Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

A Context-Enriched Neural Network Method for Recognizing Lexical Entailment

Kun Zhang,[†] **Enhong Chen,**[†*] **Qi Liu,**[†] **Chuanren Liu,**[‡] **Guangyi Lv**[†]
[†]School of Computer Science and Technology, University of Science and Technology of China
zhkun@mail.ustc.edu.cn, cheneh@ustc.edu.cn, qiliuql@ustc.edu.cn, gylv@mail.ustc.edu.cn
[‡]Drexel University
chuanren.liu@drexel.edu

Abstract

Recognizing lexical entailment (RLE) always plays an important role in inference of natural language, i.e., identifying whether one word entails another, for example, *fox* entails *animal*. In the literature, automatically recognizing lexical entailment for word pairs deeply relies on words' contextual representations. However, as a "prototype" vector, a single representation cannot reveal multifaceted aspects of the words due to their homonymy and polysemy. In this paper, we propose a supervised Context-Enriched Neural Network (CENN) method for recognizing lexical entailment. To be specific, we first utilize multiple embedding vectors from different contexts to represent the input word pairs. Then, through different combination methods and attention mechanism, we integrate different embedding vectors and optimize their weights to predict whether there are entailment relations in word pairs. Moreover, our proposed framework is flexible and open to handle different word contexts and entailment perspectives in the text corpus. Extensive experiments on five datasets show that our approach significantly improves the performance of automatic RLE in comparison with several state-of-the-art methods.

1 Introduction

Recognizing Textual Entailment (RTE) is an important task in natural language processing, particularly for applications such as text summarization, information retrieval, question answering, paraphrasing and others (Androutsopoulos and Malakasiotis 2010). RTE involves pairs of sentences, such as the following (Turney and Mohammad 2015):

Text: *George was bitten by a dog.*
Hypothesis: *George was attacked by an animal.*

The objective of RTE is to develop algorithms that can determine whether the *text* sentence entails the *hypothesis* sentence. In many cases, to recognize the *entailment between sentences*, we must first be able to recognize the *entailment between words* (Geffet and Dagan 2005), which is called Recognizing Lexical Entailment (RLE). In the example above, if we are able to recognize that *bitten* entails *attacked* and *dog* entails *animal*, we can conclude that the *text* entails the *hypothesis*.

*Corresponding author.

The state-of-the-art methods for RLE rely on representing each word x with a "prototype" vector $\boldsymbol{x}$ of contextual features, i.e., distribution of other words that tends to appear in its vicinity. Based on the vector representations of the words, most works can be grouped into two parts: unsupervised methods and supervised methods. The unsupervised RLE methods mainly use asymmetric similarity functions to measure the existence of the entailment relations in word pairs (Santus et al. 2014; Kotlerman et al. 2010). Supervised methods learn the asymmetric operator from a training set, which thus can perform better. Those supervised methods differ by the way they represent each candidate pair of words (x, y), such as concatenation $\boldsymbol{x} \oplus \boldsymbol{y}$ (Baroni et al. 2012) and difference $\boldsymbol{y} - \boldsymbol{x}$ (Fu et al. 2014). Recently, more sophisticated representations have also been tested (Turney and Mohammad 2015).

However, due to the homonymy and polysemy of words, capturing the semantics of a word with a single vector can be an unattainable goal (Reisinger and Mooney 2010; Hua et al.). For instance, the word *club* is similar to both *bat* and *association*, but these two words are not similar to each other in general contexts. Thus only a single "prototype" vector cannot reveal multifaceted aspects of a word.

To overcome the shortcomings of single representation of a word, in this paper, we propose a supervised Context-Enriched Neural Network (CENN) method for RLE. Specifically, we use multiple word embeddings with different contexts to represent words and word pairs, which can capture much more aspects of the language. Then we use different combination methods to integrate the multiple semantic information and neural network structure with attention mechanism to determine the importance of the information from different contexts for learning the entailment relations. Our design makes CENN flexible and adaptable to different entailment perspectives in the text corpus. The extensive evaluations show that our method significantly improves the performance of automatic RLE in comparison with several state-of-the-art methods. To the best of our knowledge, this is the first work using neural network to integrate multiple language embeddings for the task of automatic RLE.

2 Preliminary and Problem Statement

In this section, we formulate the RLE task with multiple contextual information as a supervised classification problem.

