Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

# Natural Language Inference Using LSTM Model with Sentence Fusion

Senlin Zhang, Siyang Liu, Meiqin Liu
College of Electrical Engineering, Zhejiang University, Hangzhou 310027, P. R. China
E-mail: {slzhang, liusiyang, liumeiqin}@zju.edu.cn

**Abstract:** Natural language inference (NLI) is a challenge and the foundation of realization of artificial intelligence. With the availability of large-scale annotated corpus, the neural network model can be widely used in natural language inference. In this paper, we propose a long short-term memory (LSTM) model with Sentence Fusion architecture for NLI task. Instead of modifying the internal structure of the LSTM recurrent neural network (RNN) model, we focused on how to make full use of the distributed expression of sentence generated by the LSTM encoder. We improved the performance of basic LSTM recurrent neural networks on Stanford natural language inference (SNLI) corpus by adding Sentence Fusion modules which enrich the distributed expression of sentence generated by the LSTM. Our results demonstrate that the LSTM with Sentence Fusion which reads premise and hypothesis to produce a final fusion representation from which a three-way classifier predicts label has a better performance than LSTM RNN encoders and Lexicalized classifier.

**Key Words:** Natural Language Inference, Long Short-Term Memory, Recognizing Textual Entailment, Deep Learning

## 1 Introduction

Natural language inference (NLI) is a challenge and the foundation of realization of artificial intelligence. Natural language inference is also known as recognizing textual entailment (RTE). The goal of RTE is judging whether a natural language hypothesis h can reasonably be inferred from a natural language premise p [1]. For many natural language processing applications like Reading Comprehension(RC), Question Answer(QA) and Information Retrieval(IR), natural language inference is an essential step.

Natural language inference has been extensively studied by previous work. Methods using hand engineered feature such as frequency-based term weighting with lexical similarity measures [2], directed graph extracted from a dependency parser [3], syntactic features and a general purpose thesaurus [4], string similarity at the lexical and shallow syntactic level [5], inference rules generation with hand-crafted lexical resource [6] have been proposed in the past decade. But it's impossible to hand-pick all the features because of the complexity of features.

The recently released Stanford Natural Language Inference(SNLI) corpus[7] has 570k human-written English sentence pairs manually labeled for balanced classification, supporting the task of NLI. It has made it possible to implement deep learning network on the task of natural language inference. Like the simple example of SNLI corpus showed in Tables 1. There are three relationships between premise and hypothesis which are Entailment(hypothesis can be inferred to be true), Contradiction(hypothesis can be inferred to be false) and Neutral(truth can't be inferred).

Table 1: Example of relation between premise and hypothesis in SNLI Corpus

| |
|---|
| P: *Two women having drinks and smoking cigarettes at the bar.* |
| $H_1$: *Two women are at a bar. -Entailment* |
| $H_2$: *Three women are at a bar. -Contradiction* |
| $H_3$: *Women are celebrating at a bar. -Neutral* |

Methods with neural networks have been introduced into natural language inference on the SNLI corpus, such as building a basic 100D SUM/RNN/LSTM network to do the SNLI task in Bowman et al.(2015)[7], LSTM with conditional encoding and attention-based technique in Rocktäschel et al.(2015)[8].Mou et al.(2016) proposed Tree-based CNN [9]. Bowman et al.(2016) proposed a model which combines parsing and interpretation within a single tree-sequence hybrid model by integrating tree-structured sentence interpretation into the linear sequential structure of a shift-reduce parser [10]. Liu et al.(2016) applied word-level bidirectional LSTM with inner-attention mechanism [11].Munkhdalai and Yu(2016) proposed Neural Tree Indexers which provides a middle ground between the sequential RNNs and the syntactic tree-based recursive models [12].Wang and Jiang proposed the match-LSTM architecture which performs word-by-word matching of the hypothesis with the premise [13]. Parikh et al.(2016) proposed a decomposable attention model which is simple but very effective by using attention to decompose the problem into subproblems that can be solved separately [14].

