Attention-Based Convolutional Neural Network for Machine Comprehension

Wenpeng Yin, Sebastian Ebert, Hinrich Schütze
LMU Munich, Germany
wenpeng, ebert@cis.lmu.de

Abstract

Understanding open-domain text is one of the primary challenges in NLP. Machine comprehension benchmarks evaluate a system's ability to understand text based on the text content only. In this work, we investigate machine comprehension on MCTest, a question answering (QA) benchmark. Prior work is mainly based on feature engineering approaches. We come up with a neural network framework, named hierarchical attention-based convolutional neural network (HABCNN), to address this task without any manually designed features. Specifically, we explore HABCNN for this task by two routes, one is through traditional joint modeling of document, question and answer, one is through textual entailment. HABCNN employs an attention mechanism to detect key phrases, key sentences and key snippets that are relevant to answering the question. Experiments show that HABCNN outperforms prior deep learning approaches by a big margin.

1 Introduction

Machine comprehension is an open-domain question-answering problem which contains factoid questions, but the answers can be derived by extraction or induction of key clues. Figure 1 shows one example in MCTest (Richardson et al., 2013). Each example consists of one document, four associated questions; each question is followed by four answer candidates of which only one is correct. Questions in MCTest have two categories; "one" questions can be answered based on a single sentence from document where "multiple" questions require several

The road to Grandpa's house was long and winding. [...] Jimmy liked to collect insects on the way to his Grandpa's house, so had picked the longer path. As he went along, Jimmy found more and more insects to add to his jar. [...]. Finally, Jimmy arrived at Grandpa's house and knocked. Grandpa answered the door with a smile and welcomed Jimmy inside. They sat by the fire and talked about the insects. They watched the lightning bugs light up as night came.

1: multiple: Why did Grandpa answer the door?
A) Because he saw the insects
B) Because Jimmy was walking
*C) Because Jimmy knocked
D) Because the trip took a long time

2: one: Where do Jimmy and his Grandpa sit?
A) On insects
B) Outside
*C) By the fire
D) On the path

Figure 1: One example with 2 out of 4 questions in the MCTest. "*" marks correct answer.

sentences. To correctly answer the first question in the example, the two blue sentences are required; for the second question instead, we only need the red sentence. The following observations hold for the whole MCTest. (i) Most of the sentences in the document are irrelevant for a given question. It hints that we need to pay attention to just some key regions. (ii) Answer candidates vary in length and abstraction level and usually do not appear in the document. For example, candidate B for the second question is "outside", which is one word and does not exist in the document, while the answer candidates for the first question are longer texts with some auxiliary words like "Because" in the text. This requires our system to handle flexible texts via *extraction* as well as *abstraction*. (iii) Some questions require multiple sentences to infer the answer, and those vital sentences mostly appear close to each other (we call them *snippet*). Hence, our system should be able to make a choice or compromise between potential single-sentence clue

Proceedings of 2016 NAACL Human-Computer Question Answering Workshop, pages 15–21,
San Diego, California, June 12-17, 2016.

and snippet clue.

Prior work is mostly based on feature engineering. We take the lead in presenting a deep neural network without linguistic feature engineering.

Concretely, we propose HABCNN, a hierarchical attention-based convolutional neural network, to address this task in two roadmaps. In the first one, we project the document in two different ways, one based on question-attention, one based on answer-attention and then compare the two projected document representations to determine whether the answer matches the question. In the second one, every question-answer pair is reformatted into a statement, then the whole task reduces to textual entailment.

In both roadmaps, convolutional neural network (CNN) is explored to model all types of text. As human beings usually do for such a QA task, our model is expected to be able to detect the key snippets, key sentences, and key words or phrases in the document. In order to detect those informative parts required by questions, we explore an attention mechanism to model the document so that in its representation the required information is emphasized. In practice, instead of imitating human beings in QA task top-down, our system models the document bottom-up, through accumulating the most relevant information from word level to snippet level.