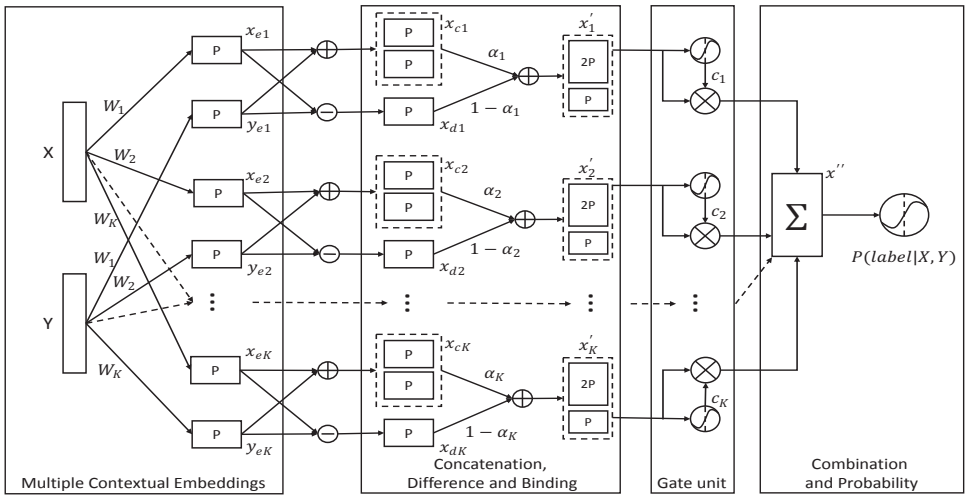


Figure 1: The CENN Structure. Notations: 1) $(\boldsymbol{X}, \boldsymbol{Y})$: the one-hot representation of a word pair. 2) $\boldsymbol{W_k}$: the contextual embedding/representation matrix, $k = \{1, 2, \cdots, K\}$. 3) P: the embedding/representation dimension.

First of all, the input for automatic RLE, $(\boldsymbol{X}, \boldsymbol{Y})$, contains the one-hot representation vectors of a word pair (x, y), i.e., $\boldsymbol{X_d} = 1$ if x is the d-th word in vocabulary, $\boldsymbol{X_d} = 0$ otherwise. $\boldsymbol{Y}$ is defined in the same way.

Then in the existing work of RLE, each word x is represented by a single vector. In this paper, we use multiple contextual embeddings $\boldsymbol{W_k} \in \mathbb{R}^{P \times D}$, $k \in \{1, 2, \cdots, K\}$, to represent the words. Here, K is the number of representation contexts. D is the total number of words, and P is the representation vector's dimension. The contexts of words are the distribution of other words that tend to appear in target word's vicinity, e.g., Bag-of-Words context is the distribution of a specific number of words that appear in the both sides of target words (Mikolov et al. 2013). Section 4 will provide a detailed description.

In the supervised setting of automatic RLE, we have training data $\{(x, y, label)\}$, where x and y represent the words. For the $label$, if x entails y, label is 1, otherwise, label is 0. Please note that, generally (x, y) and (y, x) are two different word pairs. Thus the respective labels can be different, i.e., there might be an entailment relation in (*x, y*), but not in (*y, x*). Taking (*fox, animal*) as an example, we can infer *animal* from *fox*, but we cannot infer *fox* from *animal*.

Our task in this paper is to learn an accurate classification model, to predict the $label$ given a word pair (x, y). To this end, we propose a supervised neural network based model to tackle this issue.

3 The Context-Enriched Neural Network

In this section, we formally introduce the structure and technical details of the supervised Context-Enriched Neural Network (CENN) model for RLE.

The overall structure of CENN is shown in Fig.1. First, the input of CENN is $(\boldsymbol{X}, \boldsymbol{Y})$, containing the one-hot representation vectors of a word pair. Second, we multiply the one-hot representation vectors of words by different embedding matrices $\boldsymbol{W_k}$ to get the corresponding contextual representations. Third, we arrange the representations of x and y by different combinations. Finally, we use attention mechanism (gate unit) to determine the contribution of each contextual vector and estimate the output function $P(label|\boldsymbol{X}, \boldsymbol{Y})$. More details are given as follows.

Multiple Contextual Embeddings. We multiply the one-hot vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ by the embedding matrices $\boldsymbol{W_k}$ to get the contextual representations in the embedding spaces:

$$\boldsymbol{x_{ek}} = \boldsymbol{W_k} \cdot \boldsymbol{X}, \quad \boldsymbol{y_{ek}} = \boldsymbol{W_k} \cdot \boldsymbol{Y}. \tag{1}$$

Intuitively, each embedding matrix can compress the vocabulary into a low-dimensional space (Li et al. 2015; Lv et al. 2016). However, because the multiple embedding spaces are constructed based on different contexts in the text corpus, the embedding spaces are different from each other. Thus they together can reveal multifaceted aspects of the words and the word pairs for entailment learning.

