Beyond Word2Vec: Embedding Words and Phrases in Same Vector Space

Vijay Prakash Dwivedi
CSED, MNNIT Allahabad
Allahabad, UP 211004, India
mail@vijaydwivedi.com.np

Manish Shrivastava
LTRC, IIIT Hyderabad
Hyderabad, TS 500032, India
m.shrivastava@iiit.ac.in

Abstract

Word embeddings are being used for several linguistic problems and NLP tasks. Improvements in solutions to such problems are great because of the recent breakthroughs in vector representation of words and research in vector space models. However, vector embeddings of phrases keeping semantics intact with words has been challenging. We propose a novel methodology using Siamese deep neural networks to embed multi-word units and fine-tune the current state-of-the-art word embeddings keeping both in the same vector space. We show several semantic relations between words and phrases using the embeddings generated by our system and evaluate that the similarity of words and their corresponding paraphrases are maximized using the modified embeddings.

1 Introduction

Vector embeddings in computational linguistics is a model that encodes words in a vector space. These vector encodings are used in mathematical models and serve as a base for computation in NLP.

Development of word embedding technique started in 2000 when Bengio et al. built neural probabilistic language models to reduce the high dimensionality of word representations in contexts by learning a distributed representation for words (Bengio et al., 2003). After that, continuous research has been done in the field resulting in remarkable improvements in word vector representations as well as the methods of learning the embeddings (Mikolov et al., 2013a; Mikolov et al., 2013b; Pennington et al., 2014). The primary reason for the increase in quality and performance of word vector embeddings is the huge growth of data and and development in computational capabilities as of today.

Natural language has both single word and multi-word units. If we want vector semantics to be near perfect, we need to embed multi-word units with the same quality as we do with the single word units. Improvements in phrase representation will eventually help the areas of question answering, search and translation. For a phrase that is similar to a certain word, the embedding of both the word and phrase should be similar and should lie in the same space. Only then a manipulation on a word and its paraphrase embedding can prove them to be similar.

Currently, compositional models are used to build phrase embeddings with less emphasis on building the compositions using deep learning and more using specific composition functions. Our major contribution in this work is employing deep neural architectures to transform constituent word embeddings of a multi-word units into its vector representation. We build a Siamese deep neural network architecture (Siamese LSTM, to be precise) that accepts two inputs, one being a word while another a phrase. The energy function in the Siamese network measures the similarity between these two input units. In the course of training the network, baseline word embeddings (Section 5.2) are modified and phrase embeddings are generated. We describe the model in detail in further sections.

2 Related Work

There has been a significant development in phrase embeddings after the word2vec breakthrough by Mikolov et al. in 2013. Earlier, word vectors were combined with some functions to create phrase vectors. (Mitchell and Lapata, 2008) developed systems with predefined composition operators. In their work, they created datasets of similarity for adjective-noun, noun-noun and verb-object bigram units. They found the simple additive and multiplicative function to be quite effective. However, these simple functions ignored word orientation in phrases and their interaction.

To make these compositions robust in order to handle complex structures in sentences, Matrix composition functions (Zanzotto et al., 2010; Socher et al., 2012) and Tensor composition functions (Bride et al., 2015) were proposed. In 2013, Mikolov et al. generated phrase representation using the same method used for word representation in word2vec (Mikolov et al., 2013a; Mikolov et al., 2013b). High-frequency multi-word units such as *New York* was embedded along with the words by taking them as single token, or pseudo-words, i.e. *New_York*. Though this method is useful for learning short phrase representations with good quality, it does not generalize well to relatively longer and rare occurring phrases in the dataset.

In 2011, Socher et al. used a recursive neural network to learn representations for multi-word phrases. In particular, they used an unsupervised autoencoder and their model performed well on the sentiment classification task but not so well on phrase similarity related problems. The primary reason for this was the low-dimensional representations (upto 50) they had used to reduce the computational complexity. More recently (Yu and Dredze, 2015), the idea of learning composition functions based on phrase structure and context was proposed to compose phrase embeddings using baseline word embeddings.

The composition model developed by Yu and Dredze used Feature-rich Compositional Transformations (FCT) from words to phrases in which the summation weights were defined by the linguistic features of component words such as POS tags, head words and so on.

3 Methodology

We develop our model with the objective of training a system to predict similarity between a word and a phrase leveraging a similarity dataset. In our work, unlike the previous approaches, we capture the sequence of words in a phrase and their interaction as well. This is achieved by using a recurrent neural network to train our model. The model learns to generate phrase representations according to its closest single word meaning. The more the semantic similarity between the word and the phrase, the closer is their similarity metric output to 0 (1 otherwise). Table 1 shows some examples giving more insight into this.