In this paper, we propose a LSTM Model with Sentence Fusion architecture for the NLI task on the SNLI corpus. Instead of modifying the internal structure of the LSTM recurrent neural network model, this architecture focused on how to make full use of the distributed expression of sentence generated by the LSTM.

The main contribution of this paper is that we have improved the performance of basic LSTM recurrent neural networks on SNLI by adding Sentence Fusion modules which enrich the distributed expression of sentence generated by LSTM. The improved model achieved a score of 82.10% in the SNLI corpus.

Our code is available online[1].

## 2 Background

### 2.1 LSTM

A Long Short-Term Memory (LSTM) recurrent neural network architecture was introduced by Hochreiter and

[1] http://source.64yang.com/ccc2017.zip

Schmidhuber in 1997 [15]. This model can overcome the problem of vanishing gradients of previous models. They introduced a memory cell unit to release the ordinary node. The memory cell unit makes it possible that the gradient can flow through many time steps without vanishing or exploding by a self-connected recurrent edge [16] . Input gate and output gate can learn to control how much information can access to the cell. Forget gate was introduced into LSTM in 2000 by Gers, Schmidhuber and Fred [17] which gives the ability to learn to forget internal resources of to memory cell unit to avoid the network break down caused by the state grow indefinitely. we use $X = (x_1, x_2, ..., x_N)$to denote the input sequence, where $x_t \in \mathbb{R}^l$,There are many variants of the LSTM modelhere we show one adopted by Rocktäschel[8] as follows.

$$InputGate : i_t = \sigma(W^i x_t + V^i h_{t-1} + b^i) \quad (1)$$

$$ForgetGate : f_t = \sigma(W^f x_t + V^f h_{t-1} + b^f) \quad (2)$$

$$OutputGate : o_t = \sigma(W^o x_t + V^o h_{t-1} + b^o) \quad (3)$$

$$CellState : c_t = f_t \odot c_{t-1} + i_t \odot tanh(W^c x_t + V^c h_{t-1} + b^c) \quad (4)$$

$$HiddenState : h_t = o_t \odot tanh(c_t) \quad (5)$$

where $W^i$,$W^f$,$W^o$,$W^c \in \mathbb{R}^{d*l}$ and $V^i$,$V^f$,$V^o$,$V^c \in \mathbb{R}^{d*d}$ are trained matrices. $b^i$,$b^f$,$b^o$,$b^c$ are trained biases. d is hidden size of LSTM. $\sigma$ denotes sigmoid function and $\odot$ denotes element-wise multiplication.

### 2.2 Word Vector

Hinton presented the distributed representations of concepts in 1986 [18]which is the beginning of distribution representations of words. The word vector provides a direct idea of the similarity between words. The basic idea of the word vector is that, by a large number of corpus training, each word in corpus is mapped into a fixed length vector, which in general is much smaller than the size of the word dictionary, usually in the tens to hundreds of dimensions. All of these vectors constitute the word vector space, and each vector is visible as a point in the space. In this space, the "distance" measure is introduced, and the similarity between the words and the semantics can be judged according to the distance of the word vector.

#### 2.2.1 Word2vec Algorithm

Word2vec was presented and implemented by Tomas Mikolov team of Google in 2013[19, 20]. The algorithm can learn high-quality word vectors from large-scale corpus in a short period of time. In this way, the low-dimensional vector expression of each word is obtained, which can easily calculate the semantic similarity between the word and the word, and this similarity has multi-angle and supports the linear operation. For example, v(*King*) - v(*Man*) + v(*Woman*) is close to the vector of *Queen*. The advantage of the way of feeding is that the corpus needs to learn the word vector does not require training data like the usual training classifier, nor does it require a classification system like Wikipedia that has human experience, Only need a lot of normal preparation of the contents of the file.