Concatenation, Difference, and Binding. The next layer after the contextual representations is designed to arrange the information in the embeddings $\boldsymbol{x_e}$ and $\boldsymbol{y_e}$. To quantify the relationship in the word pair (x, y), we need to combine their information in an appropriate way. Thus we compute the concatenation and the difference in this layer:

$$\boldsymbol{x_{ck}} = \boldsymbol{x_{ek}} \oplus \boldsymbol{y_{ek}}, \quad \boldsymbol{x_{dk}} = \boldsymbol{y_{ek}} - \boldsymbol{x_{ek}}\ . \tag{2}$$

We use these operations based on the following reasons: $\boldsymbol{x_e} \oplus \boldsymbol{y_e}$ can retain all the information of words from the context and measure the antecedent and consequent expressions in word pairs (Baroni et al. 2012). However, word pairs are non-symmetry, which means (*x, y*) and (*y, x*) are two different pairs. Concatenation cannot deal with it. As suggested by (Weeds et al. 2014), $\boldsymbol{y_e} - \boldsymbol{x_e}$ can capture the degree of distributional inclusion on each embedding dimension and the overall difference may capture the direction of the entailment (Weeds et al. 2014).

We further bind each concatenation $\boldsymbol{x_c}$ and difference $\boldsymbol{x_d}$ together using a context specific weight α_k, which is learned during the model training:

$$\boldsymbol{x_k^{'}} = (\alpha_k \cdot \boldsymbol{x_{ck}}) \oplus ((1-\alpha_k) \cdot \boldsymbol{x_{dk}}) \ . \quad (3)$$

Gate Unit. By utilizing attention mechanism, the gate unit layer modulates the flow of information inside the contextual units (Chung et al. 2014) and is the key component of the model. Indeed, vectors from different contexts represent different aspects of words. However, more information may not lead to better performance. If we just sum all these contexts up without differentiation, some irrelevant contexts may comprise the performance. Thus we utilize gate unit to calculate the weights of different contexts, so that the model can focus on relevant contexts. The weights can be optimized by the learning algorithm of CENN. In this way, if one context is more relevant to the given RLE task, its weight will be bigger, which means it will get more attention. Specifically, we define:

$$c_k = \sigma(\boldsymbol{G_k} \cdot \boldsymbol{x_k^{'}} + b_{gk}) \ , \quad (4)$$

here, c_k is the scalar weight of the *k*-th context passing the gate unit. $\boldsymbol{G_k}$ and b_{gk} denote the weights and bias of the *k*-th gate unit. $\sigma(\cdot)$ is the sigmoid function.

Combination and Probability. In the last step of CENN, we combine all contexts' representation vectors together for predicting the entailment relations in word pairs and calculate the probability of entailment relation existing under the condition of a word pair (*x, y*). The formulas are as follows:

$$P(label|\boldsymbol{X}, \boldsymbol{Y}) = \sigma(\boldsymbol{M} \cdot \sum_{k=0}^{K}(c_k \cdot \boldsymbol{x_k^{'}}) + b_m) \ , \quad (5)$$

here, $\boldsymbol{M}$ and b_m are the weights and bias of σ function. We highlight the superiority of our proposed CENN model in the following two aspects:

- Our model is context-enriched with multiple constant embedding matrices $\boldsymbol{W_k}$ for $k = \{1, 2, \cdots, K\}$. Representing words by multiple vectors from different contexts, it can reveal multifaceted aspects of the words due to their homonymy and polysemy. The objective of CENN is to unify the necessary information to predict the entailment relations in word pairs.
- By utilizing the attention mechanism, CENN can pay more attention to the preferable contexts for the word pairs. CENN optimally computes the weights of different contexts instead of simply adding all the contexts naively as done by other methods.

Learning Algorithm

In this subsection, we introduce the learning algorithm of CENN, including its loss function, the initialization of the model and the updating method of parameters.

