A Dynamic Word Representation Model Based on Deep Context

Xiao Yuan[1], Xi Xiong[2(✉)], Shenggen Ju[1], Zhengwen Xie[1], and Jiawei Wang[1]

[1] College of Computer Science, Sichuan University, Chengdu 610065, China
[2] School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China
flyxiongxi@gmail.com

Abstract. The currently used word embedding techniques use fixed vectors to represent words without the concept of context and dynamics. This paper proposes a deep neural network CoDyWor to model the context of words so that words in different contexts have different vector representations of words. First of all, each layer of the model captures contextual information for each word of the input statement from different angles, such as grammatical information and semantic information, et al. Afterwards, different weights are assigned to each layer of the model through a multi-layered attention mechanism. At last, the information of each layer is integrated to form a dynamic word with contextual information to represent the vector. By comparing different models on the public dataset, it is found that the model's accuracy in the task of logical reasoning has increased by 2.0%, F1 value in the task of named entity recognition has increased by 0.47%, and F1 value in the task of reading comprehension has increased by 2.96%. The experimental results demonstrate that this technology of word representation enhances the effect of the existing word representation.

Keywords: Word representation · Attention mechanism · Logical reasoning · Named-entity recognition · Reading understanding

1 Introduction

The NLP (natural language processing) system available based on deep learning usually first converts the text input into a vectorized word representation [1–4], i.e. the word embedding vector for further processing. Researchers have proposed a large number of embedding methods to encode words and sentences into dense fixed vectors, which tremendously enhance the ability of neural networks to process textual data. The most commonly used word embedding methods include word2vec [5], FastText [6] and GloVe [7] and so. Studies have demonstrated that these word embedding methods can significantly improve and simplify a great number of text processing applications [8, 9].

However, at present, the commonly used word embedding techniques consider no of context and dynamics, which all regard words as fixed atomic units, and represent words by using the indices of word lists or fixed values in the pre-trained word vector matrix. Although the embedding methods that assign fixed values to represent words

J. Tang et al. (Eds.): NLPCC 2019, LNAI 11839, pp. 1–10, 2019.
https://doi.org/10.1007/978-3-030-32236-6_60

are simple, yet they limit their effectiveness in many tasks [10]. In complex natural language processing tasks such as sentiment analysis, text categorization, speech recognition, machine translation and reasoning, et al., dynamic word representations with contextual meaning are needed, namely the same word has different representation vectors in different contexts. For instance, in the sentences: "A plant absorbs water from the soil by its roots" and "There is a big amount of water in what he has said", the meanings of "water" are different. If a pre-trained word vector is used, the word "water" in both sentences can solely be represented by the same word vector. In order to solve the above problem, this paper proposes a dynamic word representation model based on deep context, which is a multi-layer deep neural network. Each layer captures the contextual information for each word of the input statement from different views [10], including grammatical information and semantic information, et cetera and then assigns different weights to each layer of the neural network through a multi-layered attention mechanism, and finally integrates the information of each layer to form a vectorized representation of the word.

2 Related Work

A word embedding model based on the shallow Neural Network Language Model (NNLM) can convert words into continuous vectors [11, 12]. At present, the mainstream word embedding models include CBOW, Skip-Gram, FastText and GloVe models, of which CBOW and Skip-Gram belong to the renowned word2vector framework. The leading improvement of FastText compared with the original word2vec is that it has introduced n-grams. GloVe is a word representation model based on overall word frequency statistics, which makes up for the lack of word2vec that doesn't consider the overall co-occurrence information of words. Experiments have proven that the word vector generated by GloVe model is better under quite a few scenarios [7]. However, both the word2vec model and the GloVe model are too simple and are limited by the representational capacity of the shallow model used (typically 3 layers).

The word representation model MT-LSTM [13] that is based on machine-translation model utilizes the Encoder-Decoder framework to pre-train the corpus for machine translation and extract the Embedding layer and the Encoder layer of the model. And then it designs a model based on the new task, and uses the output of the trained Embedding layer and Encoder layer as the input of this new task model, and lastly does the training under the new task scenario. However, this machine translation model needs a large amount of supervised data, meanwhile, the Encoder-Decoder structure limits the model to capture some semantic information.