Our approach is novel in three aspects. (i) A document is modeled by a hierarchical CNN for different granularity, from word to sentence level, then from sentence to snippet level. (ii) In the example in Figure 1, apparently not all sentences are required given a question, and usually different snippets are required by different questions. Hence, the same document should have different representations based on what the question is. To this end, attention is incorporated into the hierarchical CNN to guide the learning of dynamic document representations which closely match the information requirements by questions. (iii) Document representations at sentence and snippet levels both are informative for the question. Therefore a highway network is developed to combine them, enabling our system to make a flexible tradeoff.

Overall, we make three contributions. (i) We present a hierarchical attention-based CNN system "HABCNN". It is, to our knowledge, the first deep learning (DL) based system for this MCTest task.

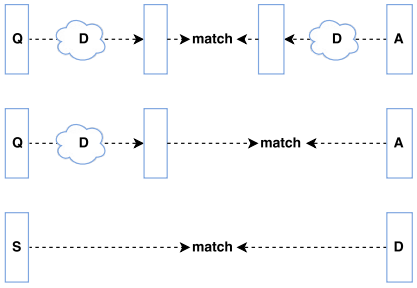

Figure 2: Illustrations of HABCNN-QAP (top), HABCHH-QP (middle) and HABCNN-TE (bottom). Q, A, S: question, answer, statement; D: document

(ii) Prior document modeling systems based on deep neural networks mostly generate generic representation, this work is the first to incorporate attention so that document representation is biased towards the question requirement. (iii) Our HABCNN systems outperform other DL competitors by big margins.

2 Related Work

Existing systems for MCTest are mostly based on manually engineered features, e.g., (Narasimhan and Barzilay, 2015; Sachan et al., 2015; Wang et al., 2015; Smith et al., 2015). In these works, a common route is first to define a loss function based on feature vectors, then the effort focuses on designing effective features based on various rules. Even though this research is groundbreaking for this task, its flexibility and capacity for generalization is limited.

DL approaches appeal to increasing interest in analogous tasks. Weston et al., (2015b) introduce memory networks for factoid QA. Memory network framework is extended in (Weston et al., 2015a; Kumar et al., 2016) for Facebook bAbI dataset. Peng et al. (2015)'s Neural Reasoner infers over multiple supporting facts to generate an entity answer for a given question and it is also tested on bAbI. All of this work deals with short texts with simple grammar, aiming to *generate* an answer that is restricted to be one word denoting a location, a person etc.

There is also work on other kinds of QA, e.g., (Iyyer et al., 2014; Hermann et al., 2015). Overall, for open-domain MCTest machine comprehension task, we are the first to use deep neural networks.

HABCNN shares similarities with the model published by Trischler et al. (2016) six weeks after our submission on arxiv. It considers multiple levels of granularity in a way that is similar to our approach.

