Question-Question Similarity in Online Forums

Yllias Chali
University of Lethbridge
chali@cs.uleth.ca

Rafat Islam
University of Lethbridge
rafat.islam@uleth.ca

ABSTRACT

In this paper, we applied deep learning framework to tackle the tasks of finding duplicate questions. We implemented some models following the siamese architecture using the popular recurrent network such as Long-Short term memory (LSTM), Bi-directional Long-Short term memory (biLSTM) to find the semantic similarity between questions. We started with a basic model and further extended the basic model into three different models. Our models provide a refined, composite representation of the questions. The addition of Convolutional Neural Network (CNN) with the recurrent networks is a new approach for the sentence representation. We also applied attention mechanism for getting better contextual meaning of the questions. We generated a representation of a question according to the context of another question for solving the task. As neural models are data driven, we trained our models extensively by making pairs, such as question-question over a large-scale real-life dataset. We used a datset consisting of 400K labeled question pairs which are published by a well known question-answer forum Quora. We evaluate our models based on metrics like accuracy, precision, recall, F1 scores. Our methods and experiments demonstrate some significant improvements over the baseline systems and the state-of-the-art systems.

ACM Reference Format:
Yllias Chali and Rafat Islam. 2018. Question-Question Similarity in Online Forums. In Forum for Information Retrieval Evaluation (FIRE'18), December 6–9, 2018, Gandhinagar, India. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3293339.3293345

1 INTRODUCTION

Question-Answer forums are gaining more and more popularity day by day. Each forum itself becoming a community where people are more active than ever. As a result a large number of questions are posted each day, among them there are some questions which have already been answered. This redundant posting of the same question by using different words leads to duplicate question problem. For example if a person asks, "What do you mean by force in physics?" in a forum while in the same forum another question was asked by another person like "Can someone explain force in terms of physics?". Both of these questions are asking for the same answer but are different from each other in terms of the order of the word and syntax. It may happen that one of the question remain unanswered or wait for longer period to get answered as the users of the forums providing the feedback get frustrated by receiving the same question again and again with slight variations. Thus, a good paraphrase detection or duplicate question detection system can classify these two questions and mark them as similar. Therefore restricting the user posting a redundant question by bringing the answers of the questions that are already there in the forum.

In this paper, we propose four deep neural models to detect similar question using neural network architecture. The models correctly identify duplicate questions and mark them as duplicate or not duplicate. These models are independent and do not require any external knowledge, feature engineering

FIRE'18, December 6–9, 2018, Gandhinagar, India

ACM ISBN 978-1-4503-6208-5/18/12…$15.00
https://doi.org/10.1145/3293339.3293345

and language tools. All of the models use variations of Recurrent Neural Networks (RNN) such as Long-Short term memory (LSTM), Bi-directional Long-Short term memory (biLSTM). Siamese architecture allows us to use this RNNs simultaneously and separately for both questions in a pair. This architecture also showed inspiring results in distance based learning methods [2, 3] and also for classification tasks such as categorizing semantically similar questions [4].

This paper contributes to develop an efficient question duplication detection system using the deep neural architecture. We also introduced a hybrid model using both the recurrent neural network and the convolutional neural network together. The main idea is to combine this two networks to get a compact and better contextual representation of the questions.

The rest of the paper is organized as follows: Section 2 describes some related works on duplicate question detection systems. Section 3 provides the task description, explains our approach and models. Section 4 includes discussion on the dataset used for the training and experimentation, the data processing and training setup, evaluation of our models, our baseline systems and comparison of our results with the baseline models and within our models. Finally section 5 draws the conclusion of the paper.

2 RELATED WORKS

Our model structures are inspired form non-factoid answer selection models [16]. Rocktaschel [14] provided a reasoning based model to find the semantic similarity between two sentences. Inference techniques were used to achieve the semantic similarity. A neural attention model for machine translation was initially implemented by [1]. The sentence representation vectors are broken into vectors of sub-phrases after the tokens of these two sentences are soft aligned according to the attention model. This attention was more efficiently used for answer selection and interactive neural network by [21]. Recently [20] introduced an attention mechanism implementing word level and sentence level attention following hierarchical structure. Siamese network architecture was first introduced by [2] for signature verification. Since then it has been prominently used for learning the complex similarity metrics. This architecture was used for face verification [3] and matching sentences along with convolutional neural network [8].