Loss Function and Parameters. The loss function of CENN is the cross-entropy as follows:

$$L = -\frac{1}{d}\sum_{i=0}^{n}(label \cdot \log(P(label|\boldsymbol{X}, \boldsymbol{Y}))) + (1-label) \cdot \log(1 - P(label|\boldsymbol{X}, \boldsymbol{Y})) \ . \quad (6)$$

As the overall architecture has been proposed, the parameters of CENN are: $\{\boldsymbol{M}, b_m, \boldsymbol{G}, b_g, \alpha\}$. To initialize the model, we randomly set the weights $\boldsymbol{M}$ in Eq.(5) and $\boldsymbol{G}$ in Eq.(4) following the uniform distribution in the range between $-\sqrt{6/(nin+nout)}$ and $\sqrt{6/(nin+nout)}$ as suggested by (Orr and Müller 2003). b_m in Eq.(5) and b_g in Eq.(4) are the bias of sigmoid functions. We initialize both of them to 0. α in Eq.(3) is set as 0.5, meaning that the operations are equal at the beginning. *P*, denoting the dimension of word embeddings, is set as 300.

Updating Method. After all the parameters are initialized, BP algorithm is used to train the model, where the loss function is minimized through stochastic gradient descent (SGD) (Bottou 2010). To be specific, we use "mini-batch" to speed up the training process, in which the batch size can be set from 100 to 300. At the back propagate stage, the learning rate is initialized with one value from 1 to 2. Due to the different characteristics of the datasets, their respective batch size and learning rate might be different. Moreover, in order to avoid overfitting, the learning rate is dynamically updated after a period of iterations (usually 100). We halve the learning rate for every specific number of batches until it reaches the user-specified minimum threshold.

4 Experiments

In this section, we first introduce datasets, different contexts, baselines, as well as the performance metrics for RLE. Then we compare CENN with several baselines, and give a detailed analysis about the experiments.

Datasets Description and Pre-processing. We use 5 labeled datasets for evaluation. In the datasets, each data entry contains a word pair (x, y) and a $label$ indicating whether x entails y. Note that each dataset was created with a slightly different goal in mind, affecting word-pair generation and annotation. For example, both Bless2011 and Baroni2012 datasets were designed to capture hypernyms, while others tried to capture broader entailment relations (e.g. causality) (Levy et al. 2015). Table 1 provides statistics of each dataset. From Table 1, we can find that positive and negative examples in both Baroni2012 and Turney2014 datasets are balanced, while the rest datasets are extremely unbalanced. This phenomenon will affect the performance of models.

Please note that, there is a problem called *lexical memorization*, i.e., *the classifier learns that a specific word in a specific slot is a strong indicator of the label* (Levy et al. 2015). For example, if a model finds plenty of positive examples when word *y* is *bird*, it may treat any examples positive where *y* is *bird* due to memorizing the word *bird*. This leads the model overfitting. In order to overcome this problem, we first randomly split the vocabulary into "train" and "test" words. The word pairs, whose words are only "train" words or "test" words, are called train-only or test-only pairs. Then we extract train-only and test-only subsets of each dataset following (Levy et al. 2015). After this procedure, word overlap between training and test set is avoided, which means memorizing the specific words is useless. Thus the lexical memorization problem is overcome.

Table 1: Summary statistics of five datasets.

Dataset	#Instances	#Positive	#Negative
Kotlerman2010	2,940	880	2,060
Bless2011	14,547	1,337	13,210
Baroni2012	2770	1,385	1,385
Turney2014	1,692	920	772
Levy2014	12,602	945	11,657

Word Representations. The key idea of this study is to use multiple contextual word embeddings to enhance the performance of automatic RLE. Specifically, we use four contexts as follows:

- **Bag-of-Words**. Uses original *word2vec* (Mikolov et al. 2013) implementation. With the commonly used setting, the topical window size is 5, i.e., 5 tokens to each side of the target word (10 context words in total). The negative-sampling parameter is 15, and the words and contexts that appeared less than 100 times are discarded.
- **Dependency-based**. First tags the corpus with parts-of-speech using Stanford Tagger (Toutanova et al. 2003) and then parses the tagged corpus into labeled dependencies (Goldberg and Nivre 2012; Levy and Goldberg 2014). The words and contexts that appeared less than 100 times are discarded.
- **Domain-space**. The contextual patterns are simply the first noun to the left of the given n-gram (if there is one) and the first noun to the right (if there is one). As studied by (Turney 2012; Ran et al. 2015), these nouns characterize the domain or topic of a target word.
- **Function-space**. The function (or role) of a word is characterized by the syntactic context that relates it to the nearby verbs. Following (Turney 2012; Ran et al. 2015), we use 6 contextual patterns to construct the function space and generate the tagged phrases.

We should note that our CENN model is a general and open framework to handle different word contexts and different entailment perspectives.

Baseline. As the classic methods, SVM and LR have achieved brilliant performance in various tasks (Turney and Mohammad 2015). What's more, the effectiveness of combination methods and gate unit layer in CENN are worthy of study too. Thus, we design the sNN and sCENN as baseline methods. The details are shown as follows.

