Natural Language Inference with Definition Embedding Considering Context On the Fly

Kosuke Nishida, Kyosuke Nishida, Hisako Asano, Junji Tomita
NTT Media Intelligence Laboratories, NTT Corporation
1-1 Hikarinooka Yokosuka, Kanagawa, Japan
nishida.kosuke@lab.ntt.co.jp

Abstract

Natural language inference (NLI) is one of the most important tasks in NLP. In this study, we propose a novel method using word dictionaries, which are pairs of a word and its definition, as external knowledge. Our neural definition embedding mechanism encodes input sentences with the definitions of each word of the sentences on the fly. It can encode definitions of words considering the context of the input sentences by using an attention mechanism. We evaluated our method using WordNet as a dictionary and confirmed that it performed better than baseline models when using the full or a subset of $100d$ GloVe as word embeddings.

1 Introduction

Recognition of the entailment relationship between two sentences is one of the most important tasks in the field of natural language processing. An understanding of entailment relationships among sentences is useful for performing tasks such as question answering, information retrieval, and summarization.

The task of recognizing the entailment relationship between two sentences is called recognizing textual entailment (RTE) or natural language inference (NLI). NLI has recently been getting more attention from researchers, owing to the release of large-scale corpora such as SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018).

These corpora consist of pairs of sentences, such as 'A soccer game with multiple males playing.' and 'Some men are playing a sport.', and ground-truth labels. Each label is a judgment of whether the latter sentence, which is the premise, is inferred from the former one, which is the hypothesis. In this example, the label is 'entailment'.

In this study, we propose a novel method that uses word dictionaries as external knowledge. Word dictionaries are useful for domain adaptation, where we need to understand rare or novel words in which we do not have good embedding representations. For NLI, there is related work that does use dictionaries (Bahdanau et al., 2017). In it, a definition embedding method is proposed that obtains representations of out-of-vocabulary (OOV) words from dictionaries on the fly. In this method, however, the description of a word is converted into the same embedding anytime without considering the context of the input sentences.

On the other hand, we consider that word representation from dictionaries should reflect the context of the input sentences. In the dictionary, we can explain the meaning of a word from many aspects. However, the required information varies depending on the context of the input sentences. This problem also occurs for pre-trained word embeddings, which are usually fixed for all contexts in the previous studies.

The proposed method can obtain different representations of words according to the contexts of the input sentences. It introduces an attention mechanism that improves the encoded representations of each word in input sentences, by using the encoded word definitions of each word in the input sentences. Moreover, unlike Bahdanau's method, it obtains the representation of *all* words from dictionaries on the fly in order to improve the representations of non-OOV words.

2 Task Definition

We follow the task definition of SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). We define a dictionary as follows.

Proceedings of the 3rd Workshop on Representation Learning for NLP, pages 58–63
Melbourne, Australia, July 20, 2018. ©2018 Association for Computational Linguistics

Def. 1 (Dictionary)**.** A dictionary $\mathcal{D}$ has the following components.

Headword y is an arbitrary token. **Definition** D^y is represented as a token sequence that defines the headword y. This study assumes that each headword has only one definition. For a polysemic headword with multiple definitions, we use the concatenation of the definitions. **Vocabulary** V_{D} is the set of all headwords in the dictionary.

3 Related Work

Bahdanau et al. (2017) proposed a method that enables dictionary information to be used in NLP tasks, such as NLI, reading comprehension, and language modeling. Their method can obtain the embeddings of OOV words efficiently, because they obtain the definition embeddings for only OOV words instead of the random embeddings of the words. Our method is similar to theirs, but our purpose is different; we refine word embeddings considering the contexts of input sentences for all words.

The definition embedding is also useful for other tasks. Hill et al. (2016) used the definition embedding to understand the phrases. They presented two applications: reverse dictionaries and crossword question answering. They tackled these applications with phrase embeddings obtained from their definitions. Long et al. (2016) used the encoding of the word definition for the initialization of TransE (Bordes et al., 2013), which obtains the embedding of the relationship between two entities.

There is related work that uses other external resources for refining word representations. For NLI, Chen et al. (2017a) proposed a model that uses a knowledge graph to reflect word relationships (e.g., synonymy, hypernymy). Their method achieved state-of-the-art performance on SNLI; however, it cannot handle the definition description of each word.

Moreover, there are general frameworks to refine word embeddings by using external knowledge. Weissenborn et al. (2017) proposed a method that refines the word embedding by encoding the text transformation of ConceptNet (Speer and Havasi, 2012). McCann et al. (2017) proposed context vectors (CoVe), which uses a RNN encoder trained on machine translation datasets to introduce context information to the word embedding. Peters et al. (2018) proposed the Embeddings from Language Models (ELMo), which obtains contextualized word representations. They used the states of the middle layers in the deep language model. These methods are also effective at the NLI task.