The most recent work on paraphrase detection is based on the bilateral multi-perspective matching of the sentence vectors [17]. Deep neural networks are used as an encoder for encoding the sentences. These encoded vectors are then matched against each other to form the multi-perspectives.

3 APPROACH

In this section, we describe our basic model and its variations to solve our task. Our basic model uses only long short-term memory on both questions in the question pair and then aggregate over each set of vectors using summation technique [12]. Finally passing the results through a softmax classifier for classifying the questions as either duplicate or not-duplicate. In the following five subsections, we formalize our task, discuss about the basic model, extensions and variations of our basic model. All our models are using similar siamese network structure [11] and inspired by the models proposed by [16] and [19].

3.1 Task Description

The main goal of this task is to find identical questions pairs. Our models can classify the questions into two categories duplicate or not-duplicate. We formalize our task of finding identical or duplicate questions as follows: given two sets of question $Q_1 = (q_{11}, q_{12}, q_{13}, \ldots, q_{1n})$ and $Q_2 = (q_{21}, q_{22}, q_{23}, \ldots, q_{2n})$, our models takes a question from Q_1 and Q_2 to make pair (q_{1i}, q_{2i}) as input, and finds out whether the questions are duplicate or not-duplicate.

3.2 Question-Question LSTM

Currently Recurrent Neural Networks (RNNs) are extensively used to deal with the input sequence of variable lengths. As RNNs can connect the previous information to the present information by forming a neural network of repeated module chain, it allows us to look into the present information describing the previous information. Long Short-Term Memory (LSTM) [7] has been proven to solve

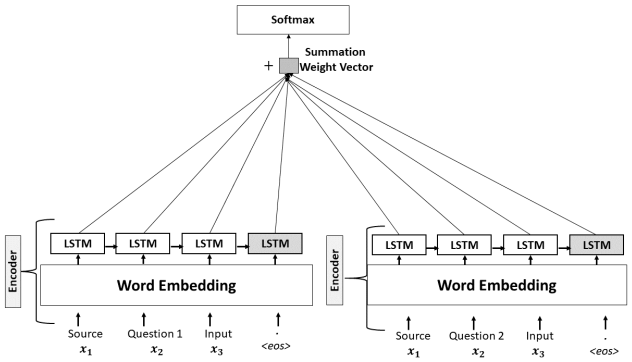

Figure 1: Question-Question LSTM model

the shortcomings of the RNN by handling the gradient vanishing problem. It is also the most popular variant of RNN. We implemented a simple LSTM based model in Figure 1 that produces distributed weight representations for question 1 and question 2 independently. We used separate encoders for processing the questions following a siamese architecture. The encoder in this case is a simple unidirectional LSTM that receives the d-dimensions word vectors as an input sequence such as $x = x_1, x_2, \ldots, x_n$ to learn the hidden annotations h_t at time t as processing the input from left to right as follows:

$$h_t = f(x_t, h_{t-1})$$

Where, $h_t \in \mathbb{R}^n$ encodes all of the contents seen so far at time step t which is computed from h_{t-1} and x_t, where $x_t \in \mathbb{R}^m$ is the m-dimensional embedding of the current word x_t. It generates an output h_t when t^{th} unit along with the previous output h_{t-1} are passed to the current input x_t. When calculating h_t it uses i_t, f_t, c_t, o_t which are input gate, forget gate, memory cell and output gate. respectively, to achieve a good performance over longer sequences. The forward propagation of LSTM is as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (4)$$

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

In the above equations, W_x, W_c ,W_f, $W \in \mathbb{R}^{n \times m}$ are weight matrices, n is the number of hidden units, $\sigma()$ is the sigmoid function, tanh is the hyperbolic tangent and $\odot$ denotes an element-wise multiplication. As shown in Figure 1, LSTM processes the word embeddings generated from the questions. The LSTMs from both the encoders produce separate vector outcomes. We now aggregate over the vectors by summation following the formula:

$$x_1 = \sum_{i=1}^{length_{question1}} x_{1,i} \quad , \quad x_2 = \sum_{i=1}^{length_{question2}} x_{2,i} \quad (6)$$