- *SVM*. The words in pair (*x, y*) are represented by a single "prototype" vector respectively as the input. We test two combinations for representing the word pair as a feature vector: concat ($\boldsymbol{x} \oplus \boldsymbol{y}$) (Baroni et al. 2012) and diff ($\boldsymbol{y} - \boldsymbol{x}$) (Roller, Erk, and Boleda 2014). For each composition, we train SVM with a linear kernel or RBF kernel, tuning hyperparameters with a validation set.
- *LR*. We choose the same word representations and combinations which we did in the experiment of SVM to represent (*x, y*) in LR. For each composition, we train LR with L_1 or L_2 regularization.
- *sNN*. We design a simple neural network to integrate the multiple vector representations of x and y. As is shown in Fig.2, it merges multiple vectors from different contexts of x and y respectively, and then concatenates the combined x and y. Compared with CENN, sNN does not have the different combined methods and gate unit layer for integrating multiple vectors of words.
- *sCENN*. In order to measure the effectiveness of gate unit layer, we design a sCENN model as another baseline. This model also utilizes multiple vector representations as the input. The main difference between sCENN and CENN is that CENN has the gate unit layer, which is shown in Fig.1, to weight the contexts precisely, while sCENN just treats them as a simply weighted sum.

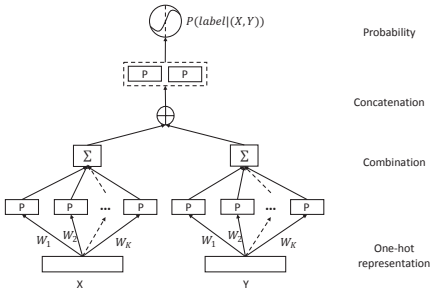


Figure 2: The sNN structure.

Performance Metrics. We apply the fitted model on testing data and compute following performance metrics. Here, c_{ij} means the number of word pairs that are actually in class i and predicted in class j. Pre^l, Rec^l, F^l are the precision, recall and $F1$-value of the class l where $i, j, l \in \{0, 1\}$

$$\begin{aligned} w^0 &= (c_{00} + c_{01})/(c_{00} + c_{01} + c_{10} + c_{11}) \ , \\ w^1 &= (c_{11} + c_{10})/(c_{00} + c_{01} + c_{10} + c_{11}) \ , \\ Pre &= w^0 \cdot Pre^0 + w^1 \cdot Pre^1 \ , \\ Rec &= w^0 \cdot Rec^0 + w^1 \cdot Rec^1 \ , \\ F1 &= w^0 \cdot F1^0 + w^1 \cdot F1^1 \ . \end{aligned} \tag{7}$$

Different from (Levy et al. 2015), we take both positive ($label = 1$) and negative ($label = 0$) instances into consideration, which will be helpful to avoid model overfitting. Thus we set the average $F1$-value (Turney and Mohammad 2015) as the evaluation standard in our experiment. With this evaluation standard, we can get a more appropriate evaluation of CENN on RLE task.

Results and Analysis

In this section, we give a detailed analysis about the results and discuss the sensitiveness of parameters and contexts.

Overall Results. The overall experimental results are summarized in Table 2. We can see that CENN outperforms the other models in four of the five datasets. As we expected, CENN not only utilizes a variety of representations for words to capture as many aspects of words as possible, but also pays close attention to the contexts which are crucial to the inference of word pairs. By taking advantage of attention mechanism, CENN can learn to recognize lexical

Table 2: Overall performance on RLE task with $F1$-value

Methods		Kotlerman 2010	Bless 2011	Baroni 2012	Turney 2014	Levy 2014
LR	concat	62.9	74.4	52.9	53.4	83.9
	diff	63.7	84.0	62.1	53.1	81.6
SVM	concat	60.4	91.3	71.8	**69.6**	72.4
	diff	55.8	92.5	77.4	66.3	69.5
sNN		56.8	74.0	60.6	41.6	88.4
sCENN		64.1	93.0	71.3	66.8	88.7
CENN		**65.7**	**94.3**	**78.6**	69.4	**89.6**
Improvement		+1.6	+1.3	+1.2	-0.2	+0.9

entailment precisely, which also proves that the relations in word pairs always depend on their appropriate contexts.