#### 2.2.2 GloVe Algorithm

GloVe was presented by Jeffrey et al.(2014) from the Stanford NLP group[21], which is another unsupervised learning algorithm for word embedding like Word2vec.

In GloVe, global word-word co-occurrence counts are trained by a specific weighted least squares model which produces a word vector space with meaningful substructure.

## 3 Model

Our method was motivated by the basic 'Siamese' architecture in in Bowman et al.[7, 10] and Matching Heuristics of the Tree-based CNN in Mou et al.(2016) [9].

In our model, we considered NLI tasks on SNLI as a three-way classification problem, The overall model architecture is shown in Fig. 1. First of all, We fed premise and hypothesis into two word encoders respectively. The word in sentence will be replaced with corresponding word vector in the word embedding layer. Then sentence vector will be generated by feeding the word vector into LSTM network serially. The sentence vector of premise and hypothesis will do some fusion operation and then feed into the three-way classifier.

### 3.1 Word Embedding

We adopted the GloVe model [21] as our word embedding model. For the reason of comparison, we did not train our own word vectors, but used *glove.840B.300d*[2] which are pre-trained word vectors by Pennington. This pre-trained word vectors dataset contains 840 billion word vectors with the dimension of 300.

There are 46433 vocabularies in the SNLI corpus, of which 4043 vocabularies are not found the corresponding word vector in *glove.840B.300d*. We initialized these unknown vocabularies to zero.

The top 10 vocabularies which are not found in *glove.840B.300d* are shown in Tables 2, most of the unknown vocabularies (1108 in 4043) are the combination of the two words vocabularies By '-'.

Table 2: The top 10 unknown vocabularies not found in *glove.840B.300d*

| Unknown Vocabularies | Count |
|---|---|
| *blond-hair* | 1513 |
| *rollerskaters* | 59 |
| *surfboarder* | 56 |
| *Blond-haired* | 52 |
| *graffited* | 43 |
| *parasailer* | 30 |
| *rollerskater* | 30 |
| *blond-headed* | 29 |
| *boogieboard* | 28 |
| *casually-dressed* | 27 |

### 3.2 Sentence Embedding

The SNLI corpus provides enough data which make it possible to train deep learning network to obtain the sentence distributed representations of the sentence, To learn distributed representations of the sentence, we fed one word vector form premise and hypothesis sequentially into two

[2]http://nlp.stanford.edu/data/glove.840B.300d.zip

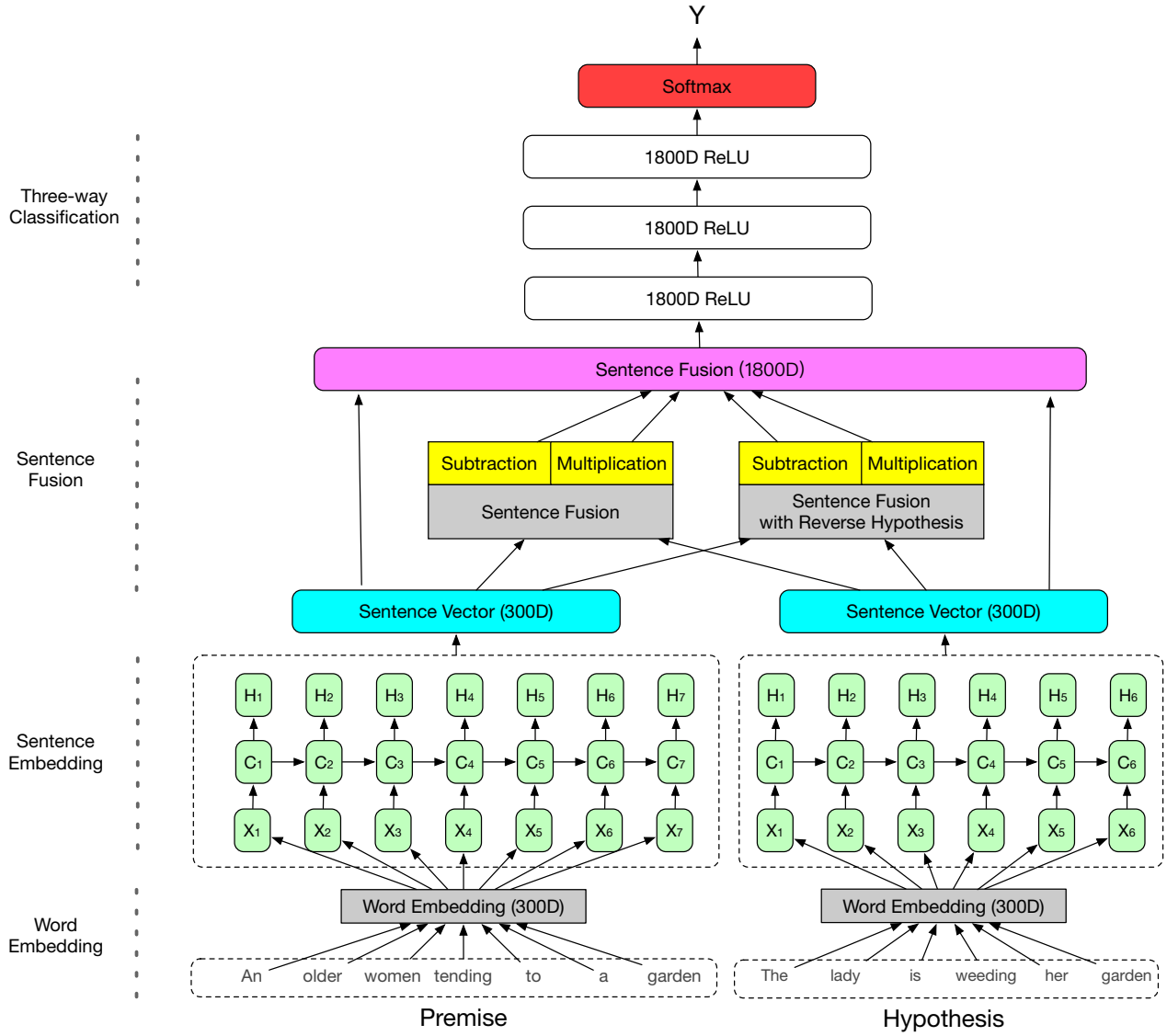


Fig. 1: The overall model architecture of the LSTM model with Sentence Fusion

separate LSTM networks at a time. The last output of LSTM networks denotes the distributed representations of this sentence. At this stage of training sentence vector, we did not use any context from the other sentence.

### 3.3 Sentence Fusion

After two sentence vectors are trained, we applied Sentence Fusion on the two vectors generated by sentence embedding.

Like Mou et al.(2016)[9], we used $P = (p_1, p_2, ..., p_{300})$to denote the vector of premse, $H = (h_1, h_2, ..., h_{300})$to denote the vector of hypothese.

There are two way to apply Sentence Fusion, first, like it shown in Fig. 2, we applied element-wise subtraction and element-wise multiplication on the P and H like formula (6) and formula (7) where $\odot$ denotes element-wise multiplication. We also applied Sentence Fusion with the reverse hypothesis like Fig. 3. This two fusion methods enrich the distributed expression of sentence.

$$PH_s = P - H \tag{6}$$

$$PH_m = P \odot H \tag{7}$$

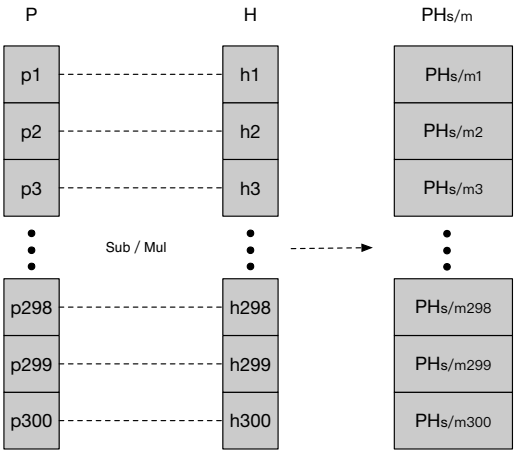


Fig. 2: Sentence Fusion

Table 3: The Result of LSTM with Sentence Fusion

| model | dimension | Train acc. (%) | Dev acc. (%) | Test acc. (%) |
|---|---|---|---|---|
| Lexicalized classifier (Bowman et al. 2015) | - | 99.7 | - | 78.2 |
| LSTM RNN encoders (Bowman et al. 2015) | 100 | 84.8 | - | 77.6 |
| LSTM RNN encoders (Bowman et al. 2016) | 300 | 83.9 | - | 80.6 |
| LSTM with Sentence Fusion (this paper) | 300 | 89.01 | 83.07 | 82.10 |
| LSTM with Sentence Fusion(without Mul.) (this paper) | 300 | 89.14 | 82.89 | 81.80 |
| LSTM with Sentence Fusion(without Sub.) (this paper) | 300 | 87.79 | 81.89 | 80.74 |
| LSTM without Sentence Fusion (this paper) | 300 | 82.80 | 79.84 | 79.11 |

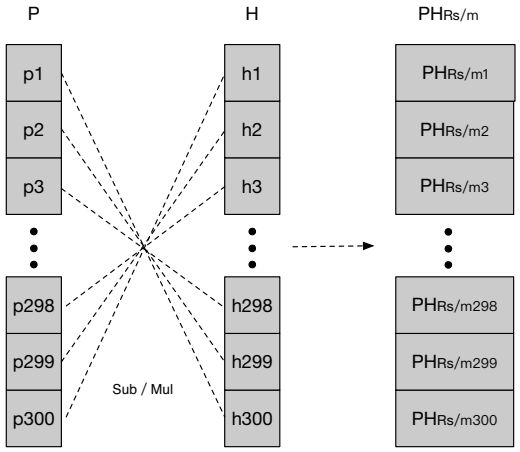


Fig. 3: Sentence Fusion with Reverse Hypothesis

We merged two sentences to generate four fusion vectors by applied element-wise addition and element-wise subtraction separately on the sentence vector of premise and hypothesis. then we connected six sentence vectors to generate a new distributed fusion representation of the sentence. This distributed fusion representation was the ultimate encoding of this RTE problem, which we used to classify.

## 4 Experiments

### 4.1 Experiments Environment

We implemented our method on Keras[3] with the backend of Tensorflow on Ubuntu 14.04 on a server with two 2.40GHz 16 core Intel Xeon E5-2630 processors, 32Gib memory, Nvidia Tesla K40m GPU.

### 4.2 Date Pretreatment

We used SNLI corpus to test our model. This corpus contains 570152 pairs of sentences with the label of 'entailment', 'contradiction', 'neutral' and '-', where '-' indicates that human have not reached a consensus on this sentence.

Because we consider SNLI tasks as a three-way classification problem, we removed the data labeled '-' from the original corpus and use the remain ones to do our experiments like Bowman[7]. The remaining corpus contains 549367 pairs of sentences for train, 9842 pairs of sentences for validation, 9824 pairs of sentences for test.

### 4.3 Model Parameter

We used RMSProp [22] for optimization with the learning rate of 0.001, the rho of 0.9, the epsilon of 1e-08 and decay of 0. We set the hidden layer size of both LSTM networks 300. The dropout rate was 0.2. We used l2 regularization with the strength of 4e-6. Batch size was set to 512. Accurary and loss information is shown in Fig. 4

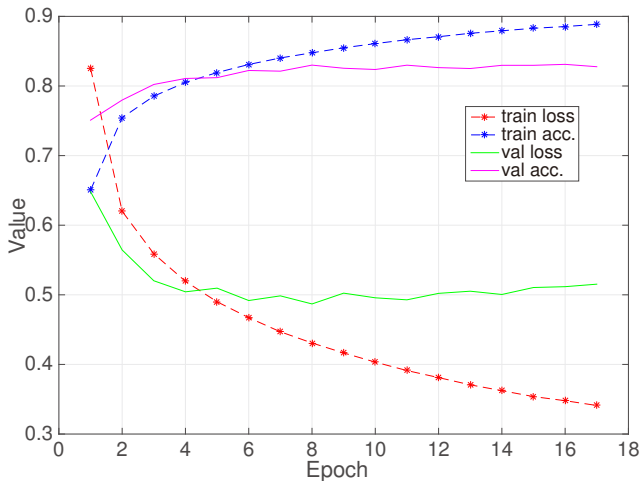


Fig. 4: Accuracy and Loss

## 5 Results and Analyses

On the basis of in Bowman et al.[7, 10], We applied the Sentence Fusion module which integrates premise and hypothesis. The result is shown in Tables 3, Our LSTM with Sentence Fusion had a score of 82.10% on the test set. When we removed the element-wise subtraction of Sentence Fusion module, the score dropped to 81.80%. When we removed the element-wise multiplication of Sentence Fusion module, the score dropped to 80.74%. When we removed the Sentence Fusion module, the score dropped to 79.11%.

## 6 Conclusions

In this paper, we propose a LSTM model with Sentence Fusion architecture for NLI task on the SNLI corpus. We have not focused on modifying the internal structure of the Long Short-Term Memory recurrent neural network model but how to make full use of the distributed expression of sentence generated by the LSTM. we show how Sentence Fusion enriches the distributed expression of sentence. We used basic LSTM recurrent neural networks to generate sentence vectors of premise and hypothesis respectively. then we used Sentence Fusion to make full use of these sentence vectors. Finally a 3-way classifier was used to predicts label.

Our results demonstrate that LSTM recurrent neural networks with Sentence Fusion which read premise and hypothesis to produce a final fusion representation from which a three-way classifier predicts label has a better performance

[3] https://github.com/fchollet/keras

than LSTM RNN encoders (Bowman et al. 2016)[10] and Lexicalized classifier (Bowman et al. 2015)[7].

In the future, we will improve our model by combining vocabulary similarity and attention mechanisms. We will add syntax and semantic information to enrich the distributed representation.

## References


[1] Bill MacCartney. *Natural language inference*. PhD thesis, Citeseer, 2009.

[2] Valentin Jijkoun, M Rijke, et al. Recognizing textual entailment using lexical similarity. 2005.

[3] Aria D Haghighi, Andrew Y Ng, and Christopher D Manning. Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 387–394. Association for Computational Linguistics, 2005.

[4] Lucy Vanderwende, Arul Menezes, and Rion Snow. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the Second PASCAL Challenges Workshop*, 2006.

[5] Prodromos Malakasiotis and Ion Androutsopoulos. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics, 2007.

[6] Georgiana Dinu and Rui Wang. Inference rules and their application to recognizing textual entailment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 211–219. Association for Computational Linguistics, 2009.

[7] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[8] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[9] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. In *The 54th Annual Meeting of the Association for Computational Linguistics*, volume 130, 2016.

[10] Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016.

[11] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*, 2016.

[12] Tsendsuren Munkhdalai and Hong Yu. Neural tree indexers for text understanding. *arXiv preprint arXiv:1607.04492*, 2016.

[13] Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015.

[14] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[17] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[18] G E Hinton. Learning distributed representations of concepts. 1986.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[21] P Jeffrey, S Richard, and M Christopher. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, 2014.

[22] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.