The outputs are passed through a softmax classifier to classify between the duplicate and not-duplicate questions. $length_{question1}$ and $length_{question2}$ represents the question lengths and x_1, x_2 representing the individual questions in a pair. Categorical crossentropy has been used as the loss function. The following equation calculates the loss function between the predicted value and actual value which is the ground truth.

$$H(p, q) = -\sum_x p(x) \log(q(x))$$

Where $p(x)$ denotes the actual value and $q(x)$ denotes the predicted value of the input pair of questions in either 0 or 1 and x is the number of outcomes or predictions.

3.3 Question-Question LSTM with Attention

In this model the encoder learns the hidden representations in the same way as described in section 3.2. We now pass these two output vectors coming form the encoder to an attention mechanism layer. In this section we added an dot attention mechanism layer [10] for its efficiency and simplicity in the implementation. The dot attention mechanism is actually the dot product between two hidden vectors. Using attention mechanism removes the bottleneck of processing the fixed length hidden vectors which propagates important information over longer distances. An attention mechanism captures the context of the vectors, keeping in mind that we do not lose information while reducing matrix dimensions. The attention layer adds an extra weight to the vectors. At each time step t, a context vector c_t is generated by the LSTM hidden state (h_i) from the encoder. The weight α_{ij} is the strength

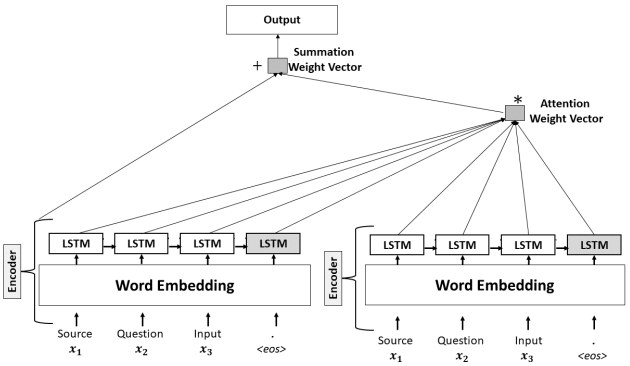

Figure 2: Question-Question LSTM with Attention model

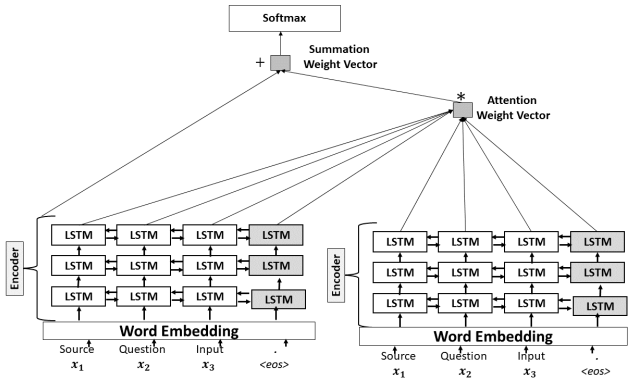

Figure 3: Question-Question Bi-Directional LSTM-Stacking with Attention

of the attention i^{th} word in the target vector to the j^{th} word in the source vector. The general idea behind the attention vector is that it tells us how much focus should be given to a particular source word at a particular time step of the encoder. The larger the value of α_{ij}, the more focus will be given to a particular word while forming the next vector representation.

$$c_t = \sum_{i=1}^{N} a_{ij}\ h_t$$

$$a_{ij} = \frac{exp(h_t^{Q_1}.\ h_i^{Q_2})}{\sum_i exp(h_t^{Q_1}.\ h_i^{Q_2})}$$

Where, $h_t^{Q_1}$ and $h_t^{Q_2}$ are the LSTM hidden states at each time step t coming from the question 1 encoder and question 2 encoder, respectively. This strategy is also used in other tasks such as factoid question answering [6, 16], sentence summarization [15], machine translation [1]. Then we merge the encoder outcome of question 1 with the outcome of the attention layer to form the final representation. We use the same softmax classifier as described in the previous model to mark the questions as duplicate or not-duplicate. We minimized the loss function using the categorical crossentropy loss function.