Compared with CENN, SVM and LR just utilize one aspect of words. Thus they cannot handle the homonymy and polysemy of words. E.g., *book* has the meanings *"a collection of sheets of papers"* and *"to reserve (something) for future use"*, which are far different from each other. sNN just puts all contexts information together without differentiating, so that it cannot pick up the proper contexts for RLE. As for sCENN, whose architecture is similar to CENN's, preforms better than other methods except CENN. These results illustrate that the concatenation, difference and binding layer is effective. However, due to the lack of gate unit layer, sCENN cannot pay attention to the important contexts as precisely as CENN does.

As mentioned before, the convergence conditions of five datasets are the same, which is performing best in validate sets. However, we find that our model does not perform well in Turney2014 dataset, i.e., not performing well in test set. After further analysis, we find that there are too many types of semantic relations and few instances in this dataset, which causes the model more iterations and may lead the model a little overfitting. Furthermore, we observe that almost all the models do not perform well in Turney2014 dataset, which means the dataset may not be suitable for RLE task.

Sensitivity of Parameters. There is one parameter in CENN to be determined: the threshold θ, which is used to predict whether the input is positive or negative. To be specific, we evaluate the Precision, Recall and $F1$-value (Turney and Mohammad 2015) with different values of θ on each dataset. The results are shown in Fig.3. For Fig.3(a), 3(b) and 3(c), we can find that with the increasing of threshold θ, the trend of $F1$-value is increasing since these three datasets are unbalanced. The proportions of positive and negative examples in Bless2011 and Levy2014 datasets are even *1:10*. As a result, when threshold θ is increasing, we can determine the input a negative example with higher probability.

In Fig.3(d) and 3(e), the trend of $F1$-value increases at first then decreases. It reaches the highest point when threshold θ is close to 0.5. As mentioned before, both of these two datasets are balanced. The difference between them is that Turney2014 has more types of relations than Baroni2012. As a result, we can find that the precision of Turney2014 dataset has a very abnormal changing trend, i.e., when the threshold θ gets higher, the precision is also becoming higher, which may also indicate that Turney2014 dataset is not suitable for recognizing lexical entailment task.

Sensitivity of Contexts. The experimental results in Table 2 indicate that simply increasing the number of types of the word contexts does not improve the performance for RLE, so that the impact of number of the context types on CENN needs to be further discussed. A qualitative way of analyzing this problem is to check the weights of contexts learned in CENN. Thus we measure the weights that gate unit layer has calculated. Fig.4 shows the weights of contexts in each dataset.

These contexts induce different aspects of words. As we can see, all of the contexts contribute to the task, while their proportions may be different. For the contexts that have obviously larger weights, we call it the *dominate context*. Among all the datasets, we observe that Bless2011 has dominate contexts *dependency* and *function-space*, and Levy2014's is *function-space*. We want to figure out whether CENN can get proper results with only dominate contexts. Thus, we make additional experiments with only dominate contexts to verify this problem. The result of the $F1$-value with only dominate contexts is 93.7% in Bless2011 dataset and is 88.7% in Levy2014 dataset, which are a little worse than overall results. These phenomena prove that all of the contexts are very important though the degree of importance is different. In order to explain this phenomenon more clearly, we pick up one word pair for case study.

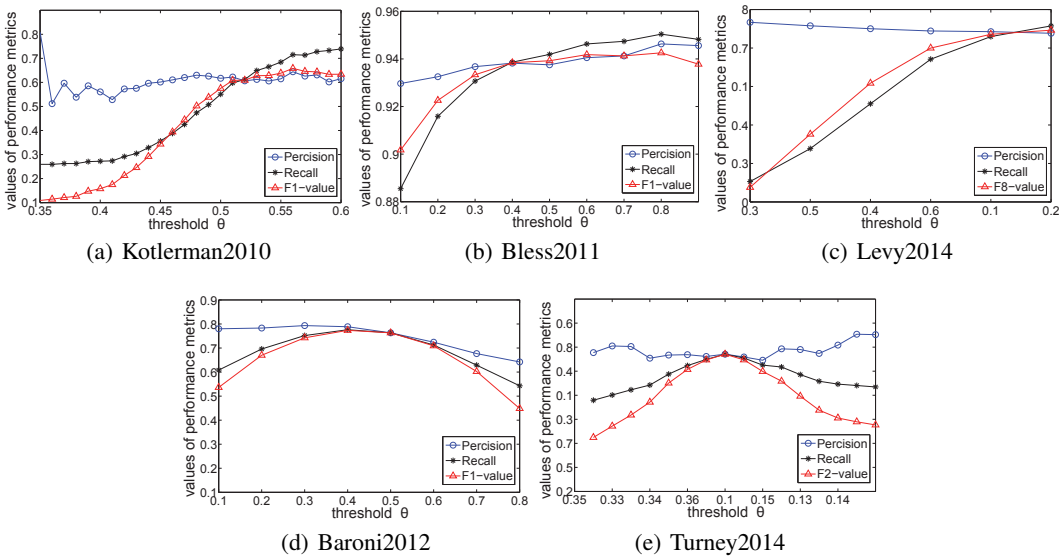
Case Study. We choose the word pair (*chicken, solid, True*) from Baroni2012 dataset for case study. We can find that Bag-of-Words contexts and Domain-space contexts have higher weights than the rest contexts from Fig.5. At the first sight, we may believe that there is no relation between *chicken* and *solid*. Fortunately, we find that *solid* has an explanation that *"food which is not liquid-based"*, where *chicken* has the meaning: *"The flesh of a chicken used for food"*. Therefore, there is the entailment relation between the word pair under the topical measurements, and CENN does find this kind of relation by paying more attention to Bag-of-Words contexts and Domain-space contexts. Thus, CENN has actually learned to recognize lexical entailment.

