010]. The proposed method outperformed Recursive NN [Socher et al., 2011b] by a large margin. Moreover, the proposed method achieved a comparable performance with MV-RNN [Socher et al., 2012].

However, the best result obtained by the proposed method was lower than that of the state-of-the-art methods. These methods specially train vector representations for words such that it predicts predefined relation labels in the SemEval-2010 Task

Method	WS353	MC	RG	SCWS
Baseline (Skip-gram without relational patterns)	63.0	69.5	74.2	60.3
Proposed ($\lambda = 10^6$)	**68.4**	**73.7**	**75.4**	**61.5**

Table 3.3: Spearman's rank correlation coefficients on the word similarity tasks.



(a) $\lambda = 0$ (b) $\lambda = 1$

(c) $\lambda = 10^3$ (d) $\lambda = 10^6$

Figure 3.3: Examples of the matrices learned using the proposed method ($\lambda = 0$, 1, 10^3, and 10^6).

8 dataset. In other words, they fine-tuned vector representations for this task. In fact, Yu et al. [2014] reported that fine-tuning improved the performance. In addition, Hashimoto et al. [2015] indicated that they achieved a better performance when they refined vector representations for initialization. Therefore, we may obtain a further improvement if we tune matrices and vectors in our model specialized for the SemEval-2010 Task 8. In contrast, our focus is to model semantic composition of relational patterns in a generic and unsupervised fashion. We will explore the possibility of fine-tuning in future work.

Word similarity Table 3.3 reports the results for word similarity on four different datasets. In this task, we compared the proposed method with the Skip-gram model ignoring relational patterns. Table 3.3 shows that the proposed method yielded the better performance than the Skip-gram model without relational patterns. In other

words, the result indicates that our approach also improved the quality of the semantic vectors of words.

3.2.4 Visualizing the matrices

Figure 3.3 shows a visualization of matrices for the words 'have' and 'prevent' learned by the proposed method with different parameters for l_1-regularization ($\lambda = 0$, 1, 10^3, and 10^6). The values of the diagonal elements in the matrix for 'have' are high while the off-diagonal elements are close to zero. In other words, the matrix for 'have' is close to the identity matrix, implying that the word 'have' inherits the meaning from content words. The proposed method learned this behavior because a number of relational patterns (e.g., "have access to" and "have an impact on") include the word 'have', but their contexts are similar to those for content words (e.g., 'access' and 'impact'). We could observe the similar tendency for verbs such as 'make' and 'take'.

In contrast, the matrix for 'prevent' is entirely different from that for 'have'. With the small l_1-regularization parameters ($\lambda = 0$ and 1), the matrix for 'prevent' does not have an obvious tendency. With the large l_1-regularization parameters ($\lambda = 10^3$ and 10^6), the matrix is close to the diagonal matrix. However, the matrix is different from the identity matrix: each diagonal element have a non-uniform value. This is probably because the word 'prevent' tends to negate the meaning of content words, as in "prevent the growth of". Thus, the proposed method found a matrix so that it does not pass the meaning of the content word (e.g., 'growth') directly to that of the whole.

3.3 Related Work

Relation Extraction A number of previous studies extracted semantic relations between entities using linguistic patterns [Carlson et al., 2010; Min et al., 2012; Nakashole et al., 2012; Pantel and Pennacchiotti, 2006; Rosenfeld and Feldman, 2007]. These studies mostly explored methods for obtaining relation instances with high-precision e.g., using pointwise mutual information between entity pairs and patterns [Pantel and Pennacchiotti, 2006], checking types of arguments of relational patterns [Rosenfeld and Feldman, 2007], and extracting entities and relation instances simultaneously [Carlson et al., 2010]. On the other hand, Min et al. [2012] improved recall

by incorporating various knowledge sources into the extracting algorithm. However, these approaches suffer from the data sparseness problem described in Section 1.

Nakashole et al. [2012] presented *PATTY*, a large resource for relational patterns. PATTY has an automatic method for inducing rules for generalizing relational patterns with part-of-speech tags, wildcards, and argument types. For example, PATTY can generalize the relational pattern "*singer* sings her *song*" into "*singer* sings [prp] *song*", where [prp] represents a pronoun. This approach could reduce the data sparseness problem to some extent, but could not model the compositionality of relational patterns, e.g., similarity between words in two relational patterns.

Semantic composition Mitchell and Lapata [2010] demonstrated the ability of computing the meaning of a phrase from constituent words. They explored various functions for composing phrase vectors, e.g., additive and multiplicative compositions. Mikolov et al. [2013b] proposed the Skip-gram model, which was inspired by neural language models [Bengio et al., 2003; Collobert and Weston, 2008]. The Skip-gram model exhibits additive compositionality. Levy and Goldberg [2014b] and Levy and Goldberg [2014a] provided theoretical analyses of additive compositionality of the Skip-gram model with negative sampling. Pennington et al. [2014] demonstrated that semantic composition could be modeled also by a co-occurrence matrix between words and their context words. Although these studies achieved good performance in additive composition, they cannot model the case in which a word such as 'prevent' or 'inhibit' changes the meaning of an entire phrase, e.g., "prevent the growth of." Baroni and Zamparelli [2010] suggested representing modifiers with matrices rather than with vectors.

MV-RNN [Socher et al., 2012], which is the extension method of Recursive NN [Socher et al., 2011b], can handle a word changing the meaning but they requires supervision data for specific tasks (e.g., sentiment analysis). In addition, those authors did not determine whether the vector representation of internal nodes of a tree really exhibits the meanings of the phrases. Muraoka et al. [2014] proposed a method that reduces MV-RNN parameters. The method uses a single matrix for composing a phrase with the same part-of-speech pattern (e.g., adj–noun). However, they did not evaluate the method for composing a phrase vector from three or more words. Socher et al. [2011a] proposed a method to learn word vectors and a matrix from an unlabeled corpus us-

ing an autoencoder but this approach uses only a single matrix for vector composition. In other words, the method cannot take modification of each word into account. Hashimoto et al. [2014] proposed a method for training weights for linear combinations of word vectors. Although Their method jointly learns the vector representation and weighting factors of words from an unlabeled corpus, they cannot model changing aspects of words without the capability of linear transformations.

3.4 Conclusion

In this chapter, we proposed a novel method for computing the meanings of relational patterns based on semantic compositionality. We extended the Skip-gram model to incorporate semantic compositions modeled by Recursive NNs. In addition, we introduced l_1-regularization to obtain a simpler model. The experimental results showed that the proposed method can successfully model semantic compositions of relational patterns, outperforming strong baselines such as additive composition. The experiments also demonstrated the contribution of this work to the task of relation classification. We confirmed that the proposed method could improve not only the quality of vectors for relational patterns but also that for words.

In this study, we defined transformational verbs heuristically. Even though this study could demonstrate superiority in handling transformational verbs, we need to explore a better approach for determining whether a word should have a vector or matrix.

Chapter 4

Data Construction for Modeling Semantic Similarity

In the previous chapter, we use inference relation dataset to evaluate the ability to compute the meanings of relational patterns. However, the inference relation dataset is insufficient for the evaluation because inference relations are quite different from semantic similarity. This chapter describes the difference between inference relations and semantic similarity and presents a new dataset representing semantic similarity for relational pattern pairs.

4.1 Data Construction

4.1.1 Target relation instances

We build a new dataset upon the work of Zeichner et al. [2012], which consists of relational patterns with semantic inference labels annotated. The dataset includes 5,555 pairs[1] extracted by Reverb [Fader et al., 2011], 2,447 pairs with inference relation and 3,108 pairs (the rest) without one.

Initially, we considered using this high-quality dataset as it is for semantic modeling of relational patterns. However, we found that inference relations exhibit quite

[1]More precisely, the dataset includes 1,012 *meaningless* pairs in addition to 5,555 pairs. A pair of relational patterns was annotated as *meaningless* if the annotators were unable to understand the meaning of the patterns easily. We ignore the *meaningless* pairs in this study.

different properties from those of semantic similarity. Take a relational pattern pair "*X* be the part of *Y*" and "*X* be an essential part of *Y*" filled with "*X* = the small intestine, *Y* = the digestive system" as an instance. The pattern "*X* be the part of *Y*" does not entail "*X* be an essential part of *Y*" because the meaning of the former does not include 'essential'. Nevertheless, both statements are similar, representing the same relation (PART-OF). Another uncomfortable pair is "*X* fall down *Y*" and "*X* go up *Y*" filled with "*X* = the dude, *Y* = the stairs". The dataset indicates that the former entails the latter probably because falling down from the stairs requires going up there, but they present the opposite meaning. For this reason, we decided to re-annotate semantic similarity judgments on every pair of relational patterns on the dataset.

4.1.2 Annotation guideline

We use instance-based judgment in a similar manner to that of Zeichner et al. [2012] to secure a high inter-annotator agreement. In instance-based judgment, an annotator judges a pair of relational patterns whose variable slots are filled with the same entity pair. In other words, he or she does not make a judgment for a pair of relational patterns with variables, "*X* prevent *Y*" and "*X* reduce the risk of *Y*", but two instantiated statements "Cephalexin prevent the bacteria" and "Cephalexin reduce the risk of the bacteria" ("*X* = Cephalexin, *Y* = the bacteria"). We use the entity pairs provided in Zeichner et al. [2012].

We asked annotators to make a judgment for a pair of relation instances by choosing a rating from 1 (dissimilar) to 7 (very similar). We provided the following instructions for judgment, which is compatible with Mitchell and Lapata [2010]: (1) rate 6 or 7 if the meanings of two statements are the same or mostly the same (e.g., "Palmer team with Jack Nicklaus" and "Palmer join with Jack Nicklaus"); (2) rate 1 or 2 if two statements are dissimilar or unrelated (e.g., "the kids grow up with him" and "the kids forget about him"); (3) rate 3, 4, or 5 if two statements have some relationships (e.g., "Many of you know about the site" and "Many of you get more information about the site", where the two statements differ but also reasonably resemble to some extent).

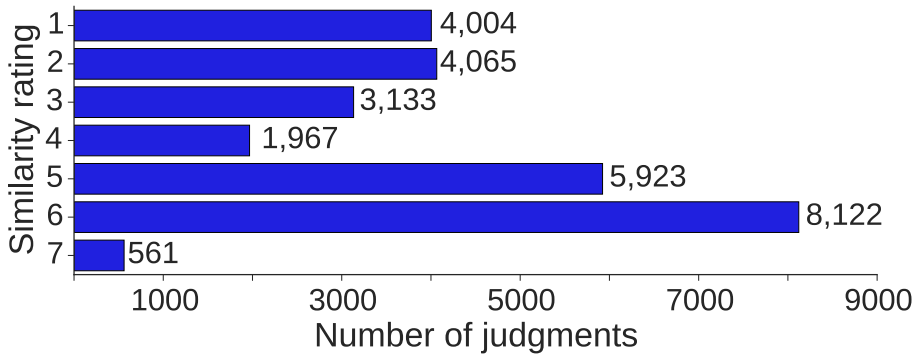


Figure 4.1: Number of judgments for each similarity rating. The total number of judgments is $27,775$ ($5,555$ pairs $\times$ 5 workers).

4.1.3 Annotation procedure

We use a crowdsourcing service CrowdFlower[1] to collect similarity judgments from the crowds. CrowdFlower has the mechanism to assess the reliability of annotators using Gold Standard Data (Gold, hereafter), which consists of pairs of relational patterns with similarity scores assigned. Gold examples are regularly inserted throughout the judgment job to enable measurement of the performance of each worker[2]. Two authors of this paper annotated 100 pairs extracted randomly from 5,555 pairs, and prepared 80 Gold examples showing high agreement. Ratings of the Gold examples were used merely for quality assessment of the workers. In other words, we discarded the similarity ratings of the Gold examples, and used those judged by the workers.