The word representation model based on depth NNLM is generally better than shallow NNLM. Peters et al. [10] proposed a renowned model ELMo, which used the internal state of multi-layer BiLSTM (Bi-directional Long Short-Term Memory) to generate word vectors. Compared to N-gram models, word2vec models, and GloVe [10, 14], it has a better effect. However, ELMo is limited by BiLSTM's serial computer system and feature extraction capabilities, which limits its application under a good number of scenarios.

The method in this paper adopts the deep NNLM, meanwhile, it takes the idea of generating the word vector from the internal state of NNLM in ELMo, and replaces the BiLSTM encoder in the model with the Transformer encoder with concurrent computing and contextual coding capability, and introduces a multi-layer attention mechanism, blending word representation information of different levels in neural network, and generating the word vector with contextual meanings.

3 Proposed Approach

See Fig. 1.

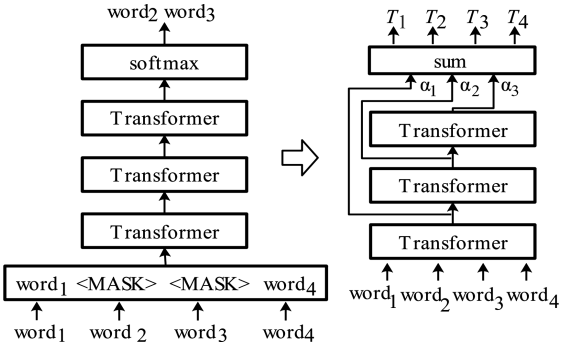

Fig. 1. Overall framework

3.1 Overall Framework

The overall framework of the dynamic context representation model proposed in this paper is illustrated in Fig. 1. It consists of two main processes: (1) the masked language model on the left of Fig. 1 (see Sect. 3.2 for details); (2) the Transformer layer in the pre-trained masked language model is extracted and a new output layer is added to form the model of this paper—the deep contextual dynamic word representation model (on the right side of Fig. 1), which blends multiple outputs of Transformer layer through a multi-layered attention mechanism, generating a deep dynamic word representation vector (see Sect. 3.3 for details). The Transformer layer of the main structure in the framework is made up of a Transformer encoder, which is a two-way contextual information extractor (see Sect. 3.4 for details).

3.2 Masked Language Model

The language model encodes the sequence of words by means of a distributed representation. The objective function of a general language model is a log-likelihood function of the probability of occurrence of all word sequences in the corpus, such as the popular CBOW, Skip-Gram, FastText, and GloVe models, et cetera. In this paper, we utilize the masked language model composed of Transformer encoder that can capture the contextual information of words in sentences to extract textual information

[16]. Unlike the general language models, the objective function of the masked language model is the log-likelihood function of the probability of occurrence of all masked words in the corpus, which is:

$$L(C) = \frac{1}{n}\sum_{i=1}^{n}\sum_{w_k \in Mask_i} \log P(w_k|context_i - Mask_i) \tag{1}$$

In it, *Mask* is a set made up of words that are masked in a word sequence *context*. n is the number of word sequences in the corpus. The model predicts *Mask* to the best of its ability in accordance with the remaining words. The whole process of prediction is similar to an English cloze test. As shown on the left side of Fig. 1, in the masked language model, the input word sequence *context* is first represented as a vector made up of the word sequence: $c = [word_1, word_2, \ldots, word_t]$, and then some words are blocked in the input word sequence *context* to obtain the word sequence that is partly masked: $u = [word_1, <MASK>, \ldots, word_t]$. Then, the information of the input word sequence is extracted by the multi-layer Transformer encoder, and lastly the $P(w_k|context_i - Mask_i)$ value is calculated using the normalized exponential function. The entire calculation process is shown in Eq. (2):

$$\begin{aligned} u &= MASK(c) \\ h_0 &= uW \\ h_i &= Transformer(h_{h-1}), \forall i \in [1, L] \\ P(w_k|context_i - mask_i) &= softmax(h_L M) \end{aligned} \tag{2}$$

In it, *MASK(c)* represents the masked operation for some words in the word sequence c, W and M denote the weight matrix, Transformer indicates that the Transformer encoder extracts the information of the input word sequence, and L means the number of layers of the Transformer encoder. Softmax is a normalized exponential function, which converts the input into a probability distribution.

3.3 Multi-layered Attention Mechanism

The right side of Fig. 1 is the model structure chart of deep contextualized dynamic word representations (CoDyWor). In the figure, *word* denotes the input words, Transformer is the encoder, α is the weight of different layers, and *sum* demonstrates the summation of information captured by Transformer of different layers. T is the generated word representation. CoDyWor is stacked by the Transformer encoder with a multi-layered attention mechanism. The model merely retains the Transformer layer of masked language model (keeping the knowledge that the masked model acquires on the dataset) and then adds a new output layer.

A CoDyWor with an L-layer Transformer generates L types of word representations for each input *word*: $\{h_1, h_2, \ldots, h_L\}$. Under the simplest scenario, CoDyWor directly uses the output of the last layer of Transformer as words' contextualized word representation, ie, $CoDyWor(word) = h_L$. Since different levels of Transformer can capture different types of information [17], multiple layers of attention mechanism can

be used to give different layers of Transformer different weights $\alpha_1, \alpha_2, \ldots \alpha_T$. The formula for the CoDyWor word representation is as follows:

$$CoDyWor(word) = \beta \sum_{j=1}^{L} \alpha_j h_j, where \begin{cases} \alpha_1 + a_2 + \ldots + a_L = 1 \\ a_j \geq 0 \end{cases}, \beta \neq 0 \quad (3)$$

In it, h_j and a_j are respectively the output vector and the corresponding weight of the Transformer encoder of the j layer, β is a scaling parameter, and both α and β are automatically adjusted by the stochastic gradient descent algorithm. α is guaranteed by the Softmax layer to satisfy the probability distribution. Adding the β parameter mainly adjusts the norm of word representation vector generated by the model to represent the norm of the vector [10], which is convenient for model training.

3.4 Transformer Encoder

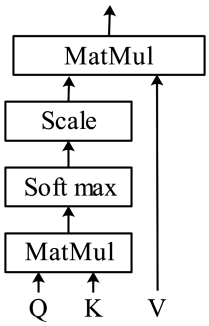

Fig. 2. Multi-head dot-product attention mechanism

The calculation diagram of Transformer encoder's multi-head scaling dot-product attention mechanism is shown in Fig. 2, in which MatMul represents matrix multiplication, *Softmax* represents normalized exponential operation, and Scale denotes scaling vector operation. The Transformer encoder duplicates the input three times, which means that the three input contents are the same. Here, Q, K, and V respectively denote of query, key, and value. First, through the query of the key, it is calculated that different keys should be given different weights, and then the values corresponding to the keys are taken out and the values are added based on the weights to form an output, the times of repeating this process is called the number of Transformer headers. The query q, the key k, and the value v are all d-dimensional. The Transformer multi-head scaling dot-product attention mechanism calculates as follows: (1) calculate the dot-product result of q and k, and then divide the result by a constant $\sqrt{d}$; (2) softmax function converts the result into probability value; (3) use probability value dot product v to obtain scaling dot-product attention operational input. In order to improve the computational efficiency, a set of queries q are put together into a matrix Q, and then

the attention function is applied to a set of queries at the same time, and likewise, the key k and the corresponding value v are also placed in the matrices K and V respectively. The multi-head scaling dot-product attention formula in the entire Transformer encoder is as follows:

$$Attention(Q, K, V) = soft\max(\frac{QK^T}{\sqrt{d}})V \tag{4}$$

4 Experiments

We evaluate the proposed CoDyWor model on three public datasets, MultiNLI, CoNLL03 and SQuAD.

4.1 Datasets and Experimental Settings

The datasets of natural language inference, named entity recognition and reading comprehension tasks are multi-domain logical reasoning dataset MultiNLI [18] with more than 430,000 pieces of data, named entity dataset CoNLL03 [19] containing more than 10,000 pieces of data, and Stanford reading comprehension dataset SQuAD [20] with more than 100,000 pieces of data. In order to evaluate the effect of word representational learning, the accuracy is used as the evaluative metric in the logical reasoning task, and the F1 value is used as the evaluative criteria in the named entity recognition and reading comprehension tasks. The higher the accuracy rate or the F1 score is, the better the model's effect is. The hyperparameters of the model are set as follows: the length of maximum input sentence is 128, the number of training batches is 32, the learning rate is 2e-5, and the number of epochs is 6. In addition, in order to ensure the stability of the results, the experiment is to be repeated 10 times, and the average value is adopted as the final predicted result of the model.

4.2 Logical Reasoning

In this section an experiment was carried out on the public MultiNLI dataset. MultiNLI is one of the largest corpora in logical reasoning tasks. The fields of corpus include speeches, letters, novels, and government reports, et cetera. In the logical reasoning task, MultiNLI-A is used to indicate that the data of both the training set and the test set are from the same domain, and MultiNLI-B is used to show that the data of both the training set and the test set are from the different domains. The requirement of the MultiNLI dataset is to predict a given pair of (premise, hypothesis) sentences to determine whether the hypothesis sentence is implicit, contradictory, or neutral in relation to the premise sentence.

The experimental results are shown in Table 1. The model CoDyWor proposed in this paper is apparently better than the enhanced sequence reasoning model GloVe_ESIM using GloVe word representation and the CoVe_BiLSTM and ELMo_-BiLSTM models using CoVe and ELMo word representation, indicating that the word

representation with contextual meaning can improve the accuracy of logical reasoning tasks. Compared to the model GPT_Transformer [15] using the one-way Transformer, the accuracy of CoDyWor (using the two-way Transformer encoder) has increased by 2.0% (on the MultiNLI-A test set) and 2.3% (on the MultiNLI-B test set). It means that the contextual text information obtained from both directions at the same time is richer than the text information obtained from one direction, which is more conducive to the model to further comprehend the meaning of the text.

Table 1. Comparison of accuracy in logical reasoning (MultiNLI datasets, %)

Method	MultiNLI-A	MultiNLi-B
GloVe_ESIM	72.3	72.1
CoVe_BiLSTM [22]	71.6	71.5
ELMo_BiLSTM [10]	76.9	76.7
GPT_Transformer [15]	82.1	81.4
CoDyWor	84.1	83.7

4.3 Named Entity Recognition

The experiment was carried out on the famous named entity identification dataset CoNLL03 in this section. The task of the CoNLL03 dataset is to identify four named entities in the sentence: people, places, organizations, and miscellaneous items (not belonging to the first three entities).

The experimental results are shown in Table 2. The F1 value of the model CoDyWor proposed in this paper is 92.69%, which is 1.48% and 0.99% respectively higher than the GloVe_BiLSTM model and the CoVe_BiLSTM model using the popular GloVe and CoVe word representation. Compared with the ELMo_BiLSTM model using ELMo word representation, it has increased by 0.47%, which reveals that the use of deep contextual dynamic word representation can enhance the F1 value of the named entity recognition task.

4.4 Reading Comprehension

In this section the experiment was carried out on the famous Stanford reading comprehension dataset SQuAD. Given a question and a paragraph from Wikipedia that contains the answer to this question, SQuAD's task is to find out the range where the answer to the question lies in the paragraph.

The experimental results are shown in Table 3. The F1 value of the model CoDyWor proposed in this paper is 88.76%, which is 2.96% higher than the ELMo_BiLSTM model using the ELMo word representation. Meanwhile, it is also superior to GloVe_BiLSTM, which uses GloVe word embedding and uses a special model structure (simulating multi-step reasoning in machine reading comprehension), indicating that using the word representation with contextual meanings can simply achieve fairly good results without designing a special model structure targeted on reading comprehension task.

Table 2. Comparison of F1 values in named entity recognition (CoNLL03 dataset, %)

Method	F1
GloVe_BiLSTM	91.21
CoVe_BiLSTMt	91.70
ESIM [3]	92.22
ELMo_BiLSTM [10]	92.69
CoDyWor	79.6

Table 3. Comparison of F1 value in reading comprehension (SQuAD dataset, %)

Method	F1
GloVe_BiLSTM	81.10
CoVe_BiLSTM	82.56
ELMo_BiLSTM [10]	85.80
CoDyWor	88.76

5 Ablation Research

In this section ablation experiments were performed on CoDyWor's multi-layered attention mechanism and Transformer encoder to analyze the effects of these two modules in the CoDyWor model.

Table 4. Effect of multi-layered attention mechanism (SQuAD dataset, %)

Layers	T1:F1	T2:F1
First layer	77.63	77.44
Last layer (twelfth layer)	77.96	77.79
Three layers (ahead)	85.15	84.96
Three layers (behind)	85.25	85.08
Six layers (ahead)	88.56	88.37
Six layers (behind)	88.64	88.47
All layers	88.76	88.56

Table 5. Effect of the size of Transformer (MultiNLI-A dataset, %)

Layers	Head	Acc
3	3	76.4
3	12	77.6
6	3	80.3
6	12	81.6
12	12	84.1
12	16	84.4

5.1 Effect of Multi-layered Attention Mechanism

Experiments were performed on the SQuAD dataset to analyze the effects caused by the number of layers (Transformer number) of the multi-layered attention mechanism of the CoDyWor model, the position of the attention layer, and the regularization parameter β. The results are illustrated in Table 4, in which the first column "Layers" indicates that the multi-layered attention mechanism is applied to different layers, the second column T1 represents the use of regularization parameter β, and the third column T2 indicates that the regularization parameters are not used. "Ahead" indicates the output of the first layer of the multi-layered neural network, and "behind" demonstrates the output of the last layer of the neural network. Three rules can be found: (1) The effect of the model is significantly improved with the increase of the number of layers of attention; (2) When the number of layers is the same, the words output from the lower layer Transformer represent better vector effects, especially when the number of layers is small, the differences are obvious; (3) The regularization parameter β can be used to increase the model's F1 value by around 0.19%.

5.2 Effect of the Size of Transformer

Experiments were carried out on the MultiNLI dataset to analyze the impact of the number of different Transformer layers and the number of self-attention heads in the Transformer adopted by the CoDyWor model on the inference accuracy. The experimental results are shown in Table 5. It can be found that increasing the number of layers of Transformer within a certain range or increasing the number of self-attention heads in Transformer can both improve the inference accuracy of the model.

6 Conclusion

This paper proposes an efficient, simply-structured deep contextual dynamic word representation model CoDyWor that can be widely used in natural language processing tasks. The contextual dynamic word representation generated by the model can be used for natural language processing tasks such as logical reasoning, named entity recognition and reading comprehension and so forth, and may be universally utilized to a certain extent. The contextual dynamic word representation generated by the CoDyWor model in the above tasks performs better than the current mainstream static word representations.

Acknowledgements. The work was partially supported by the China Postdoctoral Science Foundation under Grant No. 2019M653400; the Sichuan Science and Technology Program under Grant Nos. 2018GZ0253, 2019YFS0236, 2018GZ0182, 2018GZ0093 and 2018GZDZX0039.

References

1. Hashimoto, K., Xiong, C., Tsuruoka, Y., et al.: A joint many-task model: growing a neural network for multiple NLP tasks. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1923–1933. Association for Computational Linguistics, Copenhagen (2017)
2. Bowman, S.R., Potts, C., Manning, C.D.: Recursive neural networks can learn logical semantics. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 12–21. Association for Computational Linguistics, Beijing (2015)
3. Nallapati, R., Zhou, B., Gulcehre, C., et al.: Abstractive Text Summarization Using Sequence-To-Sequence RNNs and Beyond, pp. 280–290. Association for Computational Linguistics (2016)
4. Xiong, C., Merity, S., Socher, R.: Dynamic memory networks for visual and textual question answering. In: International Conference on Machine Learning, pp. 2397–2406 (2016)
5. Mikolov, T., Chen, K., Corrado, G., et al.: Efficient estimation of word representations in vector space. CoRR, abs/1301.3781 (2013)
6. Bojanowski, P., Grave, E., Joulin, A., et al.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
7. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)

8. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 384–394. Association for Computational Linguistics (2010)
9. Collobert, R., Weston, J., Bottou, L., et al.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
10. Peters, M.E., Neumann, M., Iyyer, M., et al.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237. Association for Computational Linguistics, Stroudsburg (2018)
11. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
12. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
13. McCann, B., Bradbury, J., Xiong, C., et al.: Learned in translation: contextualized word vectors. In: Advances in Neural Information Processing Systems, pp. 6294–6305 (2017)
14. Devlin, J., Chang, M.W., Lee, K., et al.: Bert: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805 (2018)
15. Radford, A., Narasimhan, K., Salimans, T., et al.: Improving language understanding by generative pre-training (2018). https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf
16. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
17. Yosinski, J., Clune, J., Bengio, Y., et al.: How transferable are features in deep neural networks?. In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
18. Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 1112–1122. Association for Computational Linguistics, New Orleans (2018)
19. Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition, pp. 142–147. Association for Computational Linguistics (2003)
20. Rajpurkar, P., Zhang, J., Lopyrev, K., et al.: Squad: 100,000+ Questions for Machine Comprehension of Text, pp. 2383–2392. Association for Computational Linguistics (2016)