Input word	**Input phrase**	**Output**
remorse	deep regret	0
athletes	bring up	1
suez	the suez canal	0
payment	earth orbit	1

Table 1: Input-Output samples. 'Output' is the similarity metric output, *i.e. 0 for similar and 1 for dissimilar*

We join the outputs of the two inputs fed to the network using a Siamese similarity function where the input of one sub-network is the baseline embedding of a word and the input of another sub-network is the embeddings of constituent words of a phrase which are fed sequentially. The two outputs thus obtained are the resultant vector embeddings of the corresponding word and phrase generated by our system. The weights learned are common during both the inputs. In this way, we build a common abstract transformation for both the word and the phrase.

While training our system over a large dataset containing input-output pairs as in Table 1, the model learns weights with which we build phrase embeddings and fine-tune baseline word embeddings such that both are embedded in the same space.

3.1 Siamese Neural Network

For the task of signature verification, Siamese Neural Networks were first proposed by Bromley et al. in 1993. After that, the architecture has been used in several works of similarity and discrimation such as for face verification (Chopra et al., 2005), visual search (Liu et al., 2008), sentence similarity (Mueller and Thyagarajan, 2016), similar question retrieval in Q/A (Das et al., 2016), etc.

Suppose Out(X) is a set of functions which has a set of parameters *W*. In Figure 1, input A is first given to the network. Then another input B is fed and a similarity function gets the outputs of these A and B, i.e. Out(A) and Out(B). Siamese network learns a value of *W* such that the similarity metric is small if *Inp. A* (first input) and *Inp. B* (second input) are similar and large if they are not. The similarity function can be defined as:

$$S(Inp.A, Inp.B) = ||Out(A) - Out(B)|| \quad (1)$$

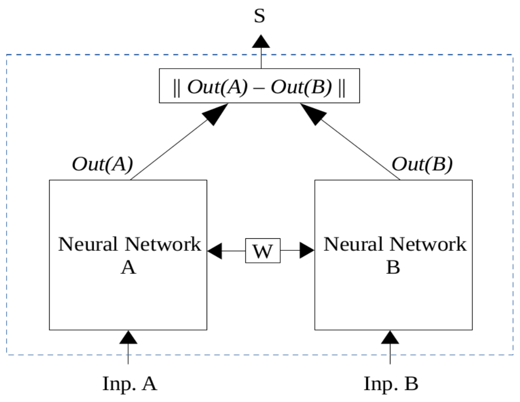

Figure 1: Siamese Neural Network

4 Model Architecture: Siamese LSTM

Long Short Term Memory Networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are often used for problems with temporal (sequential) data. Since in our work, we are dealing with phrases which are sequences of words, we use LSTM network in our Siamese architecture. We first briefly describe LSTM networks and then the model architecture in detail.

4.1 LSTM Networks

Long Short Term Memory Network is a special variant of RNN which is capable of learning long-term dependencies. Each cell of LSTM contains four components: a memory state C_t, an output gate o_t that determines how the memory state affects further units, as well as an input gate i_t that controls what gets stored in and a forget gate f_t which determines what gets omitted from the memory based on each new input and the current state. Figure 2 shows the four components and their interaction with the inputs and past and future information.

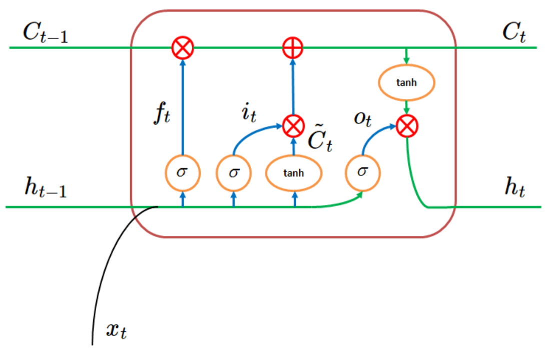

Figure 2: Block diagram of LSTM Cell

The use of LSTM network helps our system to learn the context of constituent words in a phrase.

4.2 Siamese LSTM

We use 3 layers of stacked LSTMs in our Siamese network. The hidden layers and the number of neurons were selected after repeated experiments and we obtained best results with this configuration. We limit the timesteps used in the LSTM to 5 since more than 99.8% of phrases in the dataset we use (PPDB[1]) are constituted of 5 or less units (words). Table 2 shows the composition of the PPDB data in terms of the n-grams in the phrases.

