CSReader at SemEval-2018 Task 11: Multiple Choice Question Answering as Textual Entailment

Zhengping Jiang
Laboratory of Machine Perception
Peking University
tony.jiang.zhengping@gmail.com

Qi Sun
Laboratory of Machine Perception
Peking University
1500012917@pku.edu.cn

Abstract

In this document we present an end-to-end machine reading comprehension system that solves multiple choice questions with a textual entailment perspective. Since some of the knowledge required is not explicitly mentioned in the text, we try to exploit common sense knowledge by using pretrained word embeddings during contextual embeddings and by dynamically generating a weighted representation of related script knowledge. In the model two kinds of prediction structure are ensembled, and the final accuracy of our system is 10 percent higher than the naiive baseline.

1 Introduction

K. M. Arivuchelvan et al.(2017) stated that machine reading comprehension can be defined as a task that deals with the automatic understanding of texts. In their paper, it was also mentioned that machine comprehension can be evaluated by two methods, namely (1) translating the text into formal language representations and evaluating it using structured queries. (2) evaluating it through natural language questions. Recently a lot of datasets are available for evaluating machine reading comprehension systems, for example, there are SQuAD(Rajpurkar et al., 2016) and the MCTest(Richardson et al., 2013). On many of these datasets human-like performance has been achieved. However, one of the biggest challenges in machine comprehension is how to provide common sense knowledge regarding daily events to machines(Mostafazadeh et al., 2016). The SemEval2018-Task11(Ostermann et al., 2018) provides a dataset containing questions that can only be answered with the help of common sense knowledge.To address this problem, we first propose a model to solve normal reading comprehension problems and then try to modify the model to embody common sense knowledge. In section two, we give a brief introduction to ideas and models that might be useful to a comprehensive understanding of our work. Section three carefully describes our model implementation, and why we chose this kind of model structure. And section four briefly examines the datasets used. Section five provides a simple evaluation of our result. Finally, our conclusion can be found in section six.

2 Background Knowledge

In this section, we present some basic knowledge required for a comprehensive understanding of our model. We first give a basic introduction to RNN models and the implementation of GRU Cell, then cast a little glance upon the textual entailment problems.

2.1 Recurent Neural Network

Lipton et al.(2015) stated that Recurrent neural networks (RNNs) are “connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time.” An RNN model captures features of a sequence by updating a hidden set of variables at every input element, as illustrated in figure 1. In many language modelling tasks, the output of every RNN iteration step will simply be a prediction of the one-hot representation of the next element in the sequence. However, as we intended to train our model end-to-end, we will not care much about the output of the RNN model, but pay attention instead to the hidden state because it is likely to be a contextual representation influenced by every input element.

2.2 GRU Cell

One of the commonly used RNN hidden units is LSTM (Hochreiter and Schmidhuber, 1997). This kind of hidden unit can retain short-term memory for a long time during sequence processing, thus

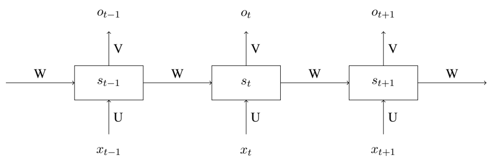

Figure 1: An illustration of unfolded RNN cells.

is able to recognize long-term dependency information. GRU Cell (Cho et al., 2014) is inspired by LSTM, and is simpler to compute. For each timestep t, two gating vector is computed, i.e., the reset gate r and the update gate z by

$$\begin{aligned} r_j &= \sigma([\mathrm{W}_r x]_j + [\mathrm{U}_r h^{<t-1>}]_j) \\ z_j &= \sigma([\mathrm{W}_z x]_j + [\mathrm{U}_z h^{<t-1>}]_j) \end{aligned}$$

Where x is the input vector and h is the hidden-state vector, and $[\cdot]_j$ means the jth element of a given vector. $\mathrm{W}_{(\cdot)}$ $\mathrm{U}_{(\cdot)}$ are matrix parameters to be trained. With these variables defined, the hidden state can be updated as

$$h_j^{<t>} = z_j h_j^{<t-1>} + (1 - z_j)\tilde{h}_j^{<t>}$$

where $\tilde{h}$ is calculated as

$$\tilde{h}_j^{<t>} = \phi([\mathrm{W}x]_j + [\mathrm{U}(r \odot h^{<t-1>})]_j)$$

Therefore with the interaction of these two gate the cell is able to learn a pattern whether to reset the hidden state using current input, or to retain the previous hidden state largely.