4 Existing Methods

This section outlines the existing NLI models and describes the conventional model that uses a dictionary as external knowledge.

4.1 NLI model

In the architecture of a general NLI model (Bowman et al., 2015; Rocktäschel et al., 2016; Chen et al., 2017b), the input of the model is a pair of token sequences $\{X^s = (x^s_1, \cdots, x^s_{l_s}) : s \in \{\mathrm{P}, \mathrm{H}\}\}$, where l_s is the length of X^s. $s \in \{\mathrm{P}, \mathrm{H}\}$ means a premise or hypothesis.

We call the following two layers together the **Encoder**.

Encoder Word Embedding Layer (WEL) This layer takes X^s as input. Let $e(y) \in \mathbb{R}^{n_e}$ be the embedding of token y. It outputs a vector sequence $E^s = (e(x^s_1), \cdots, e(x^s_{l_s})) \in \mathbb{R}^{n_e \times l_s}$.

Encoder Context Embedding Layer (CEL) This layer converts the vector sequence E^s into a contextualized vector sequence, $C^s = f(E^s) \in \mathbb{R}^{n_c \times l_s}$. The most common approach is to use an RNN as f.

The encoder outputs C^P and C^H for the premise and hypothesis sentences, respectively.

Decoder The input of the decoder is a pair of vector sequences $\{C^P, C^H\}$. The decoder outputs the score vector of the classification labels.

4.2 Definition Embedding Mechanism

We summarize the definition embedding mechanism (DEM) (Bahdanau et al., 2017) as it relates to NLI. They proposed dictionary embedding mechanisms with many variations, such as mean pooling or an RNN. We select one of their models with an RNN, because we also use an RNN for the definition embedding.

The DEM acts on each premise and hypothesis. Its input is a token sequence X^s and the encoder word embedding sequence E^s. The output is E'^s, and E'^s is passed to the encoder CEL instead of E^s. E'^s is obtained by adding E^s to the final state of the RNN encoding of the definition. The sizes of E^s and E'^s are each $n_{\mathrm{e}} \times l_s$.

5 Proposed Method

We propose a novel DEM considering the contexts of the input sentences. Our contributions are threefold. First, we introduce an attention mechanism. Second, we implement the mechanism after the encoder. Third, we consider definition embeddings of words including non-OOV ones.

The input is a token sequence X^s together with the encoder word and context embedding sequence E^s and C^s, and the output is C'^s. C'^s is passed to the *decoder* instead of C^s, where the sizes of C^s and C'^s are each $n_\mathrm{c} \times l_s$. The proposed mechanism has the following layers.

Definition Extracting Layer Let V^s be the set of target tokens of the definition embedding which are in both the token sequence X^s and the vocabulary of the dictionary V_D. The definition D^y of token $y \in V^s$ is obtained from the dictionary $\mathcal{D}$. Let m_y be the length of D^y. This layer outputs a set of target tokens V^s and a set of definitions $\{D^y : y \in V^s\}$.

Definition WEL This layer has the same parameters as the encoder WEL. For each element of D^y, it outputs a vector sequence $E^y = (e(d^y_1), \cdots, e(d^y_{m_y})) \in \mathbb{R}^{n_\mathrm{e} \times m_y}$.

Definition CEL This layer has the same model as the encoder CEL. Parameters are not shared with the encoder CEL. It converts the vector sequence E^y into the output of this layer $C^y = f(E^y) \in \mathbb{R}^{n_\mathrm{c} \times m_y}$.

Definition Attention Layer This layer obtains a fixed-length vector representation of definition D^y with an attention mechanism. It takes the outputs of the previous layers E^y, C^y, C^s, and $C^{\bar{s}}$ as input, where $\bar{s} \in \{\mathrm{P}, \mathrm{H}\}$ indicates that either the premise or hypothesis is different from s.

For $C^y \in \mathbb{R}^{n_c \times m_y}, C^s \in \mathbb{R}^{n_c \times l_s}$, we define an attention matrix $A^{y,s} = \frac{1}{\sqrt{n_c}} C^{s\top} C^y$, and an attention vector $a^{y,s} = \left(\frac{1}{l_s} \sum_i A^{y,s}_{ij}\right)_{j=1,\cdots,m_y} \in \mathbb{R}^{m_y}$. The attention vector $a^{y,s}$ represents the extent that each token in definition D^y is related with the input sentence X^s. The attended definition vector to the input sentence X^s is

$$h^{y,s} = \sum\nolimits_i \mathrm{softmax}_i(a^{y,s}) c^y_i \in \mathbb{R}^{n_\mathrm{c}},$$

where c^y_i is the i-th state of the definition context embedding C^y.