To build a high quality dataset, we use judgments from workers whose confidence values (reliability scores) computed by CrowdFlower are greater than 75%. Additionally, we force every pair to have at least five judgments from the workers. Consequently, 60 workers participated in this job. In the final version of this dataset, each pair has five similarity ratings judged by the five most reliable workers who were involved in the pair.

Figure 4.1 presents the number of judgments for each similarity rating. Workers seldom rated 7 for a pair of relational patterns, probably because most pairs have at least one difference in content words. The mean of the standard deviations of similarity ratings of all pairs is 1.16. Moreover, we computed Spearman's ρ between similarity

[1] `http://www.crowdflower.com/`

[2] We allow ± 1 differences in rating when we measure the performance of the workers.

judgments from each worker and the mean of five judgments in the dataset. The mean of Spearman's ρ of workers involved in the dataset is 0.728. These statistics show a high inter-annotator agreement of the dataset.

4.2 Related Work

Mitchell and Lapata [2010] was a pioneering work in semantic modeling of short phrases. They constructed the dataset that contains two-word phrase pairs with semantic similarity judged by human annotators. Korkontzelos et al. [2013] provided a semantic similarity dataset with pairs of two words and a single word. Wieting et al. [2015] annotated a part of PPDB [Ganitkevitch et al., 2013] to evaluate semantic modeling of paraphrases. Although the target unit of semantic modeling is different from that for these previous studies, we follow the annotation guideline and instruction of Mitchell and Lapata [2010] to build the new dataset.

The task addressed in this chapter is also related to the Semantic Textual Similarity (STS) task [Agirre et al., 2012]. STS is the task to measure the degree of semantic similarity between two sentences. Even though a relational pattern appears as a part of a sentence, it may be difficult to transfer findings from one to another: for example, the encoders of RNN and its variants explored in this study may exhibit different characteristics, influenced by the length and complexity of input text expressions.

4.3 Conclusion

In this chapter, we addressed to construct a new dataset in which humans rated multiple similarity scores for every pair of relational patterns on the dataset of semantic inference [Zeichner et al., 2012]. The new dataset shows a high inter-annotator agreement, following the annotation guideline of Mitchell and Lapata [2010]. The dataset is publicly available on the Web site[1].

[1] `http://github.com/takase/relPatSim`

Chapter 5

Composing Distributed Representations of Phrases with Gated Additive Composition

Modified recursive neural network proposed in Chapter 3 is ad hoc because it requires identifying verbs and other content words in advance. Moreover, the method applies additive composition to the content words but additive composition cannot handle the importance of each word. In this chapter, to address the problem, we propose a more general neural encoder: Gated Additive Composition.

We also conduct a comparative study of well known encoders on phrase composition task. We use several datasets including relational pattern dataset constructed in Chapter 4. In addition, we investigate the enhancement by distributed representations on the relation classification task. Through experiments, we explore the best way to compose distributed representations of phrases, usefulness of the constructed relational pattern dataset, and contribution of distributed representations to NLP applications.

5.1 Encoders

We adopt additive composition and some neural methods as encoders for phrase composition. In this section, we explain each method briefly.

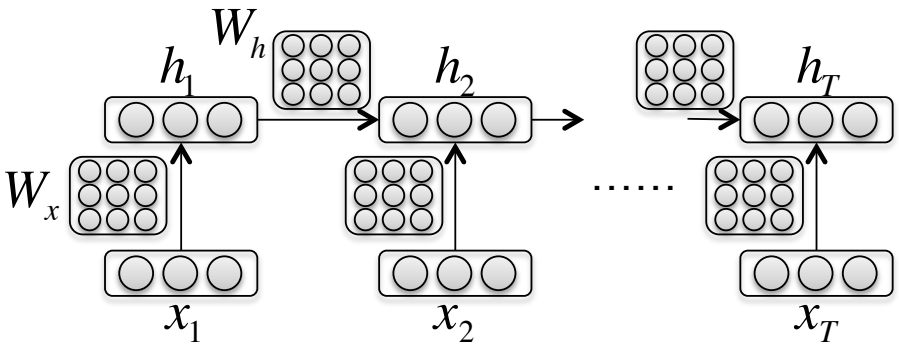


Figure 5.1: The RNN architecture.

5.1.1 Additive Composition

Additive composition, which is a simple way to embed phrases into distributed representations, is computing the mean of distributed representations of constituent words [Mitchell and Lapata, 2010]. Presuming that a target phrase consists of a sequence of T words $w_1, ..., w_T$, then we let $x_t \in \mathbb{R}^d$ the distributed representation of the word w_t. This composition method computes $\frac{1}{T}\sum_{t=1}^{T} x_t$ as the distributed representation of the target phrase. Additive composition is not only known as a strong baseline among various methods empirically [Muraoka et al., 2014] but also regarded as an approximation of distributed representations of phrases theoretically [Tian et al., 2016]. However, additive composition ignores the order of word sequence and the importance of each word. Consider the phrase "have access to", whose meaning is mostly identical to that of the verb "access". Because 'have' in the phrase is a light verb and 'to' is a functional word, it may be harmful to compose distributed representations of 'have' and 'to' to compute the meaning of the phrase.

5.1.2 Recurrent Neural Network

Recently, a number of studies reported that Recurrent Neural Network (RNN) and its variants are successful in composing distributed representations of phrases and sentences [Cho et al., 2014; Sutskever et al., 2014]. RNN and its variants can handle the order of word sequence because they scan word by word following the order. For a given distributed representation x_t at position t, the vanilla RNN [Elman, 1990] com-

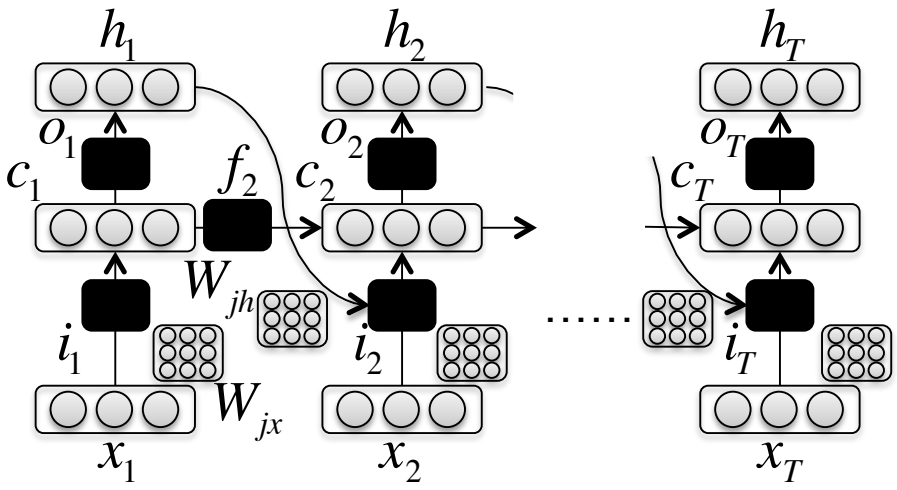


Figure 5.2: The LSTM architecture.

putes the hidden state $h_t \in \mathbb{R}^d$ by the following recursive equation

$$h_t = g(W_x x_t + W_h h_{t-1} + b). \tag{5.1}$$

Here, W_x and W_h are $d \times d$ matrices, $b \in \mathbb{R}^d$ is a bias term, and $g(.)$ is the elementwise activation function ($\tanh$ used in usual). We set $h_0 = 0$ at $t = 1$. In essence, RNN computes the hidden state h_t based on the one at the previous position (h_{t-1}) and the distributed representation x_t of the word w_t as shown in Figure 5.1. Applying Equation 5.1 from $t = 1$ to T, we use h_T as the distributed representation of the phrase.

Recursive Neural Network [Socher et al., 2011b] is regarded as an extension of RNN on a tree structure such as a parse tree. Although we are interested in exploring usefulness of a syntactic tree, we omit to investigate the issue in this study. Therefore, in this study, we focus on computing the distributed representation of given word sequence without any other linguistic information.

5.1.3 RNN Variants

It is hard for RNN to handle long distance dependencies due to the vanishing or exploding gradient problem in training procedure. To deal with the problems, Hochreiter and Schmidhuber [1997] proposed Long Short-Term Memory (LSTM) that is RNN

combined with a gate mechanism. LSTM consists of the input gate $i_t \in \mathbb{R}^d$, forget gate $f_t \in \mathbb{R}^d$, output gate $o_t \in \mathbb{R}^d$, memory cell $c_t \in \mathbb{R}^d$, and hidden state $h_t \in \mathbb{R}^d$. For a given distributed representation x_t at position t, LSTM computes each value by the following equations:

$$j_t = g(W_{jx}x_t + W_{jh}h_{t-1} + b_j), \tag{5.2}$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i), \tag{5.3}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f), \tag{5.4}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o), \tag{5.5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot j_t, \tag{5.6}$$

$$h_t = o_t \odot g(c_t). \tag{5.7}$$

In these equations, W_{jx}, W_{jh}, W_{ix}, W_{ih}, W_{fx}, W_{fh}, W_{ox}, W_{oh} are $d \times d$ matrices, $b_j, b_i, b_f, b_o \in \mathbb{R}^d$ are bias terms, $\sigma(.)$ is the elementwise sigmoid function, and the operator $\odot$ calculates elementwise multiplications. We set $c_0 = 0$ and $h_0 = 0$ at $t = 1$. In short, LSTM computes the hidden state h_t and the memory cell c_t based on those at the previous position (h_{t-1} and c_{t-1}) and the distributed representation x_t of the input word w_t. Figure 5.2 shows the way to obtain the distributed representation of the phrase by using LSTM. Figure 5.2 omits the calculation of each gate to be easy to understand the overview. In addition to resolving vanishing (or exploding) gradient problem, Takase et al. [2016b] implied that a gate mechanism helps to handle the importance of each word in phrase composition. For example, the input gate i_t closes when an input word is a functional word. Moreover, LSTM is commonly used to model sentences [Sutskever et al., 2014]. Applying these equations from $t = 1$ to T, we use h_T as the distributed representation of the phrase.

Cho et al. [2014] proposed Gated Recurrent Unit (GRU), that is the competitive method with LSTM. GRU also has a gate mechanism: the reset gate $r_t \in \mathbb{R}^d$ and the update gate $z_t \in \mathbb{R}^d$. As shown in Figure 5.3, GRU computes the hidden state $h_t \in \mathbb{R}^d$

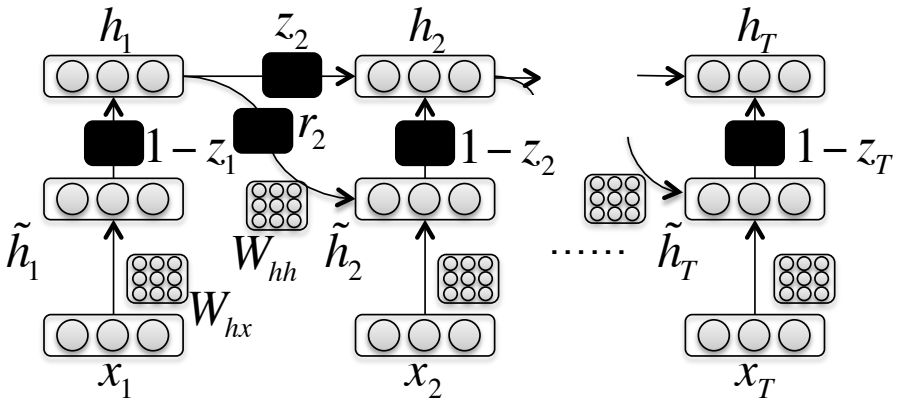


Figure 5.3: The GRU architecture.