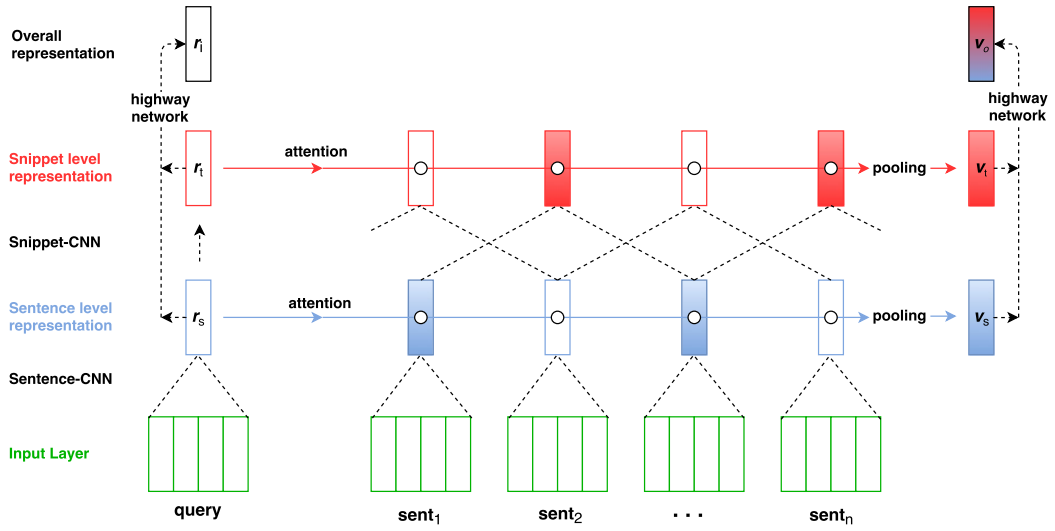

Figure 3: HABCNN. Feature maps for phrase representations $\mathbf{p}_i$ and the max pooling steps that create sentence representations out of phrase representations are omitted for simplification. Each snippet covers three sentences in snippet-CNN. Symbols ∘ mean cosine similarity calculation.

Trischler et al. (2016) achieve better performance than HABCNN, but they still use linguistically engineered features like Stanford dependencies whereas our approach is more truly end-to-end.

3 Model

We investigate this task by three approaches, illustrated in Figure 2. (i) We can compute two different document (D) representations in a common space, one based on question (Q) attention, one based on answer (A) attention, and compare them. This architecture is named HABCNN-QAP ("QAP" means projecting D based on Q and A). (ii) We compute a representation of D based on Q attention (as before), but now we compare it directly with a representation of A. We name this architecture HABCNN-QP. (iii) We treat this QA task as textual entailment (TE), first reformatting Q-A pair into a statement (S), then matching S and D directly. This architecture we name HABCNN-TE. All three approaches are implemented in the common framework HABCNN.

3.1 HABCNN

Recall that we use the abbreviations A (answer), Q (question), S (statement), D (document). HABCNN performs representation learning for triple (Q, A, D) in HABCNN-QP and HABCNN-QAP, for tuple (S, D) in HABCNN-TE. For convenience, we use "query" to refer to Q, A, or S uniformly. HABCNN, depicted in Figure 3, has the following phases.

Input Layer. The input is (query, D). Query is two individual sentences (for Q, A) or one single sentence (for S), D is a sequence of sentences. Words are initialized by d-dimensional pre-trained word embeddings. As a result, each sentence is represented as a feature map with dimensionality of $d \times s$ (s is sentence length). In Figure 3, each sentence in the input layer is depicted by a rectangle with multiple columns.

Sentence-CNN is used for sentence representation learning from word level. Given a sentence of length s with a word sequence: $v_1, v_2, \ldots, v_s$, let vector $\mathbf{c}_i \in \mathbb{R}^{wd}$ be the concatenated embeddings of w words $v_{i-w+1}, \ldots, v_i$ where w is the filter width, d is the dimensionality of word representations and $0 < i < s + w$. Embeddings for words v_i, $i < 1$ and $i > s$, are zero padding. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d_1}$ for the *phrase* $v_{i-w+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d_1 \times wd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \quad (1)$$

where bias $\mathbf{b} \in \mathbb{R}^{d_1}$. CNN has $d_1 = s + w - 1$ kernels. Sentence-CNNs for query and all document sentences share the same weights, so that the generated representations are comparable.

Sentence-Level Representation. The sentence-CNN generates a new feature map (omitted in Figure 3) for each input sentence, one column in the

feature map denotes a phrase representation (i.e., $\mathbf{p}_i$ in Equation (1)).

For the query and each sentence of D, we do *element-wise 1-max-pooling* ("max-pooling" for short) (Collobert and Weston, 2008) over phrase representations to form their representations at this level.

We now treat D as a set of "vital" sentences and "noise" sentences. We propose *attention-pooling* to learn the sentence-level representation of D as follows: first identify vital sentences by computing attention for each of D's sentences as the cosine similarity between its representation and the query representation, then select the k highest-attention sentences to do max-pooling over them. Taking Figure 3 as an example, based on the output of the sentence-CNN layer, $k = 2$ important sentences with blue color are combined by max-pooling as the sentence-level representation $\mathbf{v}_s$ of D; the other – white-color – sentence representations are neglected as they have low attention. (If $k = all$, attention-pooling returns to the common max-pooling in (Collobert and Weston, 2008).) When the query is (Q, A), this step will be repeated, once for Q, once for A, to compute representations of D at the sentence level that are biased with respect to Q and A, respectively.