3.4 Question-Question Bi-Directional LSTM-Stacking with Attention

In this section, we used bi-directional RNN (Bi-RNN) [5] as encoder for learning the hidden states but for our case we replaced the RNN with an LSTM making it a bi-directional LSTM (Bi-LSTM). Simple uni-directional LSTM that learns the hidden annotations h_t at time t as processing the input from left to right is as follows:

$$h_t = f(x_t, h_{t-1})$$

We implemented Bi-LSTM that processes each question word embedding representation in both forward and backward direction calculating the hidden annotations h_t at each time step t. For each position t, we merge the hidden states by concatenation to achieve the final hidden state:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}$$

Where operator $\oplus$ indicates concatenation and $\overrightarrow{h_t}$, $\overleftarrow{h_t}$ represents the forward and backward representation, respectively. Forward representation $\overrightarrow{h_t}$ is calculated from $\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}})$ and backward representation $\overleftarrow{h_t}$ is calculated from $\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t-1}})$. In this model, we implemented a 3-layer Bi-LSTM following [10] by stacking 3 Bi-LSTM layers on top of each other. A 3-layer Bi-LSTM makes the model deeper and each layer learns and extracts more information of a sentence. We could use more layers of LSTMs but we followed the structure in the above mentioned paper and obtained a better performance. The calculation at time step t would be as follows:

$$h_{1,t} = \text{LSTM}_1(x_t,\ h_{1,t-1}) \quad (7)$$

$$h_{2,t} = \text{LSTM}_2(h_{1,t},\ h_{2,t-1}) \quad (8)$$

$$h_{3,t} = \text{LSTM}_3(h_{2,t},\ h_{3,t-1}) \quad (9)$$

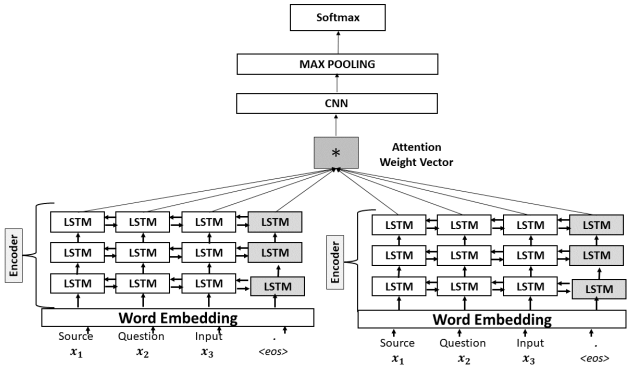

Figure 4: Question-Question Bi-Directional LSTM/attention/CNN with Maxpooling

Then, we passed the encoder outputs to an attention layer (same attention layer mechanism described in the previous model in section 3.2) to obtain an attention representation vector. We also follow the same merging technique of the previous model to merge the outcome vector of question 1 encoder with the attention representation vector to form the final representation. We use softmax classifier for identifying the classes between duplicate and not-duplicate. In this model, we use categorical crossentropy as the loss function. Figure 3 shows the model structure.

3.5 Question-Question Bi-Directional LSTM-ATTN-CNN with maxpooling

In this part we introduce convolutional Neural Network (CNN) to our model. We used the same stacked Bi-LSTM and attention mechanism described in the previous sections but added a CNN layer along with max-pooling layer. We applied a one dimensional convolution layer with a filter size of 5*d along with a stride size of 1, where d represents the dimensionality of the word embeddings. Filter size of 5 allows us to build a 5-gram model for a large sentence to predict and extract information. On top of the CNN layer, we use a max-pooling layer of 2*2 to reduce the matrix dimension generated from the convolution layer. Max-pooling layer is used to extract the crucial local features to form a fixed length feature vectors. Then we use a softmax classifier to classify the questions into duplicate or not-duplicate class. Categorical crossentropy loss function is used to minimize the loss function. Figure 4 shows the model structure.

4 EXPERIMENTS

4.1 Dataset Description

For conducting our experiments we used the Quora dataset [1]. This dataset originally released by Quora has 400K pairs of questions each marked with a ground truth of either 1 or 0, where 1 denotes duplicate pair of questions and 0 denotes the opposite. Each pair of questions has a unique identifier. Each question pair also has a unique id named as qid1 and qid2. The questions are separated in two columns as question1 and question2, thus (question1, question2) forming a pair. This dataset contains 149,306 positive examples (i.e, semantically duplicate questions) and 255,045 negative examples.

4.2 Data Processing and Training

We preprossessed the dataset removing the punctuations and stop-words. We generated the word embeddings using the 300 dimensional pretrained GloVe [13] to represent the words, pretrained with 840B tokens. We train all our models mentioned in this paper with the same parameters. We used Adaptive Moment Estimation ADAM [9] optimizer with a learning rate of 0.001. We used ADAM because of its intuitiveness during the training phase and its less memory requirements. We started with a very small learning rate to avoid over fitting in the early stage of training. ADAM optimizer is adaptive in nature, so learning rates during the training will be set accordingly after each epoch by the optimizer. We tried to minimize the loss function using the categorical crossentropy. We experimented on the sequence length using average sequence lengths of the vectors. We split the dataset into two portions for training and testing, where 90% data were placed in training and validation set while the rest 10% were kept for testing. We set the hidden size of the LSTM and the Bi-Directional LSTM layers as 64 in order to get a more high-level representation. In case of the Bidirectional RNNs, we take the outputs from the last timestep for each one and

[1] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

concatenate them to get a vector of size 128 (Forward and Backward). We processed the questions by batch of 32 question pairs each time for training the network. Typically training is faster with mini-batches as the weights are updated after each propagation. But one of the disadvantages is that the smaller batches provide less accurate estimate of the gradient. The whole training process ran for 20 epochs. Epochs are one forward pass and one backward pass on all the training examples. For the training, we used TITAN X GPU machine with 12 gigabytes of memory.

4.3 Evaluation

We compare the results of our systems in terms of Accuracy, Precision, Recall, and F1 score. We also compare our system against recent works based on the Quora dataset. We performed the evaluation on 40K question pairs randomly chosen from the dataset and different from the training set as the sets are divided before training.

While running our experiments on the models, we observed a similar pattern for the convergence. The accuracy after the 6th epoch increased sharply comparing to the previous epochs and became flat over the next 2 or 3 epochs. We observed the similar situation in the case of the training loss too. The loss at the beginning of the training was high but it quickly came down and flattened after 7th and 8th epochs.

The training time for each model was about 2 days on average. The models that include stacked Bi-directional LSTM took the longest time to train and the model with only LSTM took the least time to complete the training. This allows us to conclude that more complex the models become more time it need to finish the training. A reason behind that is the unidirectional LSTM only processes one input sequence in one direction only. In contrast, the bi-directional LSTM has to handle the reverse sequence of the input sequence thus increasing the training time.

4.4 Baseline Systems

We compared our results against the results mentioned in the paper [17]. In this paper, the authors proposed five paraphrase detection systems that detect paraphrases between two sentences. The first two systems mentioned in the paper are named "Multi-Perspective-CNN" and "Multi-Perspective-LSTM". These models use CNN and bi-directional LSTM for representing the context of the sentences. "Siamese-CNN"and "Siamese-LSTM" models use convolutional neural network and uni-directional LSTM to form the encoder respectively by following the siamese architecture. They measure the cosine similarity between the vectors obtained form the encoder for detecting the paraphrases. "L.D.C" model is the re-implementation of the model by [18]. In this model, two questions are individually passed through a bi-directional LSTM model to obtain two separate matching vectors. These two vectors are merged together into a single vector of axed dimension. Then this result vector is used for detecting the paraphrases in the prediction layer over the probabilistic distribution of the output obtained from the context of the given input questions.

4.5 Results

Table 1 shows various models and their accuracy. The results are sorted according to their accuracy and the names that are in bold highlight the models we implemented. From the results of Table 1, we observed that the complex models performed better than the simpler models. The Bi-Directional LSTM-Stacking with Attention model performed best among all other models. This model outperform the state of the art model BiPMP [17] by a little margin. The other models Bi-Directional LSTM-ATTN-CNN with maxpooling and LSTM with Attention also perform better than most of the models as shown in Table 1. We also show a comparison between the models we implemented: (1) LSTM, (2) bi-directional LSTM-ATTN-CNN with maxpooling, (3) LSTM with attention, and (4) bi-directional LSTM stacking with attention. From the results in table 2, we can conclude that Bi-directional LSTM-Stacking with Attention performs better than all of the other models. The hybrid model has 7% performance gain in overall than the basic model in accuracy. Attention over the word vectors is also playing a key factor for improving the performance.

Models	Accuracy (%)
Siamese-CNN [17]	79.60
Multi-Perspective-CNN [17]	81.38
LSTM	81.2
Siamese-LSTM [17]	82.58
Multi-Perspective-LSTM [17]	83.21
Bi-LSTM-ATTN-CNN	85
L.D.C. [18]	85.55
LSTM with Attention	87
BiMPM [17]	88.17
Bi-LSTM-Stacking with Attention	88.8

Table 1: Performance based on Accuracy for Question Duplication on the Quora dataset.

	(1)	(2)	(3)	(4)
Accuracy	81.2	85	87	88.8
Precision	77.7	84.2	85	85.8
Recall	79.4	76.4	81.4	86.2
F1 score	78.6	80.1	83	85.2

Table 2: Performance Comparisons of the implemented Models

5 CONCLUSIONS

In this paper we presented some deep learning methods, outperforming some other models for the task of finding similar questions. The models we discussed are independent of any manual feature engineering or external resources. These models can also be used for other tasks such as question-answer selection, answer ranking, etc. All of our models were based on the deep neural algorithms such as Long-Short Term Memory (LSTM), Convolutional Neural Network (CNN), attention mechanism, siamese network. We performed our experiments on Quora dataset for finding the question similarity. The hybrid models are performing better than the basic models. Attention over the word vectors is a key factor for improving the performance. In future works, we will try to develop more complex models that can handle noisy data and attain better semantic relationships between the words in the sentences.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their useful comments. The research reported in this paper was conducted at the University of Lethbridge and supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada discovery grant and the University of Lethbridge.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. CoRR abs/1409.0473 (2014). http://arxiv.org/abs/1409.0473

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. In Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 737–744. http://dl.acm.org/citation.cfm?id=2987189.2987282

[3] S. Chopra, R. Hadsell, and Y. LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1. 539–546 vol. 1. https://doi.org/10.1109/CVPR.2005.202

[4] Arpita Das, Harish Yenala, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese Networks for Similar Question Retrieval. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers. http://aclweb.org/anthology/P/P16/P16-1036.pdf

[5] Alex Graves, Navdeep Jaitly, and Abdel rahman Mohamed. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. In ASRU.

[6] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In Advances in Neural Information Processing Systems (NIPS). http://arxiv.org/abs/1506.03340

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

[8] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2042–2050.

[9] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[10] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Lisbon, Portugal, 1412–1421.

[11] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16). AAAI Press, 2786–2792. http://dl.acm.org/citation.cfm?id=3016100.3016291

[12] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. CoRR abs/1606.01933 (2016). http://arxiv.org/abs/1606.01933

[13] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar, 1532–1543. http://www.aclweb.org/anthology/D14-1162

[14] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about Entailment with Neural Attention. CoRR abs/1509.06664 (2015). http://arxiv.org/abs/1509.06664

[15] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Lisbon, Portugal, 379–389.

[16] Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for non-factoid answer selection. CoRR abs/1511.04108 (2015). http://arxiv.org/abs/1511.04108

[17] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. CoRR abs/1702.03814 (2017). http://arxiv.org/abs/1702.03814

[18] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence Similarity Learning by Lexical Decomposition and Composition. CoRR abs/1602.07019 (2016). http://arxiv.org/abs/1602.07019

[19] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. CoRR abs/1502.05698 (2015). http://arxiv.org/abs/1502.05698

[20] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In HLT-NAACL.

[21] Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive Interactive Neural Networks for Answer Selection in Community Question Answering. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. 3525–3531. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14611