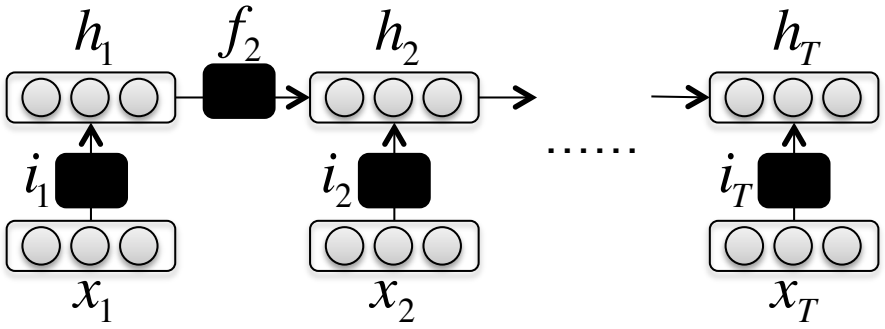


Figure 5.4: The GAC architecture.

for a given distributed representation x_t by the following equations:

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r), \tag{5.8}$$

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z), \tag{5.9}$$

$$\tilde{h}_t = g(W_{hx}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h), \tag{5.10}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t. \tag{5.11}$$

In these equations, $W_{rx}, W_{rh}, W_{zx}, W_{zh}, W_{hx}, W_{hh}$ are $d \times d$ matrices, $b_r, b_z, b_h \in \mathbb{R}^d$ are bias terms. We set $h_0 = 0$ at $t = 1$ in the same as RNN and LSTM. Some researchers reported that GRU is superior to LSTM [Chung et al., 2014] but we have no consensus about the superiority in phrase composition.

Pay attention back to additive composition. As mentioned above, additive composi-

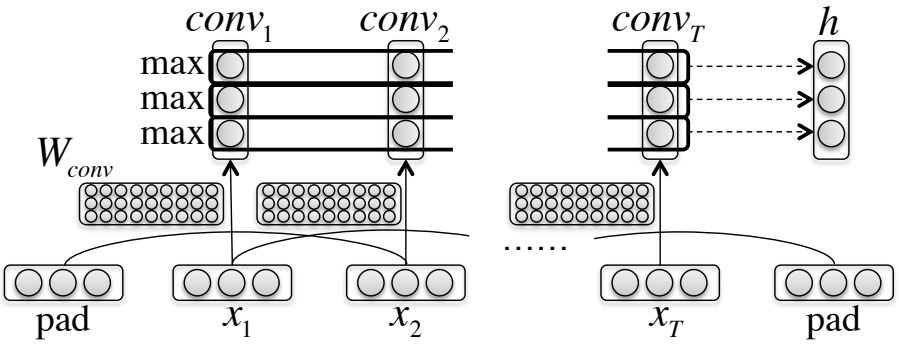


Figure 5.5: The CNN architecture.

tion can compute approximate distributed representations of phrases theoretically [Tian et al., 2016]. Therefore, it is an interesting issue to extend additive composition to be able to handle the order of word sequence and the importance of each constituent word. To address this issue, we propose Gated Additive Composition (GAC) that consists of the input gate $i_t \in \mathbb{R}^d$, forget gate $f_t \in \mathbb{R}^d$ and hidden state $h_t \in \mathbb{R}^d$:

$$i_t = \sigma(W_{ax}x_t + W_{ah}h_{t-1} + b_a), \tag{5.12}$$

$$f_t = \sigma(W_{bx}x_t + W_{bh}h_{t-1} + b_b), \tag{5.13}$$

$$h_t = g(f_t \odot h_{t-1} + i_t \odot x_t). \tag{5.14}$$

In these equations, W_{ax}, W_{ah}, W_{bx}, W_{bh} are $d \times d$ matrices, $b_a, b_b \in \mathbb{R}^d$ are bias terms. We set $h_0 = 0$ at $t = 1$ in the same as other methods. As shown in Figure 5.4, GAC employs not matrices but vectors (gates) when it computes the hidden state. Therefore, Equation 5.14 is interpreted as a weighted additive composition between the distributed representation x_t of the input word w_t and the previous hidden state h_{t-1}. In other words, GAC is interpreted as a weighted additive composition among constituent words. The input gate i_t and forget gate f_t control the elementwise weights. Essentially, the gate mechanism ignore unimportant words and forget meaningless contexts (scanned words).

5.1.4 Convolutional Neural Network

Convolutional Neural Networks (CNNs) [Lecun et al., 1998] are widely used to model the meanings of sentences [Kalchbrenner et al., 2014; Yin and Schütze, 2015; Yin et al., 2016]. In this study, we employ the basic CNN that computes the distributed representation of the phrase by the following equations:

$$\mathrm{conv_t} = \sigma(W_{conv}\left[x_{t-(n-1)/2}, ..., x_{t+(n-1)/2}\right] + b_{conv}), \tag{5.15}$$

$$\mathrm{pool} = \mathrm{maxpool}(\left[{}^{\mathrm{t}}\mathrm{conv}_1, ..., {}^{\mathrm{t}}\mathrm{conv}_{\mathrm{T}}\right]), \tag{5.16}$$

$$\mathrm{h} = g(\mathrm{pool}). \tag{5.17}$$

In these equations, W_{conv} is a $d\times(dn)$ matrix, $b_{conv} \in \mathbb{R}^d$ is a bias term, $\left[x_{t-(n-1)/2}, ..., x_{t+(n-1)/2}\right]$ means a concatenation of distributed representations of n words around w_t, n is a hyperparameter, and $\mathrm{maxpool}$ represents the max pooling operation that extracts a maximum value from each row of an input matrix. Figure 5.5 shows the overview of CNN when we define $n = 3$. If $t - (n-1)/2$ is smaller than 1, we use padding vectors as shown in Figure 5.5. In essence, CNN computes the distributed representation from each n word sequence in the target phrase. In other words, CNN pays attention to n-gram in the phrase.

2–5 are typically used as n. In this study, we define $n = 3$ because most of phrases in experiments consist of 2 or 3 words.

5.2 Experiments

We compared encoders explained in the previous section on several phrase composition datasets. We used two kinds of datasets. The one consists of only short phrases (bigram composition [Mitchell and Lapata, 2010], SemEval 2013 Task 5 [Korkontzelos et al., 2013]) and the other consists of arbitrary-length phrases (annotated PPDB [Wieting et al., 2015], the relational pattern dataset). To learn parameters of encoders, we adapted training procedures proposed in previous studies for each dataset. From a standpoint of the training procedure, we employed supervised learning for three datasets: bigram composition, SemEval 2013 Task 5, and annotated PPDB; on the other hand, we trained parameters with unlabeled data for relational pattern dataset.

In this section, we explain datasets and training procedures first, and then describe the comparison results.

5.2.1 Experimental Settings

5.2.1.1 Bigram Composition

Mitchell and Lapata [2010] produced the dataset containing semantic similarity judgments from 1 (dissimilar) to 7 (almost the same) by human annotators for each bigram pair. The dataset is composed of three phrase types: adjective noun (JN), compound noun (NN), and verb object (VN). Each phrase type contains 108 bigram pairs and each of pairs has 18 similarity judgments. For example, in compound noun, the bigram pair "phone call" and "committee meeting" is annotated 2, 5, 7, 2, and so on.

Wieting et al. [2015] suggested the similarity judgments are not consistent with the notion of paraphrase tasks. Therefore, they re-annotated semantic similarities to bigram pairs in the dataset. In this study, we exploited this refined dataset for encoder comparison.

In addition to the data construction, Wieting et al. [2015] applied neural network methods to phrase composition. In the bigram composition task, we adopted the same training procedure and training data as Wieting et al. [2015]. The training data[1] consists of 133,997 JN pairs, 35,601 NN pairs, and 62,640 VN pairs extracted from the XL portion of PPDB [Ganitkevitch et al., 2013].

We assume the training data is a set P of phrase pairs $\langle p_1, p_2 \rangle$. We trained parameters to make distributed representations of phrase pairs $\langle p_1, p_2 \rangle \in P$ to be similar to each other. To achieve the purpose, we minimized the following objective function by using AdaGrad [Duchi et al., 2011] with mini-batches:

$$\begin{aligned} \sum_{\langle p_1, p_2 \rangle \in P} & \max(0, \delta - h_{p_1} \cdot h_{p_2} + h_{p_1} \cdot h_{\overline{p_2}}) \\ & + \max(0, \delta - h_{p_1} \cdot h_{p_2} + h_{\overline{p_1}} \cdot h_{p_2}) \\ & + \lambda_\theta ||\theta||^2 + \lambda_X ||X_{initial} - X||^2. \end{aligned} \tag{5.18}$$

In this equation, λ_θ and λ_X represent the importance of regularization terms, θ is pa-

[1] ttic.uchicago.edu/~wieting/

rameters of encoders[1], X is the matrix that consists of distributed representations of all words in vocabulary, $X_{initial}$ is the initial values of X, h_p is the distributed representation of the phrase p computed by each encoder such as RNN, δ is the margin, and $\overline{p}$ is a sampled negative example taken from a mini-batch. In short, this objective function requires distributed representations of given phrase pairs to be more similar to each other than respective negative examples by a margin of at least δ.

We chose the most similar phrase in the mini-batch excluding the given phrase pair as the negative example. For example, we selected $\overline{p_2}$ for the phrase pair $\langle p_1, p_2 \rangle$:

$$\overline{p_2} = \text{argmax}_{\langle p_1, \overline{p_2} \rangle \in P_b \setminus \langle p_1, p_2 \rangle} h_{p_1} \cdot h_{\overline{p_2}},$$

where $P_b \subseteq P$ is the current mini-batch.

We initialized distributed representations of words with publicly available paragram vectors[2] [Wieting et al., 2015]. The number of dimensions of the vectors is 25. In this experiment, we update distributed representations of words besides matrices contained in each encoder.

For hyperparameters, we set the initial learning rate to 0.5 for distributed representations of words and 0.05 for the encoder parameters, the number of epoch to 5, and the margin to 1. We conducted a coarse grid search for the others: λ_θ, λ_X, and mini-batch size. We searched λ_θ in $\{10, 1, 10^{-1}, ..., 10^{-6}\}$, λ_X in $\{10^{-1}, 10^{-2}, 10^{-3}, 0\}$, and mini-batch size in $\{100, 250, 500, 1000, 2000\}$. We tuned these hyperparameters using the dataset produced by Mitchell and Lapata [2010] as development set.

5.2.1.2 SemEval 2013 Task 5

SemEval 2013 Task 5 [Korkontzelos et al., 2013] is the task to compute semantic similarity between words and bigrams. Concretely, the task is a binary classification task to judge whether an input pair of a bigram and a word is a paraphrase or not. For example, we should be able to find out that 'bag' and "flexible container" are similar whereas 'zoning' and "extended session" are not. The dataset consists of 11,722 training instances and 7,814 test instances.

[1] We exclude distributed representations of words from encoder parameters. In other words, we consider matrices belonging to each encoder as encoder parameters.