Snippet-CNN. As the example in Figure 1 shows, to answer the first question "why did Grandpa answer the door?", it does not suffice to compare this question only to the sentence "Grandpa answered the door with a smile and welcomed Jimmy inside"; instead, the snippet "Finally, Jimmy arrived at Grandpa's house and knocked. Grandpa answered the door with a smile and welcomed Jimmy inside" should be used to compare. To this end, it is necessary to stack another CNN layer, *snippet-CNN*, to learn representations of snippets, i.e., units of one or more sentences. Thus, input to snippet-CNN (resp. sentence-CNN) are sentences (resp. words) and the output is representations of snippets (resp. sentences).

Concretely, snippet-CNN puts all sentence representations in column sequence as a feature map and conducts another convolution operation over it. With filter width w, this step generates representation of snippet with w consecutive sentences. Similarly, we use the same CNN to learn higher-abstraction query representations, treating query as a document with one sentence, so that the higher-abstraction query representation is in the same space with corresponding snippet representations.

Snippet-Level Representation. For the output of snippet-CNN, each representation is more abstract and denotes bigger granularity. We apply the same attention-pooling process to snippet level representations: attention values are computed as cosine similarities between query and snippets and the snippets with the k largest attention are retained. Max-pooling over the k selected snippet representations then creates the snippet-level representation $\mathbf{v}_t$ of D. Two selected snippets are shown as red in Figure 3.

Overall Representation. Based on convolution layers at two different granularity, we have derived query-biased representations of D at sentence ($\mathbf{v}_s$) and snippet ($\mathbf{v}_t$) levels. In order to create a flexible choice for open QA, we develop a highway network (Srivastava et al., 2015) to combine the two levels of representations as an overall representation $\mathbf{v}_o$ of D:

$$\mathbf{v}_o = (1 - \mathbf{h}) \odot \mathbf{v}_s + \mathbf{h} \odot \mathbf{v}_t \tag{2}$$

where highway network weights $\mathbf{h}$ are learned by

$$\mathbf{h} = \sigma(\mathbf{W}_h \mathbf{v}_s + \mathbf{b}) \tag{3}$$

where $\mathbf{W}_h \in \mathbb{R}^{d_1 \times d_1}$. With the same highway network, we can generate the overall query representation, $\mathbf{r}_i$ in Figure 3, by combining query's representation at sentence level r_s and at snippet level r_t.

3.2 HABCNN-QP & HABCNN-QAP

HABCNN-QP/QAP computes the representation of D as a projection of D, either based on attention from Q or based on attention from A. We hope that these two projections of the document are close for a correct A and less close for an incorrect A. As we said in related work, machine comprehension can be viewed as an answer selection task *using the document D as critical background information*. Here, HABCNN-QP/QAP do not compare Q and A directly, but they use Q and A to filter the document differently, extracting what is critical for the Q/A match by attention-pooling. Then they match the two document representations in the new space.

For simplicity, we have used the symbol $\mathbf{v}_o$ so far, but in HABCNN-QP/QAP we compute two different document representations: $\mathbf{v}_{oq}$, for which attention is computed with respect to Q; and $\mathbf{v}_{oa}$ for

which attention is computed with respect to A. $\mathbf{r}_i$ also has two versions, one for Q: $\mathbf{r}_{iq}$, one for A: $\mathbf{r}_{ia}$.

HABCNN-QP and HABCNN-QAP make different use of $\mathbf{v}_{oq}$. HABCNN-QAP projects D twice, once based on attention from Q, once based on attention from A and compares the two projected representations, $\mathbf{v}_{oq}$ and $\mathbf{v}_{oa}$, shown in Figure 2 (top). HABCNN-QP only utilizes the Q-based projection of D and then compares the projected document $\mathbf{v}_{oq}$ with the answer representation $\mathbf{r}_{ia}$, shown in Figure 2 (middle).