The last state of the definition context embedding is $c^y_{m_y} \in \mathbb{R}^{n_\mathrm{c}}$. The output of this layer is a linear combination of the enhancements (Chen et al., 2017b) of the attended definition vectors,

$$\begin{aligned} z^y = [&c^y_{m_y}, h^{s,y,\mathrm{a}}, h^{s,y,\mathrm{a}} - c^y_{m_y}, h^{s,y,\mathrm{a}} \odot c^y_{m_y}, \\ &h^{\bar{s},y,\mathrm{a}}, h^{\bar{s},y,\mathrm{a}} - c^y_{m_y}, h^{\bar{s},y,\mathrm{a}} \odot c^y_{m_y}] w, \end{aligned} \quad (1)$$

where $w \in \mathbb{R}^7$ is a trainable parameter and $\odot$ is the element-wise product. n_c is the size of z^y.

Output Layer The output of the proposed mechanism is expressed as

$$c'^s_i = \begin{cases} c^s_i + z^{x^s_i} & (x^s_i \in V^s) \\ c^s_i & \text{otherwise} \end{cases}. \quad (2)$$

The decoder receives C'^s instead of C^s.

Algorithm 1 is the pseudo code of the definition embedding mechanism.

The above explanation only covers the case of NLI. However, the proposed method can be applied to any number of input sentences, because Equation (1) can take an arbitrary number of arguments. Therefore, it is applicable to other tasks that have text inputs, such as question answering and machine translation.

Algorithm 1 Definition Embedding

Input: $X^s, E^s, C^s, C^{\bar{s}}$
Output: C'^s

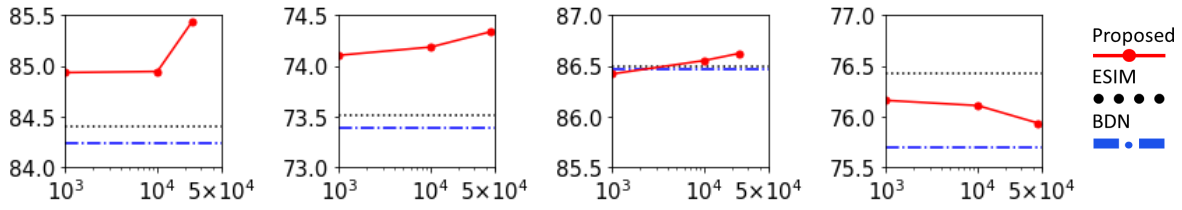
```
V^s, {D^y : y ∈ V^s} ← Def. Ext.(X^s)
for all y in V^s do
    E^y ← Def. Word Emb.(D^y)
    C^y ← Def. Context Emb.(E^y)
    z^y ← Def. Att.(E^y, C^y, C^s, C^s̄)
end for
C'^s ← Output(E^s, C^s, V^s, {z^y : y ∈ V^s})
```

6 Experiments

This section describes the results of the evaluation of the proposed method.

6.1 Experimental Setup

We chose ESIM (Chen et al., 2017b) and one of the methods in Bahdanau et al. (2017) (BDN) as the baseline models. ESIM is based on the model in Section 4.1. BDN and our method each add a DEM to ESIM. In BDN, the target tokens of the definition embedding are not contained in the pre-trained word embedding vocabulary, because



[a] SNLI, $d_e = 100$ [b] MNLI, $d_e = 100$ [c] SNLI, $d_e = 300$ [d] MNLI, $d_e = 300$

Figure 1: Classification accuracy of each model in the not-many-OOV setting. The vertical axis is accuracy, and the horizontal axis is the number of the vocabulary entries of the dictionary. The performance of ESIM and BDN was constant because their dictionary size is less than 1000.

BDN intends to supplement the embeddings of OOV words. However, in our method, the target tokens do not depend on a pre-trained word embedding vocabulary, because we intend to improve the representation of all the words by considering the context.

Our experiments were on the SNLI and MNLI benchmarks. For MNLI, we used a matched domain development dataset as our development data and a mismatched domain development dataset as our test data. The tokenizer was spaCy (Honnibal and Montani, 2018). The word embeddings were pre-trained $100d$ GloVe 6B vectors and $300d$ GloVe 840B vectors (Jeffrey Pennington and Manning, 2014). The embeddings were fixed during training, because we were interested in the difference in representation between pre-trained embeddings with and without dictionary information.

We used the vocabulary and definitions in WordNet (Miller, 1995) as dictionaries. For polysemic words with multiple definitions, we used the top-5 definitions connected in descending order of frequency of synsets, which are provided by WordNet. The number of headwords that appear in SNLI is 24103, and 45225 in MNLI.

The other settings are described in Appendix A.

6.2 Results

Does the proposed method refine the OOV word embedding? In order to investigate the effectiveness of our method against OOV words, we restricted the vocabulary of the $100d$ GloVe embedding to the most common 3000 words in each dataset and considered the other words as OOV (many-OOV setting). The word embeddings of the OOV words were randomly initialized according to a Gaussian distribution and fixed during training.