2.3 Textual Entailment Model

We try to first adapt an existing textual entailment model to this machine comprehension problem. The model is first proposed by Rocktaschel et al. (2015)., in which the premise is first contextually encoded, then a hypothesis-to-premise word by word attention is calculated. The model implicitly modify a hidden variable r_t to regulate the attention distribution at timestep t as

$$\begin{aligned} M_t &= \tanh(\mathrm{W}^y \mathrm{Y} + (\mathrm{W}^h h_t + \mathrm{W}^r r_{t-1}) \otimes e_L) \\ \alpha_t &= \mathrm{softmax}(w^T \mathrm{M}_t) \\ r_t &= \mathrm{Y}\alpha_t^T + \tanh(\mathrm{W}^t r_{t-1}) \end{aligned}$$

Here we have maintained the original symbol usage of Rocktaschel et al., where $\mathrm{W}_{(\cdot)}$ and $\mathrm{U}_{(\cdot)}$ are variable matrices to be trained, and Y is the matrix containing all hidden states of the premise

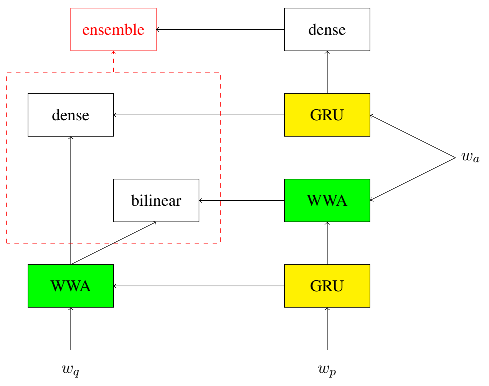

Figure 2: Our basic model, where "dense" represents fully connected layers, and WWA represents word-wise-attention structure described in section 2.3.

encoder. We generally follow the framework of this model, with modifications made to embody common sense knowledge and answer multiple choice questions.

3 Method

3.1 Bilinear Form

One of our classifiers uses the bilinear form. According to Milnor et al. (1973), a bilinear form is a function:

$$\beta{:}X \times X \rightarrow R$$

such that $\beta(x, y)$ is R-linear as a function of x for fixed y, and is R-linear as a function of y for fixed x. Then such a bilinear form can be called as an inner product on X. Thus this kind of inner product can intuitively be used to measure similarity between two representations of the same space, we design our bilinear classifier as follow:
Let α_1^N and α_2^N denote answer representations generated by our word-wise-attention mechanism from answers to text described in last section, and let q^N denote the question representation generated through the same process. Then we construct a trainable matrix $B^{N\times N}$. Then we can get a similarity score vector s where:

$$s_i = \alpha_i^T Bq$$

Then to normalize probability representation we perform a softmax function upon s to give our classification result.

3.2 Dense Classifier

We have another fully-connected classifier that works as described below. Let α_1^N and α_2^N denote the answer representations generated with pure RNN layer with GRU cell (compare with the WWA generated representation in the bilinear classifier), and let q^N denote the attended question representation generated by word-wise-attention mechanism described in subsection 2.3. Then we concatenate these three representation as c:

$$c = [q \colon \alpha_1 \colon \alpha_2]$$

Then we use c as the input of a two layer fully-connected neural network, where the hidden layer in the middle has nodes only half the number of the input, and the output layer is a softmaxed probability distribution representing our model's final choice.

3.3 General Model

Our model answers the multiple choice question by first encoding the question and the passage combined using the aforementioned textual entailment encoding, and then using two different question answering classifier to choose one of the two choices. A weighted sum is then calculated from the two answers each represented by a binary distribution. The weight is dynamically decided by a feed-forward network taking two contextually encoded answer strings as input. First the contextual embedded answer string representations of two answer choices are calculated by RNN-encoder, then we concatenate them and put them into a feed-forward network to calculate weights for the ensemble of the two aforementioned classifier. A general illustration can be seen in figure 2. Here GRU stands for an rnn unit using GRU cell, while WWA stands for word-wise attention used in textual entailment. We can see that our model can be trained end-to-end, and most of the weights can be dynamically learnt during training.

During training, the text is tokenized and lemmatized using python NLTK (Bird and Loper, 2002), and word stemming is not performed. We have made this choice because which words should be classified as stop word is hard to decide for an RNN model that are likely to capture some of the syntactic features of a given language.

The motivation for our using two different classifier is that we want to softly provide different solutions to different kinds of problems. The bilinear classifier measures similarity between question-attended and answer-attended contextual representations, which we believe should have better result on non-TF questions (By non-TF questions we mean those open questions which can not be answered by "Yes—True" or "No—False"), while the dense classifier should do better on TF questions according to our expectation. A detailed analysis of the model weights can be seen in next section.

3.4 Common Sense Knowledge

However some of the question in the test set cannot be answered merely with information provided in the passage. We try to embody some kinds of common sense knowledge representation into the general model, however their influence to system performance varies.

3.4.1 Word Embedding

In the general model, the embedding layer that converts the one-hot representation of an input word to its corresponding embedding vector is trainable, and is optimized during training using back-propagation(Rumelhart et al., 1986) algorithm. However, due to the relatively small database size, we have finally marked the embedding layer untrainable, and use GloVe (Pennington et al., 2014) word embeddings instead.