5 Related work

The related work in this section can be classified into two parts: the methods about RLE and typical applications of neural network in RTE tasks.

Recognizing Lexical Entailment. To the best of our knowledge, the first RTE Challenge (Dagan, Glickman, and Magnini 2006) took place in 2005 and it had been a regular event since then. Nowadays, most of the RTE systems have included a module for RLE (Dagan et al. 2010; Herrera, Penas, and Verdejo 2006). The early RLE solutions depended on symmetric similarity measures. However, it was understood that entailment is inherently asymmetric and any symmetric measure could make only a rough approximation (Geffet and Dagan 2005).

Due to the drawbacks of symmetric similarity measures, methods with specific assumptions were proposed. Weeds et al. (2003) introduced a balanced combination of the asymmetric APinc measure (Kotlerman et al. 2010) and the symmetric LIN measure (Lin 1998). Another typical approach



(a) Kotlerman2010 (b) Bless2011 (c) Levy2014

(d) Baroni2012 (e) Turney2014

Figure 3: Performance ($F1$-value) of CENN with different threshold parameters.

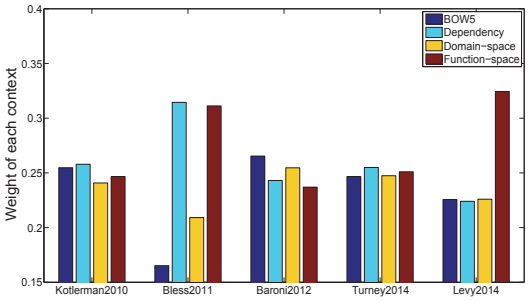


Figure 4: Weight of each context in five datasets

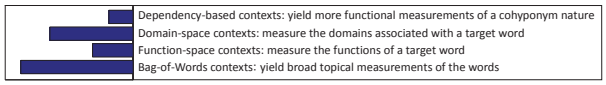


Figure 5: Weights of contexts of the word pair

was called ConVecs algorithm (Baroni et al. 2012). Even though these methods could sometimes achieve better performance, they deeply depended on assumptions, which meant they were not flexible enough to a general situation.

Hence, methods based on word context were proposed. In (Turney and Mohammad 2015), two different word-context matrices i.e., a domain matrix D and a function matrix F (Turney 2012), were involved to generate a larger and more varied set of features for supervised learning algorithms. Unfortunately, Levy et al. (2015) suspected that these state-of-the-art methods above did not actually learn to recognize the relation between two words based on their investigating. Moreover, we also find that almost all previous methods preferred to represent a word with only one vector, while single vector representation is problematic because of words' polysemous (Reisinger and Mooney 2010; Huang et al. 2012), e.g., the word *blue* can represent a color as well as an emotion. Thus, context based methods still need to be further improved.

Neural Network Methods for RTE Tasks. Neural networks, especially deep neural networks have become attractive models in RTE. Bowman et al. (2015) provide a large annotated corpus for RTE, which attracted researchers' attention. Then plenty of works for RTE have been done on this dataset, such as LSTM encoders (Bowman et al. 2016), Tree-based CNN encoders (Mou et al. 2016), and LSTMN with deep attention fusion (Cheng, Dong, and Lapata 2016). These successful applications indicate that neural networks are powerful and robust in modeling text with complicated relationships in a deep level, while few of them are used to deal with RLE. Thus the method of integrating multi-context by neural network for RLE is worthy of studying.

6 Conclusion and Future Work

In this paper, we propose a Context-Enriched Neural Network (CENN) model for RLE. In order to capture as much information about the words as possible for entailment learning, we utilize multiple representations from different contexts for a word pair, and we use attention mechanism to optimize the weights of the information from different contexts to predict the entailment relations in the word pairs. We also illustrate that CENN is a flexible and open framework applicable to datasets with different number of contexts. Experimental results on five real-world datasets indicate that our approach could effectively recognize lexical entailment with significantly improved performances in comparison with state-of-the-art methods.

In the future, we will adopt more different context-based embeddings for the representations of words, so that CENN can obtain ample information about the words and become

much more effective for recognizing lexical entailment.

7 Acknowledgements

This research was partially supported by grants from the National Key Research and Development Program of China (Grant No. 2016YFB1000904), the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010) and the Fundamental Research Funds for the Central Universities of China (Grant No. WK2350000001). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299).

References

Androutsopoulos, I., and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *JAIR* 135–187.

Baroni, M.; Bernardi, R.; Do, N.-Q.; and Shan, C.-c. 2012. Entailment above the word level in distributional semantics. In *EACL*, 23–32.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer. 177–186.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Bowman, S. R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C. D.; and Potts, C. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Cheng, J.; Dong, L.; and Lapata, M. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dagan, I.; Dolan, B.; Magnini, B.; and Roth, D. 2010. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering* 16(01):105–105.

Dagan, I.; Glickman, O.; and Magnini, B. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges.* Springer. 177–190.

Fu, R.; Guo, J.; Qin, B.; Che, W.; Wang, H.; and Liu, T. 2014. Learning semantic hierarchies via word embeddings. In *ACL (1)*, 1199–1209.

Geffet, M., and Dagan, I. 2005. The distributional inclusion hypotheses and lexical entailment. In *ACL*, 107–114.

Goldberg, Y., and Nivre, J. 2012. A dynamic oracle for the arc-eager system. In *Proc. of COLING*.

Herrera, J.; Penas, A.; and Verdejo, F. 2006. Textual entailment recognition based on dependency analysis and wordnet. In *Machine Learning Challenges.* Springer. 231–239.

Hua, W.; Wang, Z.; Wang, H.; Zheng, K.; and Zhou, X. Understand short texts by harvesting and analyzing semantic knowledge.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, 873–882.

Kotlerman, L.; Dagan, I.; Szpektor, I.; and Zhitomirsky-Geffet, M. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(04):359–389.

Levy, O., and Goldberg, Y. 2014. Dependency-based word embeddings. In *ACL (2)*, 302–308.

Levy, O.; Remus, S.; Biemann, C.; Dagan, I.; and Ramat-Gan, I. 2015. Do supervised distributional methods really learn lexical inference relations. *Proceedings of NAACL, Denver, CO*.

Li, Y.; Xu, L.; Tian, F.; Jiang, L.; Zhong, X.; and Chen, E. 2015. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *IJCAI*, 25–31.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *COLING-Volume 2*, 768–774.

Lv, G.; Xu, T.; Chen, E.; Liu, Q.; and Zheng, Y. 2016. Reading the videos: Temporal labeling for crowdsourced time-sync videos based on semantic embedding. In *Thirtieth AAAI Conference*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mou, L.; Men, R.; Li, G.; Xu, Y.; Zhang, L.; Yan, R.; and Jin, Z. 2016. Natural language inference by tree-based convolution and heuristic matching. In *The 54th ACL*, 130.

Orr, G. B., and Müller, K.-R. 2003. *Neural networks: tricks of the trade*. Springer.

Ran, C.; Shen, W.; Wang, J.; and Zhu, X. 2015. Domain-specific knowledge base enrichment using wikipedia tables. In *ICDM 2015*, 349–358.

Reisinger, J., and Mooney, R. J. 2010. Multi-prototype vector-space models of word meaning. In *HLT-NAACL*, 109–117.

Roller, S.; Erk, K.; and Boleda, G. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING*, 1025–1036.

Santus, E.; Lenci, A.; Lu, Q.; and Im Walde, S. S. 2014. Chasing hypernyms in vector spaces with entropy. In *EACL*, 38–42.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL-Volume 1*, 173–180.

Turney, P. D., and Mohammad, S. M. 2015. Experiments with three approaches to recognizing lexical entailment. *NLE* 21(03):437–476.

Turney, P. D. 2012. Domain and function: A dual-space model of semantic relations and compositions. *JAIR* 533–585.

Weeds, J., and Weir, D. 2003. A general framework for distributional similarity. In *EMNLP*, 81–88.

Weeds, J.; Clarke, D.; Reffin, J.; Weir, D. J.; and Keller, B. 2014. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, 2249–2259.