Table 1 shows the results. When there were many OOV words, our method improved test accuracy by 1.4% in SNLI and 1.5% in MNLI. In contrast, BDN did not improve accuracy in MNLI.

	SNLI	MNLI
ESIM	82.5	69.8
BDN	83.7	69.7
Proposed	**83.9**	**71.3**

Table 1: Test accuracy in the many-OOV setting

Does the larger dictionary bring higher accuracy? We also evaluated our method with the whole $100d$ GloVe embedding (not-many-OOV setting). In this experiment, we used the whole vocabulary of WordNet or restricted the WordNet vocabulary to the 1000 and 10000 most common words in the each dataset.

Figures 1a and 1b show the results when using $100d$ GloVe. We confirmed that the larger dictionary raises accuracy. We think that the pre-trained GloVe embeddings for the frequent words were more appropriate than those for the rare words. This means that our method was effective for words that had relatively poor embeddings and occur sufficiently often in the training data.

We confirmed that the threefold originality of our method contributed to the improvement in the whole WordNet setting. The proposed method using the whole WordNet achieved the higher test accuracy on each dataset. The improvement from ESIM was 1.0% in SNLI and 0.8% in MNLI. Moreover, our method without the definition attention mechanism performed worse by 0.4% in SNLI and 0.5% in MNLI in comparison with the method with it. This implies that our definition embedding layer plays an important role in the definition embedding. In particular, the implementation of the attention mechanism after the encoder, which is essential to reflecting the context of input sentences, contributes to a refined representation.

BDN did not perform well. The number of

OOV words in SNLI (MNLI) is 415 (913); therefore, BDN could not sufficiently train the representations of the words with the sentences in the datasets.

Does the improvement depend on the quality of the word embedding? Figures 1c and 1d show the results when using $300d$ GloVe. In this setting, our method provided no significant improvement. It performed slightly better (worse) than ESIM in SNLI (MNLI). BDN, as well, did not perform better than ESIM. We think the $300d$ GloVe has sufficiently correct embeddings for most of the words in SNLI and MNLI, because it was created from a much larger corpora (340 billion tokens) than that of the 100d one (eight billion tokens).

To summarize the experimental results for the first and third research questions, the effectiveness of our method is dependent on the quality and coverage of word embeddings. That is, our method is effective for rare or novel words.

7 Conclusion

We proposed a novel definition embedding method. The method considers the contexts of the input sentences with an attention mechanism for the definition embeddings. It considers the definition embeddings of words including non-OOV words. Experimental results showed that it is effective for rare or novel words that do not have good pre-trained word embeddings.

References

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286* .

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. pages 2787–2795.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*. pages 632–642. https://doi.org/10.18653/v1/D15-1075.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2017a. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289* .

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *ACL*. pages 1657–1668. https://doi.org/10.18653/v1/P17-1152.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. pages 249–256.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of ACL* 4:17–30.

Matthew Honnibal and Ines Montani. 2018. spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* .

Richard Socher Jeffrey Pennington and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Tao Lei and Yu Zhang. 2017. Training RNNs as fast as CNNs. *arXiv preprint arXiv:1709.02755* .

Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging lexical resources for learning entity embeddings in multi-relational data. pages 112–117. https://doi.org/10.18653/v1/P16-2019.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

George A. Miller. 1995. WordNet: A lexical database for english. *Commun. ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS Workshop Autodiff* .

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR*.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*. pages 3679–3686.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural NLU systems. *arXiv preprint arXiv:1706.02596* .

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

A Details of the Implementation

The section describes our implementation so that our experiments can be reproduced.

We implemented our method in PyTorch (Paszke et al., 2017) and trained it on one Nvidia GeForce GTX 1080 GPU. The RNNs in the encoder, decoder, and definition embedding mechanism were two-layer bi-directional simple recurrent units (SRUs) (Lei and Zhang, 2017). The size of the output of the RNN was $n_c = 2n_e$. The activation function in the RNN was the tanh function. Dropout with a keep ratio of 0.8 was applied to the same layer as ESIM and the definition embedding layer.

The parameters of the weights were initialized using the Xavier normal initializer (Glorot and Bengio, 2010), and the parameters of the biases were initialized as zero vectors. Word embeddings not contained in pre-trained GloVe were randomized according to a Gaussian distribution.

The mini-batch size was set to 16. The optimizer was Adadelta (Zeiler, 2012) with an initial learning rate of 0.075 and ρ of 0.9. Early stopping with a patience of 7 was used to avoid overfitting.

We removed words whose definition length was one and stop words in the Natural Language Toolkit (Bird et al., 2009) from the vocabulary of the dictionary.