[2] `http://ttic.uchicago.edu/~wieting/`

We classified each instance into similar or dissimilar by computing dot product between the distributed representation of the word and the one of the phrase. More specifically, we used the dot product as the feature for a linear classifier. To learn parameters, we minimized the following objective function J with Adam [Kingma and Ba, 2015]:

$$q = \sigma(\alpha(h_{p_n} \cdot x_n)), \tag{5.19}$$

$$J = - \sum_{\langle p_n, w_n, y_n \rangle \in P} y_n \log(q) + (1 - y_n) \log(1 - q). \tag{5.20}$$

In these equations, α is a weight scalar, P is a set of training instances, and each instance consists of a bigram p_n, a word w_n, and a label representing the pair is a paraphrase $y_n = 1$ or not $y_n = 0$. Essentially, equation 5.19 represents the probability that the bigram p_n is a paraphrase of the word w_n. That is, equation 5.20 is the sigmoid cross entropy.

SemEval 2013 Task 5 dataset does not contain development set. Therefore, to tune the number of epoch for each method, we divided training data in the ratio of 1 to 9 and used the small part as development set. We chose the number of epoch with the development set, and then we trained encoders on whole training data with the selected epoch number.

To obtain initial values, we applied word2vec[1] to ukWaC[2]. For the number of dimensions, we employed the same value as [Yu and Dredze, 2015]: $d = 200$. We initialized the distributed representations of words with the word2vec result and updated the distributed representations during training.

5.2.1.3 Annotated PPDB

Wieting et al. [2015] created the annotated PPDB that contains arbitrary-length phrase pairs with semantic similarity ratings from 1 (dissimilar) to 5 (very similar). They extracted phrase pairs from PPDB, and asked workers in Amazon Mechanical Turk to judge semantic similarity of a given phrase pair. After annotation, they divided the 1,260 annotated phrase pairs into development set (260 pairs) and test set (1,000 pairs).

[1] `code.google.com/archive/p/word2vec/`
[2] `wacky.sslmit.unibo.it/doku.php?id=corpora`

Wieting et al. [2015] also provided training data to model phrase composition. They extracted phrase pairs from PPDB, and filtered out pairs that may not have paraphrase relations. Finally, the training data contains 60,000 phrase pairs.

We employed the same training procedure as Wieting et al. [2015]: used training data described above, initialized distributed representations of words with publicly available paragram vectors, conducted a coarse grid search for hyperparameters in the same space in Section 5.2.1.1, and adopted the objective function explained in Section 5.2.1.1 with AdaGrad. In addition to parameters of encoders, we updated distributed representations of words.

5.2.1.4 Relational Pattern

We call a word sequence appearing between an entity pair by a relational pattern. For example, in the sentence "Tobacco increases the risk of cancer", we regard the phrase "increases the risk of" as a relational pattern. In Chapter 4, we provide relational pattern pairs with semantic similarity ratings from 1 (dissimilar) to 7 (very similar). We focus on the existing dataset that contains annotations whether a relational pattern pair has an entailment relation or not [Zeichner et al., 2012]. This thesis re-annotates semantic similarity judgments on every relational pattern pair by using a crowdsourcing service.

For training, we apply Skip-gram with negative sampling [Mikolov et al., 2013b] to the ukWaC corpus. More specifically, we define the log-likelihood of the relational pattern p as follows:

$$\sum_{\tau \in S_p} \left(\log \sigma(h_p^\top \tilde{x}_\tau) + \sum_{k=1}^{K} \log \sigma(-h_p^\top \tilde{x}_{\check{\tau}}) \right), \tag{5.21}$$

where S_p is a set of L words surrounding the relational pattern p, K is the number of negative samples, $x_{\check{\tau}}$ is the distributed representation of the negative sampled word $w_{\check{\tau}}$. We maximized the log-likelihood with SGD through all relational patterns extracted from the ukWaC corpus by Reverb [Fader et al., 2011].

For hyperparameters, we defined the window size $L = 5$, the number of negative samples $K = 20$, the subsampling of 10^{-5}, the number of dimensions of distributed representations $d = 300$. We initialized distributed representations of words by the

Method	Bigram Composition			SemEval 2013	Annotated
	JN	NN	VN	Task 5	PPDB
Add (init)	0.50	0.29	0.58	77.3 ± 0.1	0.32
Add	**0.62**	0.40	**0.59**	79.0 ± 0.2	**0.47**
RNN	0.55 ± 0.02	0.33 ± 0.05	0.43 ± 0.02	71.9 ± 0.7	0.23 ± 0.02
LSTM	0.58 ± 0.01	0.33 ± 0.04	0.46 ± 0.02	77.1 ± 0.4	0.30 ± 0.01
GRU	0.59 ± 0.02	$\mathbf{0.36 \pm 0.08}$	0.55 ± 0.02	78.1 ± 0.4	0.33 ± 0.01
GAC	0.56 ± 0.00	$\mathbf{0.40 \pm 0.04}$	0.54 ± 0.01	$\mathbf{82.2 \pm 0.3}$	$\mathbf{0.46 \pm 0.01}$
CNN	$\mathbf{0.60 \pm 0.03}$	0.35 ± 0.08	0.43 ± 0.01	78.0 ± 0.4	0.43 ± 0.01
Recursive NN [Wieting et al., 2015]	0.57	**0.44**	0.55	–	0.40
Hashimoto et al. [2014]	0.38	0.39	0.45	–	–
FCT-J [Yu and Dredze, 2015]	–	–	–	70.7	–
HsH [Wartena, 2013]	–	–	–	80.3	–

Table 5.1: Results on each dataset. The columns of bigram composition and annotated PPDB show Spearman's rank correlation. The column of SemEval 2013 Task 5 shows accuracy. Add (init) represents the performance of additive composition with initial distributed representations. The highest score in each column is represented by bold.

result of word2vec, and updated the distributed representations during training. In this training, we also used single words (words not included in a set of relational patterns) to update distributed representations.

5.2.2 Results

The first group of Table 5.1 shows the results of encoders trained for each dataset except for relational patterns[1]. For bigram composition and annotated PPDB, we computed cosine similarity between distributed representations obtained by each encoder, and then we evaluate Spearman's rank correlation ρ between the calculated cosine similarities and similarity judgments in each dataset. For SemEval 2013 Task 5, we computed the accuracy on test set. The first group of Table 5.1 indicates the means and standard deviations of 5 models trained from different initial values with the best hyperparameters for each encoder[2]. Table 5.1 also shows the results reported in previous researches [Wartena, 2013; Wieting et al., 2015; Yu and Dredze, 2015]. In particular,

[1]We report the result on the relational pattern dataset latter.

[2]In addition to parameters of encoders, we used different distributed representations of words except for the case in which we used publicly available paragram vectors. Therefore, we describes the mean and standard deviation of Add (init) on SemEVal 2013 Task 5 and relational pattern dataset.

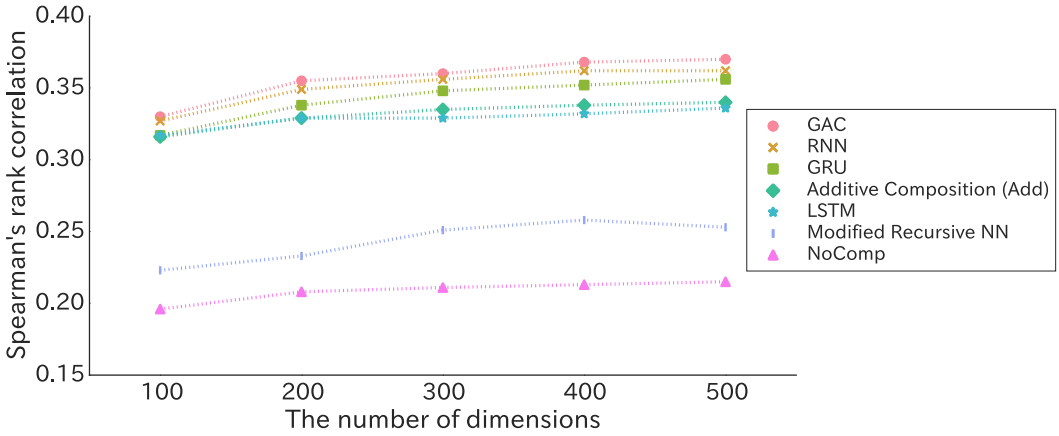


Figure 5.6: Performance of each method on the relational pattern similarity task with variation in the number of dimensions.

Recursive NN indicates the performance of Recursive Neural Network trained from the same initial values with the same manner in this study.

In bigram composition, additive composition (Add) achieves the best performance in JN and VN. For NN, Recursive NN achieves the best score, on the other hand, we can consider that GAC and GRU are comparable to Recursive NN when we take the standard deviation into account. In fact, the one of GAC models wins Recursive NN because it achieves $\rho = 0.47$ on NN.

For VN, encoders except for additive composition lower the Spearman's ρ from the initial distributed representations. In other words, most encoders fail to acquire good composition functions (parameters) in training. We suppose that the training data contains a lot of incorrect instances because the training data is generated from PPDB automatically.

In SemEval 2013 Task 5, GAC achieves the highest accuracy among encoders. Moreover, GAC outperforms HsH that obtains semantic vectors from ukWaC corpus (the same corpus in this study). This result means that GAC achieves the state-of-the-art accuracy because HsH is the best method in SemEval 2013 Task 5 to date. On the other hand, all neural encoders are defeated against trained additive composition except for GAC. We suppose they suffer from overfitting because the losses computed by them are lower than the one by additive composition during training.

In annotated PPDB, additive composition achieves the best performance among en-

coders. Moreover, GAC is comparable to additive composition because three of GAC models achieve the same ρ as additive composition. Surprisingly, Table 5.1 indicates that the Spearman's ρ of GRU and LSTM are almost the same as additive composition with initial values even though both of them are successful in several NLP tasks. By comparing to Recursive NN, we suppose that additive composition and GAC are superior encoders in bigram composition and annotated PPDB because they achieve comparable (or better) performance to Recursive Neural Network without additional information i.e., a syntactic tree.

Figure 5.6 shows Spearman's rank correlations of different encoders when the number of dimensions of vectors is 100–500 on the constructed relational pattern dataset. The figure shows that GAC achieves the best performance on all dimensions.

Figure 5.6 includes the performance of the naïve approach, "NoComp", which regards a relational pattern as a single unit (word). In this approach, we allocated a vector h_p for each relational pattern p in Equation 5.21 instead of the vector composition, and trained the vectors of relational patterns using the Skip-gram model. The performance was poor for two reasons: we were unable to compute similarity values for 1,744 pairs because relational patterns in these pairs do not appear in ukWaC; and relational patterns could not obtain sufficient statistics because of data sparseness.

Figure 5.6 also includes the result of the modified recursive neural network proposed in Chapter 3. We used $\lambda = 10^6$ for l_1 regularization because the value achieved the highest scores on the experiments in Chapter 3. In Chapter 3, we used only content words for phrase composition but used all words contained in the given relational pattern in this experiment in order for a fair comparison. Figure 5.6 shows that the modified recursive neural network is defeated additive composition even though the modified recursive neural network outperforms additive composition on the experiments in Chapter 3. We consider three reasons. First, the evaluation data and the indicator for evaluation are different from experiments in Chapter 3. Thus, naturally, the performance of each method is different from the one in Chapter 3. Second, we used more vectors and matrices than the experiments in Chapter 3 because we used all words in the given relational pattern as explained above. Third, since the modified recursive neural network has parameters more than 200 times as many as the simple RNN, it is difficult to train the modified recursive neural network.