3.3 HABCNN-TE

HABCNN-TE treats machine comprehension as *textual entailment*. We use the statements that are provided in MCTest. Each statement corresponds to a question-answer pair; e.g., the Q/A pair "Why did Grandpa answer the door?" / "Because he saw the insects" (Figure 1) is reformatted into the statement "Grandpa answered the door because he saw the insects". The question answering task is then cast as: "does the document entail the statement?"

For HABCNN-TE, shown in Figure 2 (bottom), the input for Figure 3 is the pair (S, D). HABCNN-TE tries to match S's representation $\mathbf{r}_i$ with D's representation $\mathbf{v}_o$.

4 Experiments

4.1 Dataset

MCTest[1] has two subsets. MCTest-160 contains 160 items (70 train, 30 dev, 60 test), each consisting of a document, four questions followed by one correct anwer and three incorrect answers and MCTest-500 500 items (300 train, 50 dev, 150 test).

4.2 Training Setup

Our training objective is to minimize the following ranking loss function:

$$L(d, a^+, a^-) = \max(0, \alpha + S(d, a^-) - S(d, a^+)) \quad (4)$$

where $S(\cdot, \cdot)$ is a matching score between two representation vectors. Cosine similarity is used throughout. α is a constant.

Multitask learning. Based on work showing that question typing is helpful for QA (Sachan et al., 2015), we stack a logistic regression layer over question representation $\mathbf{r}_{iq}$, with the purpose that this subtask can favor the parameter tuning of the whole system, and finally the question is better recognized and answer identification is more accurate.

To be specific, we classify questions into 12 classes: "how", "how much", "how many", "what", "who", "where", "which", "when", "whose", "why", "will" and "*other*". The question label is created by querying for the label keyword in the question. If more than one keyword appears in a question, we adopt the one appearing earlier and the more specific one (e.g., "how much", not "how"). In case there is no match, the class "*other*" is assigned.

We train with AdaGrad (Duchi et al., 2011) and use 50-dimensional GloVe embeddings (Pennington et al., 2014) to initialize word representations,[2] kept fixed during training. Table 1 gives hyperparameter values, tuned on dev.

k	lr	d_1	bs	w	L_2	α
[1,3]	0.05	[90, 90]	1	[2,2]	0.0065	0.2

Table 1: Hyperparameters. k: top-k in attention-pooling for both CNN layers; lr: learning rate; d_1: number of kernels in CNN layers; bs: mini-batch size; w: filter width; L_2: L_2 regularization; α: constant in loss function.

We consider two evaluation metrics: *accuracy* (proportion of questions correctly answered) and *NDCG*$_4$ (Järvelin and Kekäläinen, 2002). Unlike accuracy which evaluates if the question is correctly answered or not, NDCG$_4$, being a measure of ranking quality, evaluates the position of the correct answer in our predicted ranking.

4.3 Baseline Systems

(i) **Addition.** Directly compare question and answers without considering the document. Sentence representations are computed by element-wise addition over word representations. (ii) **Addition-proj.** First compute sentence representations for Q, A and all D sentences as in *Addition*. Then identify two sentences from D, taking $\mathbf{x}_q$ and $\mathbf{x}_a$ as example, satisfying that they are most similar to Q and A, respectively. The match score between Q and A is then the cosine similarity of $\mathbf{x}_q$ and $\mathbf{x}_a$. (iii) **NR.** The Neural Reasoner (Peng et al., 2015) has an en-