Size →	XL	
N-gram	**Nums**	**Percentage**
2-gram	2,98,536	79.40%
3-gram	60,657	16.13%
4-gram	13,132	3.49%
5-gram	2,993	0.80%
6-gram	682	0.18%
Total	3,76,000	100.0%

Table 2: Composition of the PPDB dataset of size XL showing only 0.18% of the phrases are of more than 5-grams; *the scores are 0.15% & 0.12% for XXL & XXXL sizes respectively*

At each timestep $t \in \{1 \dots 5\}$, we use baseline embeddings of the word at t. For cases (*eg.* phrases of length less than 5 or word of length 1) where there is no word at t, we use zero embedding vector at that t.

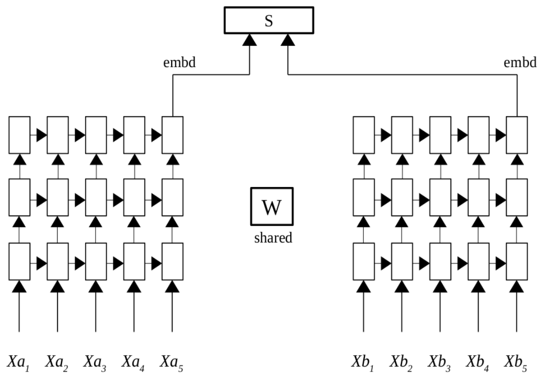
S
embd
embd
W
shared
Xa_1 Xa_2 Xa_3 Xa_4 Xa_5
Xb_1 Xb_2 Xb_3 Xb_4 Xb_5

Figure 3: Siamese LSTM model architecture; First input to the network is a single word unit and second input is a multi-word unit

Figure 3 illustrates the model architecture we use. After training over a large dataset, the model

[1] `http://www.cis.upenn.edu/~ccb/ppdb/`

learns the set of parameters W. 'embd' is the resultant embedding of the word or paraphrase which is input to the network. The outputs of the two inputs to the network are joined using a contrastive loss function (Hadsell et al., 2006) which is defined as:

$$L = (1 - Y)\frac{1}{2}(S)^2 + (Y)\frac{1}{2}\{max(0, m - S)\}^2 \tag{2}$$

where Y is a binary label assigned to a pair X_a and X_b (0 for similar and 1 for dissimilar), S is the similarity energy function which is parameterized by W and m is margin (Hadsell et al., 2006). We use $m = 1$. When two pairs are not similar the maximum energy metric outputs 1.

The gradient of the loss function with respect to W shared by the LSTM networks, is computed using back-propagation. Adamax optimizer, a variant of the Adam optimizer (Kingma and Lei Ba, 2015) is used to update the parameters of the sub-network.

5 Implementation

5.1 Dataset

We use PPDB dataset (Ganitkevitch et al., 2013) of size XL which has 3,76,000 pairs of words and their corresponding paraphrases. Since these are word-paraphrase pairs, we label the output of these pairs as 0. We augment the data by the same number of negative pairs by choosing a phrase for a word which is not its paraphrase. We label these pairs as 1. Thus, we train our model on 7,52,000 data samples.

5.2 Base Input Embedding

As base word embeddings for the input layers, we use GloVe[2] word vector embeddings (Pennington et al., 2014) of dimension 200.

The input pairs are passed through the Siamese network with the final layers of each network giving the resultant embeddings for words and phrases. We have 300 stacked neurons in the final layer of both the LSTM networks. This is the dimension of our resultant embeddings.

6 Experiments and Results

We perform the following experiments and find impressive results on various tasks.

[2]https://nlp.stanford.edu/projects/glove/

6.1 Word - Phrase Similarity Task

In this experiment, we define a classification task to determine if a word-phrase pair are semantically similar. We feed a word-phrase pair to the trained Siamese model and predict whether they are similar ($S \approx 0$) or dissimilar ($S \approx 1$).

We evaluate on the set of PPDB data (Section 5.1) left aside for validation. Out of the total data size, we choose 2,00,000 word-paraphrase pairs arbitrarily for the evaluation. Besides Siamese LSTM, we also perform this experiment on a Siamese Multi-layer Perceptron Network (MLP) where the MLP has 4 layers of neurons. As per our study in Table 2, we fix the MLP input to 5 words where each input word is in the form of a 200-D vector (we use padding and truncation for word units which are not of length 5). Therefore, the first layer has 1000 neurons. The remaining layers have 512, 512 and 300 neurons in order from second to final layer. We carry out the similarity task using Siamese MLP. However, we get better results (Figure 4) on Siamese LSTM which is also one reason why we chose it over Siamese MLP. We report the best accuracy of 76.65% on this task.