Figure 5.6 shows that neural encoders outperform additive composition, except for

Length	#	NoComp	Add	LSTM	GRU	RNN	GAC
1	636	**0.324**	**0.324**	**0.324**	**0.324**	**0.324**	**0.324**
2	1,018	0.215	0.319	0.257	0.274	0.285	**0.321**
3	2,272	0.234	0.386	0.344	0.370	0.387	**0.404**
4	1,206	0.208	0.306	0.314	**0.329**	0.319	0.323
> 5	423	0.278	0.315	0.369	0.384	**0.394**	0.357
All	5,555	0.215	0.340	0.336	0.356	0.362	**0.370**

Table 5.2: Spearman's rank correlations on different pattern lengths (number of dimensions $d = 500$).

the modified recursive neural network and LSTM. This result suggests that neural encoders can reach better composition function than additive composition. However, we consider that neural encoders except for GAC are easy to be overfitting because Table 5.1 indicates that most encoders are defeated against additive composition. Moreover, these results show that additive composition is stronger than other encoders except for relational pattern dataset even though it is much simpler than others.

5.2.2.1 Discussions

Table 5.2 reports Spearman's rank correlations computed for each pattern length. Here, the length of a relational-pattern pair is defined by the maximum of the lengths of two patterns in the pair. In length of 1, all methods achieve the same correlation score because they use the same word vector x_t. The table shows that additive composition (Add) performs well for shorter relational patterns (lengths of 2 and 3) but poorly for longer ones (lengths of 4 and 5+). GAC also exhibits the similar tendency to Add, but it outperforms Add for shorter patterns (lengths of 2 and 3) probably because of the adaptive control of input and forget gates. In contrast, RNN and its variants (RNN, GRU, and LSTM) enjoy the advantage on longer patterns (lengths of 4 and 5+).

To examine the roles of input and forget gates of GAC, we visualize the moments when input/forget gates are wide open or closed. More precisely, we extract the input word and scanned words when $|i_t|_2$ or $|f_t|_2$ is small (close to zero) or large (close to one) on the relational-pattern dataset. We restate that we compose a pattern vector in backward order (from the last to the first): GAC scans 'of', 'author', and 'be' in this order for composing the vector of the relational pattern 'be author of'.

	w_t	w_{t+1} w_{t+2} ...
large i_t (input open)	reimburse payable liable	for in to
small i_t (input close)	a a be	charter member of valuable member of an avid reader of
large f_t (forget open)	be be be	eligible to participate in require to submit request to submit
small f_t (forget close)	coauthor capital center	of of of

Table 5.3: Prominent moments for input/forget gates.

Table 5.3 displays the top three examples identified using the procedure. The table shows two groups of tendencies. Input gates open and forget gates close when scanned words are only a preposition and the current word is a content word. In these situations, GAC tries to read the semantic vector of the content word and to ignore the semantic vector of the preposition. In contrast, input gates close and forget gates open when the current word is 'be' or 'a' and scanned words form a noun phrase (e.g., "charter member of"), a complement (e.g., "eligible to participate in"), or a passive voice (e.g., "require(d) to submit"). This behavior is also reasonable because GAC emphasizes informative words more than functional words.

5.2.3 Relation classification

5.2.3.1 Experimental settings

To examine the usefulness of the dataset and distributed representations for a different application, we address the task of relation classification on the SemEval 2010 Task 8 dataset [Hendrickx et al., 2010]. In other words, we explore whether high-quality distributed representations of relational patterns are effective to identify a relation type of an entity pair.

The dataset consists of $10,717$ relation instances ($8,000$ training and $2,717$ test instances) with their relation types annotated. The dataset defines 9 directed relations

Method	Feature set	F1
SVM (Linear kernel) + NoComp	embeddings	55.2
SVM (Linear kernel) + LSTM	embeddings	73.7
SVM (Linear kernel) + Add	embeddings	71.9
SVM (Linear kernel) + GRU	embeddings	74.8
SVM (Linear kernel) + RNN	embeddings	73.8
SVM (Linear kernel) + GAC	embeddings	75.2
SVM (RBF kernel)	BoW, POS	77.3
SVM (RBF kernel) + NoComp	embeddings, BoW, POS	79.9
SVM (RBF kernel) + LSTM	embeddings, BoW, POS	81.1
SVM (RBF kernel) + Add	embeddings, BoW, POS	81.1
SVM (RBF kernel) + GRU	embeddings, BoW, POS	81.4
SVM (RBF kernel) + RNN	embeddings, BoW, POS	81.7
SVM (RBF kernel) + GAC	embeddings, BoW, POS	82.0
	+ dependency, WordNet, NE	83.7
Ranking loss + GAC w/ fine-tuning	embeddings, BoW, POS	
	+ dependency, WordNet, NE	84.2
SVM [Rink and Harabagiu, 2010]	BoW, POS, dependency, Google n-gram, etc.	82.2
MV-RNN [Socher et al., 2012]	embeddings, parse trees	79.1
	+ WordNet, POS, NE	82.4
FCM [Gormley et al., 2015] w/o fine-tuning	embeddings, dependency	79.4
	+ WordNet	82.0
w/ fine-tuning	embeddings, dependency	82.2
	+ NE	83.4
RelEmb [Hashimoto et al., 2015]	embeddings	82.8
	+ dependency, WordNet, NE	83.5
CR-CNN [dos Santos et al., 2015] w/ Other	embeddings, word position embeddings	82.7
w/o Other	embeddings, word position embeddings	84.1
depLCNN [Xu et al., 2015]	embeddings, dependency	81.9
	+ WordNet	83.7
depLCNN + NS	embeddings, dependency	84.0
	+ WordNet	85.6

Table 5.4: F1 scores on the SemEval 2010 dataset.

(e.g.,CAUSE-EFFECT) and 1 undirected relation OTHER. Given a pair of entity mentions, the task is to identify a relation type in 19 candidate labels (2×9 directed $+$ 1 undirected relations). For example, given the pair of entity mentions $e_1 =$ 'burst' and $e_2 =$ 'pressure' in the sentence "The *burst* has been caused by water hammer *pressure*", a system is expected to predict CAUSE-EFFECT(e_2, e_1).

We used Support Vector Machines (SVM) with a linear kernel and a Radial Basis Function (RBF) kernel implemented in LIBSVM[1]. Concretely, we use distributed rep-

[1]https://www.csie.ntu.edu.tw/~cjlin/libsvm/

resentations of a relational pattern, entities, and a word before and after the entity pair (number of dimensions $d = 500$) as features to train the SVM with a linear kernel. In this task, we regard words appearing between an entity pair as a relational pattern. We compare the vector representations of relational patterns computed by NoComp and the five encoders presented in Section 5.2.2.1: additive composition, RNN, GRU, LSTM, and GAC. Additionally, we incorporate lexical features: part-of-speech tags (predicted by TreeTagger), surface forms, lemmas of words appearing between an entity pair, and lemmas of the words in the entity pair as features when we train the SVM with a RBF kernel. Hyper-parameters related to SVM were tuned by 5-fold cross validation on the training data.

5.2.3.2 Results and discussions

Table 5.4 presents the macro-averaged F1 scores on the SemEval 2010 Task 8 dataset. The first group of the table shows the results of using distributed representations as features for the SVM with a linear kernel. These results indicate that the model achieving high score on the relational pattern dataset such as RNN, GRU, and GAC also achieve high score on the relation classification task. Especially, GAC that achieves the best performance on the relational pattern dataset achieves the best score on the first group of the table.

The second group of the table provides the result of lexical features and enhancements with the distributed representations. We can observe a significant improvement even from the distributed representation of NoComp (77.3 to 79.9). Moreover, the distributed representation that exhibited the high performance on the pattern similarity task was also successful on this task; GAC, which yielded the highest performance on the pattern similarity task, also achieved the best performance (82.0) of all encoders on this task.

It is noteworthy that the improvements brought by the different encoders on this task roughly correspond to the performance on the pattern similarity task. This fact implies two potential impacts. First, the distributed representations of relational patterns are useful and easily transferable to other tasks such as knowledge base population. Second, the pattern similarity dataset provides a gauge to predict successes of distributed representations in another task.

We could further improve the performance of SVM + GAC by incorporating external resources in the similar manner as the previous studies did. Concretely, SVM + GAC achieved 83.7 F1 score by adding features for WordNet, named entities (NE), and dependency paths explained in Hashimoto et al. [2015]. Moreover, we could obtain 84.2 F1 score, using the ranking based loss function [dos Santos et al., 2015] and fine-tuning of the distributed representations initially trained by GAC. Currently, this is the second best score among the performance values reported in the previous studies on this task (the second group of Table 5.4). If we could use the negative sampling technique proposed by Xu et al. [2015], we might improve the performance further[1].

5.3 Related Work

Mitchell and Lapata [2010] is a pioneer work in modeling the meanings of phrases. They introduced the bigram composition dataset that consists of bigram pairs with human annotated semantic similarity. Moreover, they compared several composition functions including additive composition on the constructed dataset. After this study, several researchers tackled to find the most suitable method to compose vector representations. Muraoka et al. [2014] compared additive composition, Recursive Neural Network, and its variants on the bigram composition dataset. Takase et al. [2016b] provided the dataset that consists of relational pattern pairs with semantic similarity judged by crowdsourcing workers and compared additive composition, RNN, GRU, LSTM, and GAC on the constructed dataset. These studies revealed pros/cons of each method empirically but they restricted a target linguistic unit. Hill et al. [2016a] investigated the performance of several neural methods such as Skip-thought [Kiros et al., 2015] on embedding sentences into distributed representations. Although they exploited various tasks for comparison, settings (e.g., an objective function) in training encoders are different from each other. In other words, it is unclear what factors contribute to phrase composition in their comparison.

Mikolov et al. [2013b] suggested that additive composition of distributed representations is useful to model the meanings of phrases. Muraoka et al. [2014] demonstrated

[1] In fact, we made substantial efforts to introduce the negative sampling technique. However, Xu et al. [2015] omits the detail of the technique probably because of the severe page limit of short papers. For this reason, we could not reproduce their method in this study.

that additive composition is stronger than the vanilla Recursive Neural Network and most of its variants. In addition to these empirical studies, for bigram, Tian et al. [2016] proved that additive composition can provide an ideal distributed representation acquired from an infinite amount of corpus.

To compose distributed representations of arbitrary-length phrases, Pham et al. [2015] and Kenter et al. [2016] employed additive composition. The difference of these studies is a training objective. Pham et al. [2015] applied Skip-gram objective [Mikolov et al., 2013b] to be able to predict surrounding words from a target phrase. Kenter et al. [2016] train distributed representations to be able to surrounding sentences from the encoded distributed representation of a target sentence. Yu and Dredze [2015] adopted weighted additive composition to model the meanings of phrases. They proposed the method to tune the weight vector for each word by using linguistic features such as POS tags. Besides additive composition, previous studies adapted neural encoders including Recursive Neural Network [Socher et al., 2011a], LSTM [Hill et al., 2016b], GRU [Kiros et al., 2015], and GAC [Takase et al., 2016b] to encode arbitrary-length phrases. In this study, we does not propose a novel composition function or objective function but conduct a fair comparison among those encoders.

Nakashole et al. [2012] approached the similar task by constructing a taxonomy of relational patterns. They represented a vector of a relational pattern as the distribution of entity pairs co-occurring with the relational pattern. Grycner et al. [2015] extended Nakashole et al. [2012] to generalize dimensions of the vector space (entity pairs) by incorporating hyponymy relation between entities. They also used external resources to recognize the transitivity of pattern pairs and applied transitivities to find patterns in entailment relation. These studies did not consider semantic composition of relational patterns. Thus, they might suffer from the data sparseness problem, as shown by NoComp in Figure 5.6.