[1] http://requery.microsoft.com/mct

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

	method	MCTest-150						MCTest-500					
		acc			$NDCG_4$			acc			$NDCG_4$		
		one	mul	all	one	mul	all	one	mul	all	one	mul	all
Baselines	Addition	39.3	32.4	35.7	60.4	50.3	54.6	35.7	30.2	32.9	56.6	55.2	55.8
	Addition-proj	42.1	38.7	40.3	65.3	61.3	63.2	39.4	36.7	38.0	63.3	60.1	61.7
	AR	48.1	44.7	46.3	70.5	68.9	69.6	44.4	39.5	41.9	66.7	64.2	65.4
	NR	48.4	46.8	47.6	70.7	68.2	69.7	45.7	45.6	45.6	71.9	69.5	70.6
	HABCNN-QP	57.9	53.7	55.7	80.4	80.0	80.2	53.7	46.7	50.1	75.4	72.7	74.0
	HABCNN-QAP	59.0	57.9	58.4	81.5	79.9	80.6	54.0	47.2	50.6	75.7	72.6	74.1
	HABCNN-TE	**63.3**	**62.9**	**63.1**	**86.6**	**85.9**	**86.2**	**54.2**	**51.7**	**52.9**	**76.1**	**74.4**	**75.2**
	Sachan et al. (2015)	–	–	–	–	–	–	67.6	67.9	67.8	86.7	86.9	86.8
	Wang et al. (2015)	84.2	67.8	75.2	–	–	–	72.0	67.9	69.9	–	–	–

Table 2: Experimental results for one-sentence (one), multiple-sentence (mul) and all cases.

coding layer, multiple reasoning layers and a final answer layer. The input for the encoding layer is a question and the sentences of the document (called facts); each sentence is encoded by a GRU into a vector. In each reasoning layer, NR lets the question representation interact with each fact representation as reasoning process. Finally, all temporary reasoning clues are pooled as answer representation. (iv) **AR.** The Attentive Reader (Hermann et al., 2015) is implemented by modeling the whole D as a word sequence – without specific sentence / snippet representations – using an LSTM. Attention mechanism is implemented at word representation level.

Baselines Addition(-proj) do not involve complex composition and inference. NR and AR are the top-performing deep neural networks for QA.

4.4 Results

Table 2 lists the performance of baselines, HABCNN systems, and two top-performing non-DL systems (Sachan et al. (2015), Wang et al. (2015)) in the first, second, and last block, respectively. Consistently, our HABCNN systems outperform all baselines, especially surpass the two competitive DL systems AR and NR. The margin between our best-performing ABHCNN-TE and NR is 15.5/16.5 (accuracy/NDCG) on MCTest-150 and 7.3/4.6 on MCTest-500. This demonstrates the promise of our architecture in this task.

As said before, both AR and NR systems aim to *generate* answers in entity form. Their designs might not suit this machine comprehension task, in which the answers are openly-formed based on summarizing or abstracting the clues. To be more specific, AR models D always at word level, attention is also paid to corresponding word representations, which is applicable for entity-style answers, but is less suitable for comprehension at sentence level or even snippet level. NR contrarily models D in sentence level always, neglecting the discovering of key phrases which however compose most of the answers. In addition, the attention of AR system and the question-fact interaction in NR system both bring large numbers of parameters, this potentially constrains their power in a dataset of limited size.

The size of MCTest is quite small. This is the most likely reason for the inferior performance of all deep learning approaches compared to non-deep-learning approaches. If the amount of training data is limited, then it may not be possible to get top performance without a large feature engineering effort.[3]

5 Conclusion

This work takes the lead in presenting a CNN based neural network system for open-domain machine comprehension task. Our systems try to solve this task in a document projection way as well as a textual entailment way. The latter demonstrates slightly better performance. Overall, our architecture, modeling dynamic document representation by attention scheme from sentence level to snippet level, shows promising results in this task. In the future, more fine-grained representation learning approaches are expected to model complex answer types and question types.

Acknowledgments. We thank DFG for supporting this work (grant SCHU 2246/4-2).

[3]Please refer to the extended version at arxiv.org/pdf/1602.04341v1.pdf for attention visualization and error analysis.

References

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*, pages 1684–1692.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of EMNLP*, pages 633–644.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of ICML*.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings ACL*, pages 1253–1262.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. *CoRR*, abs/1508.05508.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of EMNLP*, 12:1532–1543.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

Mrinmaya Sachan, Kumar Dubey, Eric P. Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of ACL*, pages 239–249.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of EMNLP*, pages 1693–1698.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NIPS*, pages 2368–2376.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL-IJCNLP*, pages 700–706.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015a. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. In *Proceedings of ICLR*.