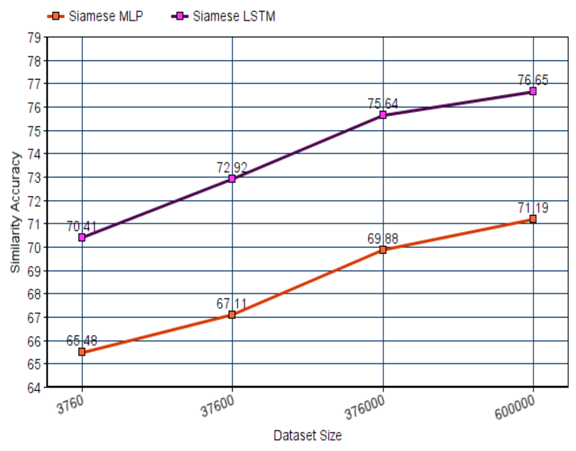

Figure 4: Accuracy of similarity over dataset size

6.2 Nearest Words and Phrases

In our work, we fine-tune the current base word embeddings while we generate phrase embeddings. Therefore, to validate the new vector embeddings of the words *(300-D)* which we obtain at the final layer of the Siamese sub-network, we perform this experiment.

For a pair $\langle U, V \rangle$, where U & V can be a single word or a multi-word unit, if U is given, we predict V (Refer Table 5). We output a list of four units which are closest to U in the vector space. We

Word	Nearest Words
viewpoints	perspectives, opinions, viewpoint, views
upbeat	optimistic, cautious, outlook, gloomy
sales	retail, selling, profits, profit
milder	colder, mild, warmer, heavier
1600	1400, 1700, 1300, 1500
asem	apec, asean, g20, summit
panelists	attendees, moderators, jurists, paragraphs
medal	medals, awarded, won, silver

Table 3: Nearest words for a given word showing that semantic relations are preserved even after the modification in the base embeddings.

Word	Nearest Phrases
viewpoints	differing views, the viewpoints, different opinions, different perspectives
upbeat	optimistic about, overly optimistic, more cautious, cautious
sales	sales volume, sales orders, export sales, the sales
milder	relatively mild, cold weather, even heavier, very mild
1600	1600 hours, 1300 hours, 1700 hours, 1100 hours
asem	the asem process, apec leaders, apec economies, summit meetings
panelists	discussion forums, two paragraphs, selected topics, panel discussion
medal	the gold medal, a gold medal, a medal, the bronze

Table 4: Nearest phrases for a given word: we show these for the same words as in Table 3 for the ease of comparison

perform this task by calculating the cosine similarity between the given U's embedding and all the units' embeddings in our vocabulary.

We finally select the top four ranked results for every U using the embeddings our model has generated and show interesting sample results in Table 3, Table 4 and Table 6. We notice several instances where semantics are preserved even after fine-tuning the baseline embeddings using our

Experiment	U	V
Table 3	Word	Word
Table 4	Word	Phrase
Table 6	Phrase	Phrase

Table 5: Experiments in this category; *word* is *1-gram* and *phrase* is *n-gram* where $n \geq 2$

approach. The phrase representations generated from this work also stays close to its corresponding similar word's embedding.

Phrase	Nearest Phrases
are crucial	are important, are needed, are essential, are necessary
2005 to 2006	2005 to 2007, 2005 to 2008, 2003 to 2006, 2004 to 2005
the violence	the acts of violence, the violent, the prevalence of violence, the cycle of violence
both leaders	the two leaders, both members, the leaders, leaders of the two countries
president hosni mubarak	president mubarak, egyptian president hosni mubarak, hosni mubarak, the egyptian president hosni mubarak
the correctness	the objectivity, the veracity, the originality, the propriety
musical works	artistic works, musical instruments, works of art, works well

Table 6: Nearest phrases for a given phrase

6.3 Semantic Similarity Task

We use the embeddings derived by our system to evaluate the phrasal semantic similarity task of SemEval2013[3] and compare our results with that of the existing systems. The task of SemEval2013 5(a) is to determine if a word-phrase pair are semantically similar (True for similar and False for dissimilar) which is notably as same as the experiment in Section 6.1. We report the results (accuracy scores) of our system along with existing benchmark methods in Table 7. RAE is the recursive auto-encoder model developed in (Socher et al., 2011) wheras FCT-LM and FCT-Joint are the Feature Rich Compositional Transformation Models proposed by (Yu and Dredze,

[3]`https://www.cs.york.ac.uk/semeval-2013`

2015) with Language Modeling and Joint Traning (Language Modeling and Task-Specific) objective respectively for updating the embeddings. We also report our results with the Recursive Neural Network (ReNN) based model developed by (Socher et al., 2011; Socher et al., 2013) and obtain comparable results with the state-of-the-art system developed for generating phrase representations evaluated on this task.