3.4.2 Script Knowledge

The script knowledge we have chosen is the OMCS (Singh et al., 2002) database. In the database are presented many different step by step descriptions of some daily events. To use this script knowledge, we first encode every single description as a passage using the RNN that was once used to encode our mission text. Then for every event e_i we calculate the event representation as an average of each description representation in that event category:

$$e_i = \frac{1}{M}\sum_{j=1}^{M} x_{i,j}^N$$

Where $x_{i,j}$ is the j^{th} description representation of event i. Then let E denote the event representation matrix with its i^{th} row representing event i, then every time context representation r_t is calculated, we calculate a similarity vector s as:

$$s = \text{softmax}(EWr_t)$$

Then if $\max_{i<M}(s_i) > t$ where t is a threshold hyper-parameter, we substitute r_t and its time series matrix with corresponding e_i and its time series matrix. But as this treatment provided little enhancement to our accuracy, we excluded

System	Accuracy
1^{st} place	0.843
SCReader	0.631
baseline	About 0.53

Figure 3: Table comparing performance of different systems.

this structure from our final submission. Still we believe that other common sense representation might be helpful, like ConceptNet (Liu and Singh, 2004).

4 Experiment Setting and Evaluation

4.1 Overview

We Trained our model using SGD with weight decay. No minibatch grouping is used, and we trained our model on training set for 20000 time steps. When near convergence, our model can reach around 80% to 90% accuracy upon training set (The accuracy is sampled), and in last two model we trained that finally lead to our only submission, we get an accuracy result of about 68% on developing set. This accuracy is a little higher than our final accuracy on test set. Our final result compared with baseline and the first rank system is given in the form.

4.2 Preprocessing

Before inputting the raw text into out model, we first transform words into their one-hot representation without stemming and lemmatization, and tokenization is done using NLTK toolkit. Then we push the data through an embedding layer in which the GloVe 50 was used due to time concerns.

4.3 Further Discussion

Providing our scarce usage of common sense knowledge, our model performed surprisingly well on time deduction problems. Even in questions where an addition of fifteen minutes to thirty minutes to get forty-five minutes as answer is required, our model successfully chose the right answer. However, the ensemble didn't work as we intended. The bilinear classifier was unalterably providing $[0, 1]$ after softmaxing, probably due to machine floating point precision limit. And the result weight determiner always assign the GRU related classifier a far greater weight. This is very counterintuitive, and we are still trying hard to find its real cause. Pre-trained word embedding boosted our accuracy for about 5% - 10% on developing set, and the training became faster and converged far more quickly. To further analyse the prediction ability of our two classification, we trained our two classifer separately by 20,000 steps, and calculated a sampled loss function according to 100 samples every 100 steps. The result is given in figure 4. We found that the bilinear classifier converges very slowly, while the dense classifier, if used separately, has a converging tendency even in 20,000 timesteps. This result corresponds with our observation that when used together, the weights assigned to these two classifiers always converge to $[0, 1]$, where little importance is given to the bilinear classifier, which is counter-intuitive.

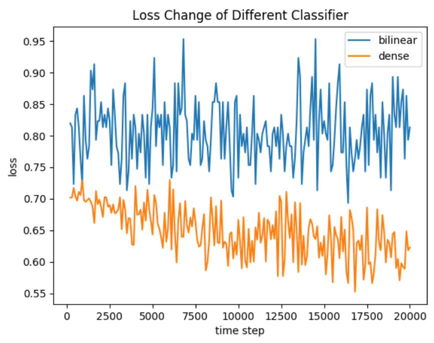

Figure 4: The loss change of our two classifier.

5 Conclusions and Future Work

Our model is a machine comprehension model based on textual entailment logics, and on the basis of previous works we made several renovations to embody common sense knowledge representation. We finally reached accuracy for about 63% on test dataset, however due to time limit, we have never tried any fine-tuning techniques. Observing this model we are able to say that it is useful to have common sense knowledge data integrated to machine comprehension problems, though a porper knowledge representation should be worked out. We are currently switching to other kinds of common sense knowledge representations, and trying to devise new answer selection logics. From the competition result it is very clear that there's still much space for our accuracy improvements.

References

KM ARIVUCHELVAN and K LAKAHMI. 2017. Reading comprehension system–a review. *Indian J. Sci. Res*, 14(1):83–90.

Steven Bird and Edward Loper. 2002. Nltk: The natural language toolkit. *CoRR*, cs.CL/0205028.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long short-term memory. 9:1735–80.

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

Hugo Liu and Push Singh. 2004. Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

John Willard Milnor and Dale Husemoller. 1973. *Symmetric bilinear forms*, volume 60. Springer.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. Semeval2018 task 11: Machine comprehension using commonsense knowledge. In *Proceedings of International Workshop on Semantic Evaluation(SemEval2018)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. 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*, pages 193–203.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer.