ISSN 1870-4069

Fine-Grained Gating Based on Question-Summary for Machine Comprehension

Christian Mayhua Tijera, José Ochoa-Luna

Universidad Católica San Pablo, Department of Computer Science,
Arequipa, Peru
{christian.mayhua,jeochoa}@ucsp.edu.pe

Abstract. Currently, Deep Learning (DL) models for performing Machine Comprehension are very focused on Interaction Encoders, which are mostly based on attention mechanisms. DL models for encoding context, usually include pre-trained features into the initial embeddings, such as: Part-of-Speech, Named Entity Recognition, Term Frequency, etc. In this paper, we propose to use a fine-grained gating mechanism that controls the flow of information from the Context Encoder towards the Interaction Encoder. This gate is based on the Question-summary and the input vector at some time step. This simple structure has been shown to improve the performance of a given baseline model. Our model achieved 68.70% of exact match and 78.25% of F1 measure, on the Stanford question answering benchmark.

Keywords: machine comprehension, question answering, natural language processing, deep learning.

1 Introduction

Reading comprehension is defined as the ability to read text, process it, understand its meaning and then be able to answer any questions about it [4]. When machines perform this task it is called Machine Comprehension (MC). Although this definition may seem simple, this is a challenging task for machines.

Datasets have become very important in recent MC progress [5,15,16]. The predominant annotation style involves selecting a text segment from a Document to answer a Question posed in natural language. An example extracted from Stanford Question Answering Dataset (SQuAD) [15] is presented in Table 1.

SQuAD offers more realistic information and poses a greater challenge, due to many of the questions require commonsense reasoning and multi-sentence reasoning. Consider the question "Why did Tesla go to Karlovac?", presented in Table 1. This question requires multi-sentence reasoning.

Nowadays, MC models powered by Deep Learning (DL) become the state-of-the-art [7,13,23]. These models are basically composed by three modules: The first one is the Context Encoder that is responsible for encoding the words of the Document and the Question according to their surrounding words. The Interaction Encoder encodes the interaction between the Document and the

Table 1. Question-Answer pairs extracted from SQuAD. Each answer is a text span selected from the paragraph.

Nikola Tesla	
In 1870, Tesla moved to Karlovac, to **attend school at the Higher Real Gymnasium**, where he was profoundly influenced by a math teacher **Martin Sekulic**. The classes were held in **German**, as it was a school within the Austro-Hungarian Military Frontier. Tesla was able to perform integral calculus in his head, which prompted his teachers to believe that he was cheating. He finished a four-year term in three years, graduating in 1873.	
In what language were the classes given?	**German**
Who was Tesla's main influence in Karlovac?	**Martin Sekulic**
Why did Tesla go to Karlovac?	**attend school at the Higher Real Gymnasium**

Question. Finally, the Answer Decoder extracts the answer to the Question based on the previous encoding. Our proposal follows this approach. However, unlike most of MC works, which aim at improving the Interaction Encoder using attention, we seek to improve the Context Encoder instead.

In order to do so, we first transform words to their corresponding word embeddings. So as to feed them to a deep neural network, which can be some type of Recurrent Neural Network (RNN) [11] or Convolutional Neural Network (CNN) [9] , or a structured set of these. In such a way that the output is the encoding of each word with respect to its current context [27,29,12]. The context encoding of the Document and Question are mostly independently generated, although the same Context Encoder is for both shared.

Related works on Context Encoder, frequently add new features to the embeddings, such as: Part-of-Speech (POS), Named Entity Recognition (NER), Term Frequency (TF), etc. [1,12,25,29]. Conversely, we propose to include a simple structure called fine-grained gate based on Question-summary. The main idea is to regulate the flow of contextual information from the Document encoding towards the Interaction Encoder. Thus, when encoding the attention, we favor words that are the most relevant to answer the Question.

This simple structure has been shown to improve the performance regarding a given baseline model. Our model achieved 68.70 of Exact Match (EM) and 78.25 of F1 measure, on the SQuAD [15].

The remainder of this paper is organized as follows. In Section 2 the problem is formally defined and Section 3 presents a common pipeline shared by previous works. Section 4 reviews related work. Section 5 describes the proposed model. Section 6 presets experiments and results. Finally, Section 7 concludes the paper.

2 Machine Comprehension

In order to define the MC task considered in this work, we follow the SQuAD syntax. A Document (paragraph) and a Question are given as inputs. The Document is a sequence of m words $(p_1, p_2, ..., p_m)$ and a Question is another sequence of n words $(q_1, q_2, ..., q_n)$. The output is a set $\{a^s, a^e\}$, where $1 \leq a^s \leq a^e \leq m$ and a^s, a^e are the boundaries of the answer span, this means that $(p_{a^s}, p_{a^s+1}, ..., p_{a^e})$ is the answer extracted from the Document sequence. An example taken from SQuAD [15] can be seen in Table 1.

3 Deep Learning Pipeline

To tackle this task, one can adopt a generic pipeline [7,17,23]. In this setting, current DL models are composed by three modules: context encoder, interaction encoder and answer decoder.

3.1 Context Encoder

The Context Encoder is responsible for encoding words according to their current context, their surrounding words. The first step is to transform the words to their corresponding word embeddings [14]. Then, these embeddings are fed to a deep neural network, generally some kind of RNN, among the most used are: Long Short Term Memory (LSTM) [6] and Gated Recurrent Unit (GRU) [2]. The output of these RNNs is the encoding of each word according its current context.

3.2 Interaction Encoder

The Interaction Encoder merges the encoding context from the Document and the Question. Frequently, an attention mechanism is used to encode the interaction between the Question and the Document [3,17,26]. The attention can be given in only one direction, so as to focus the attention in parts of the Document according to the Question, or in both directions. Recently, self-attention is used as a second step of reasoning [23]. Thus, attention is focused on parts of the previous interaction encoding based on itself.

3.3 Answer Decoder

Finally, the Answer Decoder extracts a piece of text from the Document to answer the Question. To do so, pointer networks [21] are generally used. They have the ability to learn the conditional probability of a sequence based on another, this allow us to point to a position in the sequence. Usually two pointer networks are used to determine the boundaries of the answer given the interaction encoding [8,22,26].

4 Related Work

Related work regarding the Context Encoder are scarce. Most of the works have focused on improving the Interaction Encoder [7,17,23,24,26,28].

A vast majority of models included GloVe [14] as word embeddings and character-level embeddings based on CNN [9] or RNN [23]. Some models included another word features as inputs (e.g. POS, NER, TF). This simple pre-trained features added to initial embeddings improved the models [7,13,18].

Yang et al. [27] proposed a fine-grained gating mechanism to dynamically combine word and character-level embeddings based on properties of the words and additional features (POS, NER and TF). Their results showed to improve several Natural Language Processing (NLP) tasks. Our work proposes a similar gating approach but our gating mechanism is based on Question-summary and current Document input.

FastQA [25] introduced a simple context/type matching heuristic, which is based on the first word of the Question. They added two features directly in the embeddings, called word-in-question. jNet [29] introduced syntactic information to help to encode questions, due to there are several types of questions. jNet proposed an adaptive model for representing syntactic knowledge. Document Reader [1] included three simple binary features based on exact match metric.

Liu et al. [12] proposed to use structured linguistic information such as: constituency trees and dependency trees. We found that this model can perform especially well on exact match metrics, which requires syntactic information to accurately locate boundaries of answers.

5 The Model

In this section, we describe the proposed structure called fine-grained gate based on Question-summary which is included in the pipeline depicted in Figure 1.

First, we propose a Context Encoder that seeks the interaction between the Document and the Question in early stages. This Context Encoder consists of a gating mechanism that regulates the flow of information from the Document. Then, we define the attention based Interaction Encoder, which focuses on a subset of the Document where the answer is located. Finally, the Answer Decoder is responsible for predicting the beginning and the ending index of the answer inside the Document.

Further details about these stages are given as follows.

5.1 Context Encoder

The aim of this encoder is to transform sequences of words—Document and Question—into knowledge represented through sequences of vectors (knowledge representation). These vectors can be further used by layers that extract higher level knowledge, in order to answer the Question.

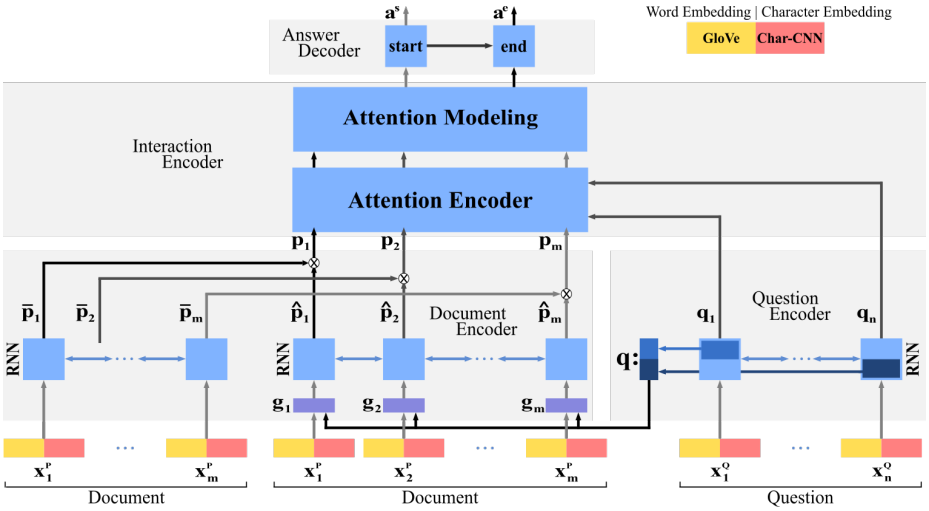

Fig. 1. DL model overview. The information flows from bottom to top. Below we have the Context Encoder, constituted by: Document Encoder and Question Encoder. In the central part is the Interaction Encoder, and finally up we have the Answer Decoder.

Word Embedding Layer. The word embedding layer maps each word to a high-dimensional vector space. We use pre-trained word vectors—GloVe [14]—to obtain the fixed word embedding of each word. Let $(x_1^{Qw}, x_2^{Qw}, ..., x_n^{Qw})$ be a vector which denotes the sequence of word vectors corresponding to words in the Question and $(x_1^{Pw}, x_2^{Pw}, ..., x_m^{Pw})$ denote the same for words in the Document.

Character Embedding Layer. The character embedding layer is responsible for mapping each word to a high-dimensional vector space. Let $(x_1^{Pc}, x_2^{Pc}, ..., x_m^{Pc})$ denote the Document words and $(x_1^{Qc}, x_2^{Qc}, ..., x_n^{Qc})$ denote the words in the Question. The character embeddings are generated using CNNs, following the proposed by Seo et al. [17], which is based on Kim's work [9].

Question Encoder. The input Question embedding is obtained by concatenating the character and word embeddings, which are fed to a two-layer Highway Network [20]. This input Question representation is denoted by $(x_1^Q, x_2^Q, ..., x_n^Q)$. Then, a bi-directional RNN computes the Question encoding. In forward direction $q_t^f = RNN_{forward}(q_{t-1}^f, x_t^Q)$ generates a matrix $Q^f = [q_1^f, q_2^f, ..., q_n^f] \in R^{d \times n}$. Similarly, we compute in backward direction $q_t^b = RNN_{backward}(q_{t+1}^b, x_t^Q)$ generating $Q^b = [q_1^b, q_2^b, ..., q_n^b] \in R^{d \times n}$. These vectors are concatenated so as to obtain a Question encoding $Q = [q_1, q_2, ..., q_n] \in R^{2d \times n}$.

In order to summarize the Question, we concatenate the last hidden state of the forward and backward RNNs. This Question summary is represented by the

following equation.

$$q = [q_n^f; q_1^b], \tag{1}$$

where (;) denotes concatenation. This summarized Question is the input for the proposed gate in the Document Encoder.

Document Encoder. Similarly to the Question Encoder, the Document Encoder relies on the concatenation of word and character embeddings, which are fed to a two-layer Highway Network [20], obtaining $(x_1^P, x_2^P, ..., x_m^P)$. Next, we use a bi-directional RNN to encode the Document embedding. For simplicity we denote the RNN in both directions as $\bar{p}_t = Bi - RNN(\bar{p}_{t-1}, \bar{p}_{t+1}, x_t^P)$, in order to obtain the Document encoding $\bar{P} = [\bar{p}_1, \bar{p}_2, ..., \bar{p}_m] \in R^{2d \times m}$.

Then, we propose a second processing branch based on a fine-grained gate, which allows us to regulate information flow from Document representation. Thus, we favor the relevant information to answer the Question before using attention. This gate is described in equation 2.

$$g_t = \sigma(W_g \cdot [x_t^P; q] + b_g), \tag{2}$$

where W_g, b_g are trainable parameters, q is the Question-summary, x_t^P is the current Document embedding and σ is the Sigmoid function. We apply the gate to each dimension of the previous Document embedding. Thus, we obtain its gated Document embeddings.

$$\hat{x}_t^P = g_t \circ x_t^P, \tag{3}$$

where $\circ$ is element-wise product. Similarly to Question Encoder, we use another a bi-directional RNN to encode the gated Document embedding, as follows: $\hat{p}_t = Bi - RNN(\hat{p}_{t-1}, \hat{p}_{t+1}, \hat{x}_t^P)$. In order to obtain its gated Document encoding $\hat{P} = [\hat{p}_1, \hat{p}_2, ..., \hat{p}_m] \in R^{2d \times m}$. Finally, we fuse Document encoding and gated Document encoding, which is defined in equation 4.

$$p_t = \bar{p}_t \times \hat{p}_t, \tag{4}$$

where $(\times)$ is an element-wise fuse operation. The element-wise product gave us better results. For simplicity, we denote the output of this Document Encoder as $P = [p_1, p_2, ..., p_m] \in R^{2d \times m}$.

5.2 Interaction Encoder

In this layer we encode the interaction between the Document and the Question. First, we use an attention mechanism, in this case we choose to use a Bi-Directional Attention Flow (BiDAF) layer [17]. This layer is defined as a function that fuses contextual information of Document P and Question Q to encode attention C.

$$C = Bi - Attention(P, Q). \tag{5}$$

The next step is the fusion of temporal information to the attention encoding so as to get attention modeling. We feed a two-layer of bidirectional RNN with C, to obtain a matrix $M \in R^{2d \times m}$. This allows us to provide an interaction representation of the Document and the Question. This interaction encoding feeds the Answer Decoder.

5.3 Answer Decoder

We use the output layer of the BiDAF model [17]. This allows us to predict beginning and ending positions of the span [22]. Pointer networks are used [21] to do so. The answer decoder returns a^s, a^e that denotes the boundaries of the answer span over the Interaction encoding M.

$$a^s = Answer - Decoder_1(M), \tag{6}$$

$$a^e = Answer - Decoder_2(M). \tag{7}$$

6 Results

This section presents results related to the gating mechanisms included into the Context Encoder. Tests were made with several gate variations. In this case, only models that improved the reported baseline results are shown, refer to Table 2.

6.1 Implementation Details

We train and evaluate the different models using SQuAD [15], a dataset for MC that contains about 100K question-answer pairs. In order to answer the Question, the aim is to extract a text span from a paragraph extracted from Wikipedia articles. For evaluating, SQuAD uses two metrics: Exact Match and F1 measure. We divide this dataset in 90K and 10K tuples for train and dev respectively.

Our baseline model is BiDAF [17]. We tokenize each Document and Question using PTB Tokenizer in order to feed the model. All RNNs are LSTM [6] whose hidden state size is 100. We use Adam [10] optimizer, with default parameters, an initial learning rate of 0.001, exponential decay rate of 0.999 and dropout [19] rate of 0.2, for 12 epochs, with a mini-batch size of 60. The training process took between 18 and 24 hours on a single Tesla K80 GPU in the *Manati* cluster[1].

6.2 Experiments

We conveniently group the experiments in three sets. The first set presents simple variations into the Context Encoder of the baseline model. The second set denotes positional variations of the gating mechanism in the Document

[1] *Manati* is a cluster located in Center for High Computational Performance of the Peruvian Amazon. http://iiap.org.pe/web/carcap.aspx

Encoder pipeline. In the third set, we introduce a parallel processing branch into the Document Encoder, with the aim of using both parallel Document representations in the following layers.

In the first set, we removed the 2-layer Highway network included into the baseline Document Encoder ($D_{highway}$). We removed it from Question Encoder ($Q_{highway}$) and it was also removed from both at the same time. Table 2 shows that only removing $D_{highway}$ from the Document Encoder pipeline (not from the Question Encoder) presented the better results for this set. Apparently, $Q_{highway}$ compensates the difference of lengths between Document and Question, when both are computed by the shared-weights LSTM. Given that the sequence of the Document is much longer than the Question.

In the second set, we introduced the fine-grained gating mechanism into the Document Encoder. We put the gate in four different position: before $D_{highway}$, replacing $D_{highway}$, after $D_{highway}$ and after D_{LSTM}. Where D_{LSTM} represents the LSTM of the Document Encoder. In Table 2, the best results arose when we added the gate after D_{LSTM}. We concluded that the gating mechanism successfully controlled the flow of information from Context Encoder towards Interaction Encoder, highlighting correctly Document words relevant to the Question.

In the third set, we introduced a parallel processing branch composed by: a gating mechanism and another LSTM. The LSTM encodes the gated embeddings produced by the gate over Document embeddings. At this step, we have two different Document representations from two different LSTMs. The next step is to fuse both Document representations. In order to feed the Interaction Encoder, we proposed three fussing operations: element-wise addition (+), element-wise product (∘) and concatenation (;). In Table 2, element-wise product (∘) gave us better results. This is because it also behaves like a kind of second fine-grained gate, i.e., it also regulates the flow of information from the Document Encoder towards the Interaction Encoder.

Table 2. Results on SQuAD development set.

Model (Single)	**EM**	**F1**
BiDAF [17]	67.70	77.30
SEDT-LSTM [12]	68.13	77.58
Our baseline implementation	68.14	77.61
$-Q_{highway}-D_{highway}$	67.68	77.43
$-Q_{highway}$	68.12	77.45
$-D_{highway}$	68.26	77.81
+Gate before $D_{highway}$	68.03	77.63
+Gate instead $D_{highway}$	68.20	77.65
+Gate after $D_{highway}$	67.96	77.43
+Gate after D_{LSTM}	68.34	77.68
+Gated branch, fused with (+)	68.31	77.76
+Gated branch, fused with (∘)	**68.70**	**78.25**
+Gated branch, fused with (;)	68.46	77.84

Given the results showed in Table 2, our structure added into the baseline Document Encoder has proven to improve the results by almost 1% in both metrics, where element-wise product ($\circ$) as fusing operation gave us the best results across all models.

6.3 Statistical Significance Testing

We perform a Student's t-test for proving that our improvement is statistically significant. We use independent two-sample t-test over 10 samples with unequal variances. We raise a null hypothesis H_0 : there is no significance difference between the mean of different samples.

We use the following decision criterion: if $t_{score} \leq \alpha$ reject H_0, else accept H_0, with an $\alpha = 0.05$ significance value. Table 3 shows the t_{score} of our model over our baseline implementation, both are less than α. Thus, we refuse the null hypothesis. It means that the difference is statistically significant at 95%.

Table 3. Student's t-test results on SQuAD development set.

Model (Single)	**EM**	t_{score}^{EM}	**F1**	t_{score}^{F1}
Our baseline implementation	67.95		77.54	
+Gated branch, fused with ($\times$)	68.34	2.3×10^{-3}	77.76	1.4×10^{-2}

7 Conclusion

Nowadays, Machine Comprehension is mostly approached using Deep Learning. The vast majority of these models focus on improving the Interaction Encoder which is strongly based on a given attention mechanism. In contrast, less effort has been spent to improve the Context Encoder. Thus, in this paper we have explored the Context Encoder. In this sense, we have proposed a gating mechanism that allowed us to regulate the flow of information from the Document, which is directly dependent on the Question. By doing so, we were able to highlight words that were relevant to answer the Question.

Experiments were performed on a benchmark dataset, SQuAD. Reported results were promising, the gating mechanism allowed us to outperform a given baseline model. We obtained 68.70% of EM metric and 78.25% of F1 score.

Acknowledgments. This work was supported by grant 234-2015-FONDECYT (Master Program) from Cienciactiva of the National Council for Science, Technology and Technological Innovation (CONCYTEC-PERU).

References

1. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. vol. 1 (Long Papers), pp. 1870–1879 (2017)
2. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1724–1734. Association for Computational Linguistics (2014)
3. Dhingra, B., Liu, H., Yang, Z., Cohen, W., Salakhutdinov, R.: Gated-attention readers for text comprehension. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. vol. 1 (Long Papers), pp. 1832–1846. Association for Computational Linguistics (2017)
4. Grabe, W.: Reading in a Second Language: Moving from Theory to Practice. Cambridge Applied Linguistics, Cambridge University Press (2009)
5. Hermann, K.M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. pp. 1693–1701. NIPS'15, MIT Press, Cambridge, MA, USA (2015)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997)
7. Hu, M., Peng, Y., Huang, Z., Qiu, X., Wei, F., Zhou, M.: Reinforced mnemonic reader for machine reading comprehension. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 4099–4106. International Joint Conferences on Artificial Intelligence Organization (2018)
8. Kadlec, R., Schmid, M., Bajgar, O., Kleindienst, J.: Text understanding with the attention sum reader network. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. vol. 1 (Long Papers), pp. 908–918. Association for Computational Linguistics (2016)
9. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. pp. 1746–1751. Association for Computational Linguistics (2014)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
12. Liu, R., Hu, J., Wei, W., Yang, Z., Nyberg, E.: Structural embedding of syntactic trees for machine comprehension. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 826–835. Association for Computational Linguistics (2017)
13. Liu, X., Shen, Y., Duh, K., Gao, J.: Stochastic answer networks for machine reading comprehension. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. vol. 1 (Long Papers), pp. 1694–1704 (2018)
14. 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. pp. 1532–1543. Association for Computational Linguistics (2014)
15. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on

Empirical Methods in Natural Language Processing. pp. 2383–2392. Association for Computational Linguistics (2016)
16. Richardson, M., Burges, C.J., Renshaw, E.: Mctest: A challenge dataset for the open-domain machine comprehension of text. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 193–203. Association for Computational Linguistics (2013)
17. Seo, M.J., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bi-directional attention flow for machine comprehension. In: International Conference on Learning Representations (2017)
18. Shen, Y., Liu, X., Duh, K., Gao, J.: An empirical analysis of multiple-turn reasoning strategies in reading comprehension tasks. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing. vol. 1 (Long Papers), pp. 957–966. Asian Federation of Natural Language Processing (2017)
19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958 (2014)
20. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. pp. 2377–2385. NIPS'15, MIT Press, Cambridge, MA, USA (2015)
21. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Advances in Neural Information Processing Systems 28, pp. 2692–2700. Curran Associates, Inc. (2015)
22. Wang, S., Jiang, J.: Machine comprehension using match-lstm and answer pointer. In: International Conference on Learning Representations (2017)
23. Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M.: Gated self-matching networks for reading comprehension and question answering. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. vol. 1 (Long Papers), pp. 189–198 (2017)
24. Wang, Z., Mi, H., Hamza, W., Florian, R.: Multi-perspective context matching for machine comprehension. CoRR abs/1612.04211 (2016)
25. Weissenborn, D., Wiese, G., Seiffe, L.: Making neural qa as simple as possible but not simpler. In: Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). pp. 271–280. Association for Computational Linguistics (2017)
26. Xiong, C., Zhong, V., Socher, R.: Dynamic coattention networks for question answering. In: International Conference on Learning Representations (2017)
27. Yang, Z., Dhingra, B., Yuan, Y., Hu, J., Cohen, W.W., Salakhutdinov, R.: Words or characters? fine-grained gating for reading comprehension. In: International Conference on Learning Representations (2017)
28. Yu, Y., Zhang, W., Hasan, K.S., Yu, M., Xiang, B., Zhou, B.: End-to-end answer chunk extraction and ranking for reading comprehension. CoRR abs/1610.09996 (2016)
29. Zhang, J., Zhu, X., Chen, Q., Ling, Z., Dai, L., Wei, S., Jiang, H.: Exploring question representation and adaptation with neural networks. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC). pp. 1975–1984 (2017)