Model	SemEval2013 Test
RAE	51.75
FCT-LM	67.22
FCT-Joint	70.64
ReNN	**72.22**
Our System	**72.14**

Table 7: Performance on the SemEval2013 5(a) Semantic Similarity Task

We see that the Siamese network based model outperforms the RAE by significant margin. However, the ReNN still has the best performance. Since the method proposed in this work is primarily dependant on the dataset containing word-paraphrase pairs, the larger this data size, the better quality embeddings we can generate and the performance on this task can be ultimately improved.

7 Conclusion and Discussion

In this work, we present a novel approach in building phrase vector embeddings by the use of its constituent word vectors through a sequential Siamese model. The Siamese network designed for this task leverages a word-phrase similarity dataset (PPDB) and generates embeddings of the phrase keeping in consideration the word position in the phrase, and its orientation and interaction with neighbour words as well which, in particular, is achieved using a Long Short Term Memory Network. Unlike previous attempts in building compositional models for phrase representation, the system presented in this paper does not employ any manual feature based technique for building phrase embedding or rely on a POS tag of particular word's neighbour or head words, or any other linguistic feature. Rather, we develop a similarity based deep learning network with contrastive loss which learns its weights after training and this set of learned parameters function as an abstract transformation which compose the phrase representation eventually. In addition, we fine-tune the base word vectors using the same abstract transformation and embed both the words and phrases in the same vector space.

The phrase representations derived from our model are computationally efficient as compared to recursive neural networks employed for this task. Besides, we are able to generate comparatively higher dimension (300-D) vectors in this work as compared to recursive networks (25-30D) which have a higher computational complexity. We show excellent results on phrase similarity task using the vector embeddings produced from this work. Also, semantic relationship between nearby words-phrases and phrases-phrases has been preserved.

8 Future Work

Since the network used in this work is trained on the paraphrase dataset, quality of phrase embeddings will improve if we use more exhaustive set and large number of word-paraphrase pairs for training. If we are able to extract more paraphrases using bigger language corpus and use it for training our system, we can considerably improve the quality of vectors derived. In future, we plan to extend this work to other languages as well by first extracting paraphrases and then learning a similarity model to derive phrase representations. However, efficient and reliable word vectors of higher dimension is required for this work to be done. Similarly, we intend to use phrase embeddings generated by our model in several NLP applications with motive of improving the performance.

Acknowledgement

We would like to thank Naveen Kumar Laskari for discussions during the course of this work and Pruthwik Mishra and Saurav Jha for their valuable suggestions.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR), 3:1137-1155.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546.*

Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Association for Computational Linguistics (ACL)*, pages 236-244.

Mo Yu and Mark Dredze. 2015. Learning Composition Models for Phrase Embeddings. In *Transactions of the Association for Computational Linguistics (ACL)*, pages 227-242.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Sackinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4).

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:539–546.

Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. 2008. Understanding and summarizing answers in community-based question answering services. In *International Conference on Computational Linguistics (COLING)*, 1:497–504.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI Conference on Artificial Intelligence*.

Arpita Das, Harish Yenala, Manoj Chinnakotla and Manish Shrivastava. 2016. Together We Stand: Siamese Networks for Similar Question Retrieval. In *Association for Computational Linguistics (ACL)*, pages 378-387.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Juri Ganitkevitch, Benjamin Van Durme and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, pages 758–764.

Raia Hadsell, Sumit Chopra, Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:1735–1742.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Ng. 2013. Parsing with compositional vector grammars. In *In Association for Computational Linguistics (ACL)*, pages 455–465.

Antoine Bride, Tim Van de Cruys, and Nicholas Asher. 2015. A Generalisation of Lexical Functions for Composition in Distributional Semantics. In *Association for Computational Linguistics (ACL)*, pages 281–291.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *The International Conference on Learning Representations (ICLR)*.

Fabio M. Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating Linear Models for Compositional Distributional Semantics. In *International Conference on Computational Linguistics (COLING)*, 1263-1271.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Empirical Methods in Natural Language Processing (EMNLP)*, 151-161.