Numerous studies have been aimed at encoding distributed representations of phrases and sentences from word embeddings by using: Recursive Neural Network [Socher et al., 2011b], Matrix Vector Recursive Neural Network [Socher et al., 2012], Recursive Neural Network with different weight matrices corresponding to syntactic categories [Socher et al., 2013] or word types [Takase et al., 2016a], RNN [Sutskever et al., 2011], LSTM [Sutskever et al., 2014], GRU [Cho et al., 2014], PAS-CLBLM [Hashimoto

et al., 2014], etc. As described in Section 5.1, we applied RNN, GRU, and LSTM to compute distributed representations of relational patterns because recent papers have demonstrated their superiority in semantic composition [Sutskever et al., 2014; Tang et al., 2015]. In this paper, we presented a comparative study of different encoders for semantic modeling of relational patterns.

To investigate usefulness of the distributed representations and the new dataset, we adopted the relation classification task (SemEval 2010 Task 8) as a real application. On the SemEval 2010 Task 8, several studies considered semantic composition. Gormley et al. [2015] proposed Feature-rich Compositional Embedding Model (FCM) that can combine binary features (e.g., positional indicators) with word embeddings via outer products. dos Santos et al. [2015] addressed the task using Convolutional Neural Network (CNN). Xu et al. [2015] achieved a higher performance than dos Santos et al. [2015] by application of CNN on dependency paths.

In addition to the relation classification task, we briefly describe other applications. To populate a knowledge base, Riedel et al. [2013] jointly learned latent feature vectors of entities, relational patterns, and relation types in the knowledge base. Toutanova et al. [2015] adapted CNN to capture the compositional structure of a relational pattern during the joint learning. For open domain question answering, Yih et al. [2014] proposed the method to map an interrogative sentence on an entity and a relation type contained in a knowledge base by using CNN.

Although these reports described good performance on the respective tasks, we are unsure of the generality of distributed representations trained for a specific task such as the relation classification. In contrast, this paper demonstrated the contribution of distributed representations trained in a generic manner (with the Skip-gram objective) to the task of relation classification.

5.4 Conclusion

In this chapter, we compared additive composition and commonly used neural encoders on phrase composition. The comparison suggests that GAC is superior to other encoders because it achieves high performance and more robust than others. Moreover, GAC outperforms the state-of-the-art score in SemEval 2013 Task 5. Furthermore, the experimental results indicate that additive composition is strong encoder.

We also investigate the usefulness of constructed relational pattern dataset. Experimental results demonstrate that the presented dataset is useful to predict successes of the distributed representations in the relation classification task. Moreover, experiments indicate that distributed representations computed by neural encoders contribute to downstream NLP applications such as the relation classification task.

Chapter 6

Encoding Semantic and Syntactic Features

Neural network-based encoder-decoder models are cutting-edge methodologies for tackling natural language generation (NLG) tasks, *i.e.*, machine translation [Cho et al., 2014], image captioning [Vinyals et al., 2015], video description [Venugopalan et al., 2015], and headline generation [Rush et al., 2015] because neural encoders can embed syntactic and semantic information into distributed representations implicitly. We consider structural, syntactic, and semantic information underlying input text such as POS tagging, dependency parsing, named entity recognition, and semantic role labeling has the potential for improving the quality of NLG tasks. However, to the best of our knowledge, there is no clear evidence that syntactic and semantic information can enhance the recently developed encoder-decoder models in NLG tasks.

To answer this research question, this chapter proposes and evaluates a headline generation method based on an encoder-decoder architecture on *Abstract Meaning Representation (AMR)*. The method is essentially an extension of *attention-based summarization (ABS)* [Rush et al., 2015]. Our proposed method encodes results obtained from an AMR parser by using a modified version of Tree-LSTM encoder [Tai et al., 2015] as additional information of the baseline ABS model. Conceptually, the reason for using AMR for headline generation is that information presented in AMR, such as predicate-argument structures and named entities, can be effective clues when producing shorter summaries (headlines) from original longer sentences. We expect that the

quality of headlines will improve with this reasonable combination (ABS and AMR).

6.1 Attention-based summarization (ABS)

ABS proposed in Rush et al. [2015] has achieved state-of-the-art performance on the benchmark data of headline generation including the DUC-2004 dataset [Over et al., 2007]. Figure 6.1 illustrates the model structure of ABS. The model predicts a word sequence (summary) based on the combination of the neural network language model and an input sentence encoder.

Let V be a vocabulary. x_i is the i-th indicator vector corresponding to the i-th word in the input sentence. Suppose we have M words of an input sentence. X represents an input sentence, which is represented as a sequence of indicator vectors, whose length is M. That is, $x_i \in \{0,1\}^{|V|}$, and $X = (x_1, \ldots, x_M)$. Similarly, let Y represent a sequence of indicator vectors $Y = (y_1, \ldots, y_L)$, whose length is L. Here, we assume $L < M$. $Y_{C,i}$ is a short notation of the list of vectors, which consists of the sub-sequence in Y from y_{i-C+1} to y_i. We assume a one-hot vector for a special start symbol, such as "⟨S⟩", when $i < 1$. Then, ABS outputs a summary $\hat{Y}$ given an input sentence X as follows:

$$\hat{Y} = \arg\max_{Y} \Big\{ \log p(Y|X) \Big\}, \tag{6.1}$$

$$\log p(Y|X) \approx \sum_{i=0}^{L-1} \log p(y_{i+1}|X, Y_{C,i}), \tag{6.2}$$

$$p(y_{i+1}|X, Y_{C,i}) \propto \exp\big(\mathrm{nnlm}(Y_{C,i}) + \mathrm{enc}(X, Y_{C,i})\big), \tag{6.3}$$

where $\mathrm{nnlm}(Y_{C,i})$ is a feed-forward neural network language model proposed in [Bengio et al., 2003], and $\mathrm{enc}(X, Y_{C,i})$ is an input sentence encoder with attention mechanism.

This paper uses D and H as denoting sizes (dimensions) of vectors for word embedding and hidden layer, respectively. Let $E \in \mathbb{R}^{D \times |V|}$ be an embedding matrix of output words. Moreover, let $U \in \mathbb{R}^{H \times (CD)}$ and $O \in \mathbb{R}^{|V| \times H}$ be weight matrices

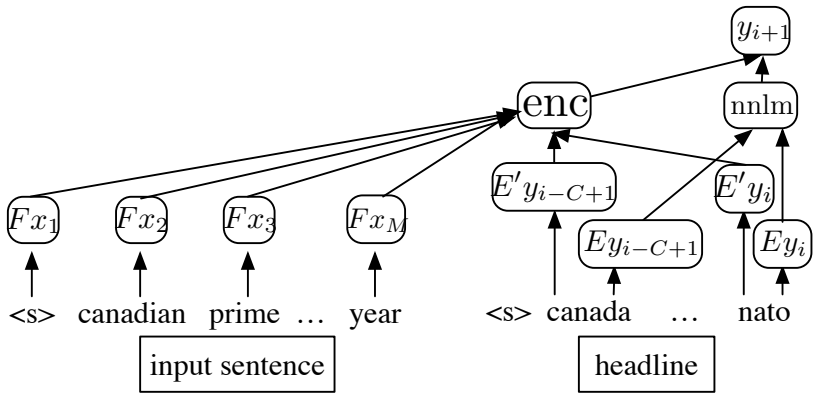


Figure 6.1: Model structure of 'attention-based summarization (ABS)'.

of hidden and output layers, respectively[1]. Using the above notations, $\mathrm{nnlm}(Y_{C,i})$ in Equation 6.3 can be written as follows:

$$\mathrm{nnlm}(Y_{C,i}) = Oh, \quad h = \tanh(U\tilde{y}_c), \tag{6.4}$$

where $\tilde{y}_c$ is a concatenation of output embedding vectors from $i - C + 1$ to i, that is, $\tilde{y}_c = (Ey_{i-C+1} \cdots Ey_i)$. Therefore, $\tilde{y}_c$ is a (CD) dimensional vector.

Next, $F \in \mathbb{R}^{D\times|V|}$ and $E' \in \mathbb{R}^{D\times|V|}$ denote embedding matrices of input and output words, respectively. $O' \in \mathbb{R}^{|V|\times D}$ is a weight matrix for the output layer. $P \in \mathbb{R}^{D\times(CD)}$ is a weight matrix for mapping embedding of C output words onto embedding of input words. $\tilde{X}$ is a matrix form of a list of input embeddings, namely, $\tilde{X} = \left[\tilde{x}_1, \ldots, \tilde{x}_M\right]$, where $\tilde{x}_i = Fx_i$. Then, $\mathrm{enc}(X, Y_{C,i})$ is defined as the following equations:

$$\mathrm{enc}(X, Y_{C,i}) = O'\bar{X}p, \tag{6.5}$$

$$p \propto \exp(\tilde{X}^{\mathrm{T}} P\tilde{y}'_c), \tag{6.6}$$

where $\tilde{y}'_c$ is a concatenation of output embedding vectors from $i - C + 1$ to i similar to $\tilde{y}_c$, that is, $\tilde{y}'_c = (E'y_{i-C+1} \cdots E'y_i)$. Moreover, $\bar{X}$ is a matrix form of a list of

[1]Following Rush et al. [2015], we omit bias terms throughout the paper for readability, though each weight matrix also has a bias term.

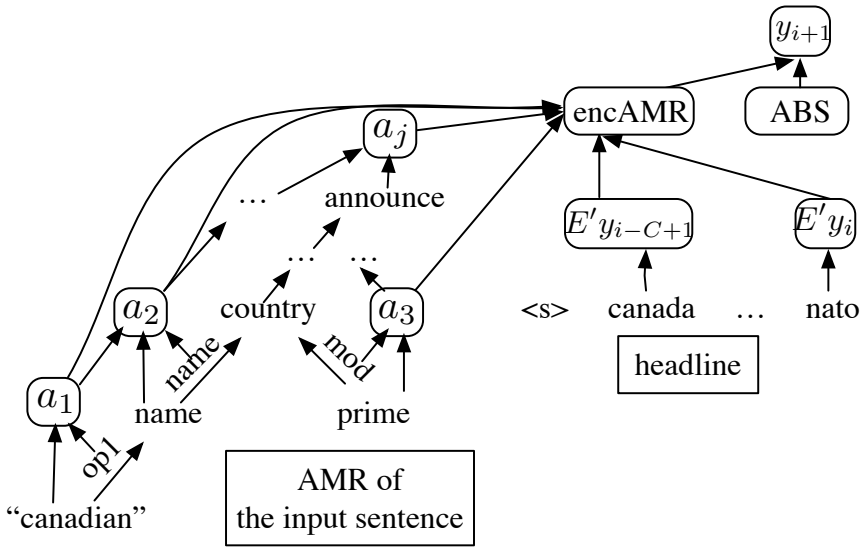


Figure 6.2: Model structure of our proposed attention-based AMR encoder; it outputs a headline using ABS and encoded AMR with attention.

averaged input word embeddings within window size Q, namely, $\bar{X} = [\bar{x}_1, \ldots, \bar{x}_M]$, where $\bar{x}_i = \sum_{q=i-Q}^{i+Q} \frac{1}{Q} \tilde{x}_q$.

Equation 6.6 is generally referred to as the attention model, which is introduced to encode a relationship between input words and the previous C output words. For example, if the previous C output words are assumed to align to x_i, then the surrounding Q words $(x_{i-Q}, \ldots, x_{i+Q})$ are highly weighted by Equation 6.5.

6.2 Proposed Method

Our assumption here is that syntactic and semantic features of an input sentence can greatly help for generating a headline. For example, the meanings of subjects, predicates, and objects in a generated headline should correspond to the ones appearing in an input sentence. Thus, we incorporate syntactic and semantic features into the framework of headline generation. This paper uses an AMR as a case study of the additional features.

6.2.1 Abstract Meaning Representation (AMR)

An Abstract Meaning Representation (AMR) is a rooted, directed, acyclic graph that encodes the meaning of a sentence. Nodes in an AMR graph represent 'concepts', and directed edges represent a relationship between nodes. Concepts consist of English words, PropBank event predicates, and special labels such as "person". For edges, AMR has approximately 100 relations [Banarescu et al., 2013] including semantic roles based on the PropBank annotations in OntoNotes [Hovy et al., 2006]. To acquire AMRs for input sentences, we use the state-of-the-art transition-based AMR parser [Wang et al., 2015].

6.2.2 Attention-Based AMR Encoder

Figure 6.2 shows a brief sketch of the model structure of our attention-based AMR encoder model. We utilize a variant of child-sum Tree-LSTM originally proposed in [Tai et al., 2015] to encode syntactic and semantic information obtained from output of the AMR parser into certain fixed-length embedding vectors. To simplify the computation, we transform a DAG structure of AMR parser output to a tree structure, which we refer to as "*tree-converted AMR structure*". This transformation can be performed by separating multiple head nodes, which often appear for representing coreferential concepts, to a corresponding number of out-edges to head nodes. Then, we straightforwardly modify Tree-LSTM to also encode edge labels since AMR provides both node and edge labels, and original Tree-LSTM only encodes node labels.

Let n_j and e_j be N and E dimensional embeddings for labels assigned to the j-th node, and the out-edge directed to its parent node[1]. W_{in}, W_{fn}, W_{on}, $W_{un} \in \mathbb{R}^{D \times N}$ are weight matrices for node embeddings n_j[2]. Similarly, W_{ie}, W_{fe}, W_{oe}, $W_{ue} \in \mathbb{R}^{D \times E}$ are weight matrices for edge embeddings e_j. W_{ih}, W_{fh}, W_{oh}, $W_{uh} \in \mathbb{R}^{D \times D}$ are weight matrices for output vectors connected from child nodes. $B(j)$ represents a set of nodes, which have a direct edge to the j-th node in our tree-converted AMR structure. Then, we define embedding a_j obtained at node j in tree-converted AMR structure via Tree-

[1] We prepare a special edge embedding for a root node.
[2] As with Equation 6.4, all the bias terms are omitted, though each weight matrix has one.

LSTM as follows:

$$\tilde{h}_j = \sum_{k \in B(j)} a_k, \tag{6.7}$$

$$i_j = \sigma\Big(W_{in} n_j + W_{ie} e_j + W_{ih} \tilde{h}_j\Big), \tag{6.8}$$

$$f_{jk} = \sigma\Big(W_{fn} n_j + W_{fe} e_j + W_{fh} a_k\Big), \tag{6.9}$$

$$o_j = \sigma\Big(W_{on} n_j + W_{oe} e_j + W_{oh} \tilde{h}_j\Big), \tag{6.10}$$

$$u_j = \tanh\Big(W_{un} n_j + W_{ue} e_j + W_{uh} \tilde{h}_j\Big), \tag{6.11}$$

$$c_j = i_j \odot u_j \sum_{k \in B(j)} f_{jk} \odot c_k, \tag{6.12}$$

$$a_j = o_j \odot \tanh(c_j). \tag{6.13}$$

Let J represent the number of nodes in tree-converted AMR structure obtained from a given input sentence. We introduce $A \in \mathbb{R}^{D \times J}$ as a matrix form of a list of hidden states a_j for all j, namely, $A = [a_1, \ldots, a_J]$. Let $O'' \in \mathbb{R}^{|V| \times D}$ be a weight matrix for the output layer. Let $S \in \mathbb{R}^{D \times (CD)}$ be a weight matrix for mapping the context embedding of C output words onto embeddings obtained from nodes. Then, we define the attention-based AMR encoder '$\mathrm{encAMR}(A, Y_{C,i})$' as follows:

$$\mathrm{encAMR}(A, Y_{C,i}) = O'' A s, \tag{6.14}$$

$$s \propto \exp(A^{\mathrm{T}} S \tilde{y}'_c). \tag{6.15}$$

Finally, we combine our attention-based AMR encoder shown in Equation 6.14 as an additional term of Equation 6.3 to build our headline generation system.

6.3 Experiments

To demonstrate the effectiveness of our proposed method, we conducted experiments on benchmark data of the abstractive headline generation task described in Rush et al. [2015].

For a fair comparison, we followed their evaluation setting. The training data was obtained from the first sentence and the headline of a document in the annotated Giga-

	DUC-2004			Gigaword test data used in [Rush et al., 2015]			Gigaword Our sampled test data		
Method	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
ABS [Rush et al., 2015]	26.55	7.06	22.05	30.88	12.22	27.77	–	–	–
ABS (re-run)	28.05	7.38	23.15	31.26	12.46	28.25	32.93	13.43	29.80
ABS+AMR	***28.80**	***7.83**	***23.62**	**31.64**	***12.94**	**28.54**	***33.43**	***13.93**	**30.20**
ABS+AMR(w/o attn)	28.28	7.21	23.12	30.89	12.40	27.94	31.32	12.83	28.46

Table 6.1: Results of methods on each dataset. We marked $*$ on the ABS+AMR results if we observed statistical difference ($p < 0.05$) between ABS (re-run) and ABS+AMR on the t-test. (R-1: ROUGE-1, R-2: ROUGE-2, R-L: ROUGE-L)

word corpus [Napoles et al., 2012][1]. The development data is DUC-2003 data, and test data are both DUC-2004 [Over et al., 2007] and sentence-headline pairs obtained from the annotated Gigaword corpus as well as training data[2]. All of the generated headlines were evaluated by ROUGE [Lin, 2004][3]. For evaluation on DUC-2004, we removed strings after 75-characters for each generated headline as described in the DUC-2004 evaluation. For evaluation on Gigaword, we forced the system outputs to be at most 8 words as in Rush et al. [2015] since the average length of headline in Gigaword is 8.3 words. For the pre-processing for all data, all letters were converted to lower case, all digits were replaced with '#', and words appearing less than five times with 'UNK'. Note that, for further evaluation, we prepared 2,000 sentence-headline pairs randomly sampled from the test data section of the Gigaword corpus as our additional test data.

In our experiments, we refer to the baseline neural attention-based abstractive summarization method described in Rush et al. [2015] as "**ABS**", and our proposed method of incorporating AMR structural information by a neural encoder to the baseline method described in Section 6.2 as "**ABS+AMR**". Additionally, we also evaluated the performance of the AMR encoder *without* the attention mechanism, which we refer to as "**ABS+AMR(w/o attn)**", to investigate the contribution of the attention mechanism on the AMR encoder. For the parameter estimation (training), we used stochastic gradient descent to learn parameters. We tried several values for the initial learning rate, and selected the value that achieved the best performance for each method. We decayed the learning rate by half if the log-likelihood on the validation set did not improve for an epoch. Hyper-parameters we selected were $D = 200$, $H = 400$,

[1]Training data can be obtained by using the script distributed by the authors of Rush et al. [2015].

[2]Gigaword test data can be obtained from `https://github.com/harvardnlp/sent-summary`

[3]We used the ROUGE-1.5.5 script with option "`-n2 -m -b75 -d`", and computed the average of each ROUGE score.

$N = 200$, $E = 50$, $C = 5$, and $Q = 2$. We re-normalized the embedding after each epoch [Hinton et al., 2012].

For ABS+AMR, we used the two-step training scheme to accelerate the training speed. The first phase learns the parameters of the ABS. The second phase trains the parameters of the AMR encoder by using 1 million training pairs while the parameters of the baseline ABS were fixed and unchanged to prevent overfitting.

Table 6.1 shows the recall of ROUGE [Lin, 2004] on each dataset. ABS (re-run) represents the performance of ABS re-trained by the distributed scripts[1]. We can see that the proposed method, ABS+AMR, outperforms the baseline ABS on all datasets. In particular, ABS+AMR achieved statistically significant gain from ABS (re-run) for ROUGE-1 and ROUGE-2 on DUC-2004. However in contrast, we observed that the improvements on Gigaword (the same test data as Rush et al. [2015]) seem to be limited compared with the DUC-2004 dataset. We assume that this limited gain is caused largely by the quality of AMR parsing results. This means that the Gigaword test data provided by Rush et al. [2015] is already pre-processed. Therefore, the quality of the AMR parsing results seems relatively worse on this pre-processed data since, for example, many low-occurrence words in the data were already replaced with "UNK". To provide evidence of this assumption, we also evaluated the performance on our randomly selected 2,000 sentence-headline test data also taken from the test data section of the annotated Gigaword corpus. "Gigaword (randomly sampled)" in Table 6.1 shows the results of this setting. We found the statistical difference between ABS(re-run) and ABS+AMR on ROUGE-1 and ROUGE-2.

We can also observe that ABS+AMR achieved the best ROUGE-1 scores on all of the test data. According to this fact, ABS+AMR tends to successfully yield semantically important words. In other words, embeddings encoded through the AMR encoder are useful for capturing important concepts in input sentences. Figure 6.3 supports this observation. For example, ABS+AMR successfully added the correct modifier 'saudi' to "crown prince" in the first example. Moreover, ABS+AMR generated a consistent subject in the third example.

The comparison between ABS+AMR(w/o attn) and ABS+AMR (with attention) suggests that the attention mechanism is necessary for AMR encoding. In other words, the encoder without the attention mechanism tends to be overfitting.

[1] `https://github.com/facebook/NAMAS`

I(1): crown prince abdallah ibn abdel aziz left saturday at the head of saudi arabia 's delegation to the islamic summit in islamabad , the official news agency spa reported .
G: saudi crown prince leaves for islamic summit
A: crown prince leaves for islamic summit in saudi arabia
P: saudi crown prince leaves for islamic summit in riyadh

I(2): a massive gothic revival building once christened the lunatic asylum west of the <unk> was auctioned off for $ #.# million -lrb- euro# .# million -rrb- .
G: massive ##th century us mental hospital fetches $ #.# million at auction
A: west african art sells for $ #.# million in
P: west african art auctioned off for $ #.# million

I(3): brooklyn , the new bastion of cool for many new yorkers , is poised to go mainstream chic .
G: high-end retailers are scouting sites in brooklyn
A: new yorkers are poised to go mainstream with chic
P: new york city is poised to go mainstream chic

Figure 6.3: Examples of generated headlines on Gigaword. **I**: input, **G**: true headline, **A**: ABS (re-run), and **P**: ABS+AMR.

6.4 Related Work

Recently, the Recurrent Neural Network (RNN) and its variant have been applied successfully to various NLP tasks. For headline generation tasks, Chopra et al. [2016] exploited the RNN decoder (and its variant) with the attention mechanism instead of the method of Rush et al. [2015]: the combination of the feed-forward neural network language model and attention-based sentence encoder. Nallapati et al. [2016] also adapted the RNN encoder-decoder with attention for headline generation tasks. Moreover, they made some efforts such as hierarchical attention to improve the performance. In addition to using a variant of RNN, Gulcehre et al. [2016] proposed a method to handle infrequent words in natural language generation. Note that these recent developments do not conflict with our method using the AMR encoder. This is because the AMR encoder can be straightforwardly incorporated into their methods as we have done in this paper, incorporating the AMR encoder into the baseline. We

believe that our AMR encoder can possibly further improve the performance of their methods. We will test that hypothesis in future study.

6.5 Conclusion

This chapter mainly discussed the usefulness of incorporating structural syntactic and semantic information into novel attention-based encoder-decoder models on headline generation tasks. We selected abstract meaning representation (AMR) as syntactic and semantic information, and proposed an attention-based AMR encoder-decoder model. The experimental results of headline generation benchmark data showed that our attention-based AMR encoder-decoder model successfully improved standard automatic evaluation measures of headline generation tasks, ROUGE-1, ROUGE-2, and ROUGE-L. We believe that our results provide empirical evidence that syntactic and semantic information obtained from an automatic parser can help to improve the neural encoder-decoder approach in NLG tasks.

Chapter 7

Conclusions

While modeling the meanings of phrases is crucial technique in natural language processing, few studies addressed to handle an arbitrary-length phrase. In this thesis, to compute the meanings of arbitrary-length phrases, we have addressed three issues:

1. **What is the most suitable way to encode an arbitrary-length phrase?** There are various neural encoders to compute distributed representations of phrases but few researchers has paid attention to revealing the performance of each encoder.

2. **Are distributed representations of phrases efficient in NLP applications?** It is intuitive that distributed representations of phrases are useful in downstream tasks but the enhancement of them is unclear.

3. **Do additional distributed representations enhance the performance of NLP applications?** To make the worth of NLP fundamental tasks clear, it is important to investigate the improvement by distributed representations of syntactic and semantic features.

The key contribution of this thesis can be summarized as follows:

- We constructed a new dataset to evaluate phrase composition. The dataset shows high inter-annotator agreement by following the annotation guideline of Mitchell and Lapata [2010]. We made the constructed dataset public on the Web[1].

[1] `http://github.com/takase/relPatSim`

- We proposed two novel encoders. One is modified recursive neural network to model the verbs that change/inherit the meaning. The other one is more general way because the neural encoder can identify the contribution of each word to the meaning of a phrase automatically.
- We explored the suitable way to compose distributed representations of phrases by comparing well know encoders on various datasets.
- We indicated that distributed representations computed by an encoder are useful to NLP applications such as relation classification.
- We proposed a novel encoder for additional syntactic/semantic information such as POS tags and demonstrated the enhancement by such information on the headline generation task.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *The First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 385–393, 2012. 32

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013. 59

Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 1183–1193, 2010. 27

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226, 2009. 2

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 238–247, 2014. 2, 9

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003. 12, 27, 56

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, pages 1306–1313, 2010. 26

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua 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 2014)*, pages 1724–1734, 2014. 4, 34, 36, 52, 55

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98, 2016. 63

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014. 37

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 594–602, 2006. 23

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 160–167, 2008. 12, 27

Cicero dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 626–634, 2015. 24, 49, 51, 53

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12: 2121–2159, 2011. 40

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 34

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1535–1545, 2011. 17, 29, 43

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web (WWW 2001)*, pages 406–414, 2001. 20

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 758–764, 2013. 32, 40

Matthew R. Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1774–1784, 2015. 49, 53

Adam Grycner, Gerhard Weikum, Jay Pujara, James Foulds, and Lise Getoor. Relly: Inferring hypernym relationships between relational phrases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 971–981, 2015. 52

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 140–149, 2016. 63

Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361, 2012. 8

Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954. iii, 1, 11

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1544–1555, 2014. 28, 44, 52

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL 2015)*, pages 268–278, 2015. 24, 25, 49, 51

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)*, pages 33–38, 2010. 20, 48

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1367–1377, 2016a. 51

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30, 2016b. 52

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *CoRR*, abs/1207.0580, 2012. 62

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 35

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, pages 57–60, 2006. 59

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 873–882, 2012. 20

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 655–665, 2014. 39

Tom Kenter, Alexey Borisov, and Maarten de Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 941–951, 2016. 52

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 42

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 3276–3284. 2015. 51, 52

Philipp Koehn. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 388–395, 2004. 23

Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. Semeval-2013 task 5: Evaluating phrasal semantics. In *Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 39–47, 2013. 32, 39, 41

Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 39

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th Advances in Neural Information Processing Systems (NIPS 2014)*, pages 2177–2185. 2014a. 10, 27

Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL 2014)*, pages 171–180, 2014b. 27

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics (TACL 2015)*, 3:211–225, 2015. 10

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the Association for Computational Linguistics Workshop*, pages 74–81, 2004. 61, 62

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013a. 7, 8

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26 th Advances in Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119. 2013b. 2, 8, 12, 18, 20, 27, 43, 51, 52

George A Miller and Walter G Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991. 20

Bonan Min, Shuming Shi, Ralph Grishman, and Chin-Yew Lin. Ensemble semantics for large-scale unsupervised relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP 2012)*, pages 1027–1037, 2012. 11, 26

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439, 2010. 4, 27, 30, 32, 34, 39, 40, 41, 51, 65

Thahir Mohamed, Estevam Hruschka, and Tom Mitchell. Discovering relations between noun categories. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1447–1455, 2011. 11

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. Finding the best model among representative compositional models. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation (PACLIC 2014)*, pages 65–74, 2014. 21, 27, 34, 51

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP 2012)*, pages 1135–1145, 2012. 2, 11, 26, 27, 52

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL 2016)*, pages 280–290, 2016. 63

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, 2012. 61

Paul Over, Hoa Dang, and Donna Harman. DUC in Context. *Information Processing and Management*, 43(6):1506–1520, 2007. 56, 61

Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, pages 113–120, 2006. 26

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014. 2, 9, 27

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 971–981, 2015. 52

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 74–84, 2013. 53

Bryan Rink and Sanda Harabagiu. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)*, pages 256–259, 2010. 24, 49

Benjamin Rosenfeld and Ronen Feldman. Using corpus statistics on entities to improve semi-supervised relation extraction from the web. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 600–607, 2007. 26

Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965. 20

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389, 2015. 55, 56, 57, 60, 61, 62, 63

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase

Detection. In *Proceedings of the 24th Advances in Neural Information Processing Systems (NIPS 2011)*, pages 801–809. 2011a. 27, 52

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine learning (ICML 2011)*, pages 129–136, 2011b. 4, 5, 13, 16, 24, 27, 35, 52

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP 2012)*, pages 1201–1211, 2012. 12, 13, 14, 18, 24, 27, 49, 52

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 455–465, 2013. 52

Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 1017–1024, 2011. 52

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112, 2014. 34, 36, 52, 53

Jun Suzuki and Masaaki Nagata. A unified learning framework of skip-grams and global vectors. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 186–191, 2015. 10

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 1556–1566, 2015. 55, 59

Sho Takase, Naoaki Okazaki, and Kentaro Inui. Modeling semantic compositionality of relational patterns. *Engineering Applications of Artificial Intelligence*, 50(C): 256–264, 2016a. 52

Sho Takase, Naoaki Okazaki, and Kentaro Inui. Composing distributed representations of relational patterns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 2276–2286, 2016b. 36, 51, 52

Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1422–1432, 2015. 53

Ran Tian, Naoaki Okazaki, and Kentaro Inui. The mechanism of additive composition. *CoRR*, abs/1511.08407, 2016. 34, 38, 52

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1499–1509, 2015. 53

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating Videos to Natural Language Using Deep Recurrent Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 1494–1504, 2015. 55

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR 2015)*, pages 3156–3164, 2015. 55

Chuan Wang, Nianwen Xue, and Sameer Pradhan. A Transition-based Algorithm for AMR Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 366–375, 2015. 59

Christian Wartena. Hsh: Estimating semantic similarity of words and short phrases with frequency normalized distance measures. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 48–52, 2013. 44

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics (TACL 2015)*, 3:345–358, 2015. 32, 39, 40, 41, 42, 43, 44

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 536–540, 2015. 24, 49, 51, 53

Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 643–648, 2014. 53

Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 901–911, 2015. 39

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016. 39

Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics (TACL 2015)*, 3:227–242, 2015. 4, 42, 44, 52

Mo Yu, Matthew R. Gormley, and Mark Dredze. Factor-based compositional embedding models. In *Workshop on Learning Semantics at the 2014 Conference on Neural Information Processing Systems (NIPS 2014)*, December 2014. 24, 25

Naomi Zeichner, Jonathan Berant, and Ido Dagan. Crowdsourcing inference-rule evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2012)*, pages 156–160, 2012. 5, 19, 29, 30, 32, 43

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2335–2344, 2014. 24

List of Publications

Journal Papers (Refereed)

1. Sho Takase, Naoaki Okazaki, Kentaro Inui. Modeling semantic compositionality of relational patterns. Engineering Applications of Artificial Intelligence, vol. 50, pp.256-264, April 2016.

2. Sho Takase, Naoaki Okazaki, Kentaro Inui. Set Expansion Using Sibling Relationships Between Semantic Categories (in Japanese). In Journal of Natural Language Processing, vol. 20, No. 2, June 2013.

International Conference/Workshop Papers (Refereed)

1. Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao and Masaaki Nagata. Neural Headline Generation on Abstractive Meaning Representation. In proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), pp. 1054-1059, November 2016.

2. Sho Takase, Naoaki Okazaki and Kentaro Inui. Composing Distributed Representations of Relational Patterns. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pp.2276-2286, August 2016.

3. Sho Takase, Naoaki Okazaki and Kentaro Inui. Fast and Large-scale Unsupervised Relation Extraction. In Proceedings of the 29th Pacific Asia Conference on Language Information and Computing (PACLIC29), pp.96-105, October 2015.

4. Sho Takase, Akiko Murakami, Miki Enoki, Naoaki Okazaki and Kentaro Inui. Detecting Chronic Critics Based on Sentiment Polarity and User's Behavior in Social Media. In Proceedings of the ACL student research workshop 2013.

5. Sho Takase, Naoaki Okazaki and Kentaro Inui. Set Expansion using Sibling Relations between Semantic Categories. In Proceedings of the 26th Pacific Asia Conference on Language Information and Computing (PACLIC26), November 2012.

Awards

1. The 21th Annual Meeting of the Association for Natural Language Processing Excellent Paper Award (2015)
2. The 29th Annual Meeting of the JSAI Conference Student Incentive Award (2015)

Other Publications (Not refereed)

1. Sho Takase, Naoaki Okazaki and Kentaro Inui. Meaning Representation Learning for Relation Knowledge Acquisition (in Japanese). In Proceedings of the 29th Annual Conference of the Japanese Society for Artificial Intelligence, May 2015.
2. Sho Takase, Naoaki Okazaki and Kentaro Inui. Computing the Meanings of Relational Patterns based on Compositionality (in Japanese). In Proceedings of the 21th Annual Meeting of the Association for Natural Language Processing, pp. 640–643, March 2015.
3. Sho Takase, Naoaki Okazaki and Kentaro Inui. Relation Acquisition from Large Corpus by usin Approximate Counting (in Japanese). In IPSJ SIG Technical Reports, Vol.2014-NL-217, July 2014.
4. Sho Takase, Naoaki Okazaki and Kentaro Inui. Unsupervised Relation Extraction with Fast Similarity Calculation (in Japanese). In Proceedings of the 20th Annual Meeting of the Association for Natural Language Processing, pp.47–50, March 2014.
5. Sho Takase, Akiko Murakami, Miki Enoki, Naoaki Okazaki and Kentaro Inui. Detecting Chronic Critics Based on Sentiment Polarity and User's Behavior in

Social Media (in Japanese). In Proceedings of the 19th Annual Meeting of the Association for Natural Language Processing, pp.260–263, March 2013.

6. Sho Takase, Naoaki Okazaki, Kentaro Inui. For Relation Knowledge Acquisition from Noun Categories (in Japanese). The 7th NLP Symposium for Young Researchers, September 2012.

7. Sho Takase, Naoaki Okazaki and kentaro Inui. Set Expansion by using Hierarchical Relationship between Semantic Categories (in Japanese). In Proceedings of the 18th Annual Meeting of the Association for Natural Language Processing, pp.475-478, March 2012.