Dynamic Recurrent Attention Network for Question Answering

Team SWY[2]
Saeed Karimi Bidhendi
Wentao Zhu
Yang Feng
Yeeleng Scott Vang

Abstract

Several algorithms have been proposed for question answering, however, they are often the result of clever engineering design moreso than reflect the way human approach the task. To address this problem, we propose a dynamic recurrent attention network (DRAN) inspired by how humans alternatively read the context paragraph and question a number of times before arriving at an answer. The DRAN network fuses paragraph information with question information and vice versa a number of times before feeding into two softmax layers predicting the start and end span of the correct answer. On the Stanford question answering dataset, our DRAN models outperforms baseline model on the training set but is outperformed on the dev set.

1 Introduction

Question answering (QA) is a computer science application within the field of natural language processing (NLP) concerned with building intelligent systems that automatically answers questions posed by humans. QA systems can derive answers from unstructured collection of documents. Most QA systems incorporate text documents as their knowledge source. NLP techniques are then used to process the question and text corpora from which answers are extracted. Many QA systems use the World Wide Web as their text corpora. As an alternative, human users can assemble knowledge in a structured database, called a knowledge base which can serve as the data set for the QA system. It is also feasible to employ a hybrid QA system that uses both structured database and unstructured text documents.

QA is an import task in both machine learning and natural language processing areas. Many standard NLP tasks, such as machine translation, named entity recognition, part-of-speech tagging, sentiment analysis, co-reference resolution, can be reformulated as a QA problem (Kumar et al., 2015). Due to its complexity, question answering is considered a reading comprehension task as it requires understanding language and making inference over knowledge. Previous work use the combinations of recurrent neural networks, such as LSTM or GRU, and attention schemes in their system. Kumar et al. proposed dynamic memory network with attention scheme to deal with the task (Kumar et al., 2015). Seo et al. further employed bidirectional scheme into the attention networks (Seo et al., 2016). Xiong et al. implemented a co-attention scheme to effectively fuse the context and question information (Xiong et al., 2016).

Previous question answer data sets are quite small in size (Berant et al., 2014; Richardson et al., 2013). Stanford Question Answering Dataset (SQuAD) is a new 100,000+ question-answer pair reading comprehension dataset consisting of questions posed by crowd workers on a set of 500+ Wikipedia articles where the answer to every question is a segment of text, or span, from the corresponding reading passage (Rajpurkar et al., 2016). Different from current QA methods, we propose a dynamic recurrent attention network inspired by how humans alternatively read the context and question a number of times before arriving at an answer, and apply our approach to the SQuAD data set.

The organization of the report is as follows. We investigate the related work in Section 2. Then we describe the implementation details of the baseline method, and our attention recurrent neural networks in Section 3. Experimental Results, dataset description, and evaluation metrics are covered in

Section 4. Finally, conclusion and future work are discussed in Section 5.

2 Related Work

Since the release of the dataset in early 2016, much improvement has been made towards reaching human-level performance for this task. The following is a summary of published approaches reflected on the leaderboard. Along with the release of the dataset by Stanford University, Rajpurkar et al. (Rajpurkar et al., 2016) established a baseline model for this task using logistic regression and 180 million features consisting of lexicalized features or dependency tree path features. Wang and Jiang (Wang and Jiang, 2016) used Match-LSTM model to predict textual entailment between passage and possible and then Pointer Network (Ptr-Net) model to generate the answer output sequence from the passage. Yu et al. (Yu et al., 2016) proposed dynamic chunk reader (DCR) which consists of bi-directional GRU units to encode passage and question, followed by an attention layer to calculate relevance of each passage word to the question, and then chuck representation layer extracts candidate answer which are ranked by a softmax classifier. Yang et al. (Yang et al., 2016) introduced and combined fine-grained word-character gating model, which uses vector gates over scalar gates, with fine-grained documents-query gating model, which uses pairwise element-wise gating, to train an end-to-end system. Seo et al. (Seo et al., 2016) proposed BiDirectional Attention Flow model which is a hierarchical pipeline that incorporates character-level, word-level, and phrase-level embedding as well as bi-directional attention flow to encode the passage and question before passing through stacked bi-directional LSTM and softmax classifier to predict starting and end point of the answer. Wang et al. (Wang et al., 2016) proposed Multi-Perspective Context Matching (MPCM), a model that adjusts each word-embedding vector in the passage using a relevancy weight computed against the question followed by matching each context point with the encoded question to produce matching vector that will be fed to a bi-directional LSTM for answer prediction. Xiong et al. (Xiong et al., 2016) proposed Dynamic Coattention Network (DCN) which allows to iteratively alternate estimating the start and end of answer span to overcome any local maxima.

3 Model Details

3.1 Baseline - Sliding Window

From the various methods proposed to solve the question-answering problem of SQuAD data set, the sliding window approach and several other methods, e.g. logistic regression, constitute the baseline on which other more sophisticated methods are built. The sliding window baseline, which is what we have implemented in this section, is originally proposed for MCTest question-answering data set. The goal of the MCTest data set was to provide a Q&A setup such that all suggested solutions can be directly compared to each other. Therefore, this data set provides 4 possible answer choices for each question on the passage. Then, the sliding window baseline associate a score to each possible answer and chooses the one with highest score as the candidate answer.

This algorithm works as follows: Assume that we denote the i^{th} word of the passage P by P_i. If the set of words in the question and i^{th} possible answer is denoted via Q and A_i respectively, we can define the following functions to count the frequency and inverse frequency of each word in the passage:

- $C(w) = \sum_i \mathbb{I}\left(P_i = w\right)$
- $IC(w) = \log\left(1 + \frac{1}{C(w)}\right)$

The algorithm is given in Table 1.

In the SQuAD dataset, the answer to each question is a span of the passage. Each span is a collection of arbitrary length of consecutive words in the passage. Thus, for the passage P, the number of possible answers is given by:

$$\binom{|P|+1}{2} = \frac{|P| \times (|P|+1)}{2}$$

Therefore, the set of possible answers for SQuAD data set is much larger than MCTest data set. Thus, in order to make the search possible in a reasonable time, we assume that the answer to each question is completely contained in one sentence of the passage; so, we only need to consider all possible spans within each sentence of the passage. The answer to any question in the training data is given by both the true sequence of words and the number of starting character in the passage. Note that the sliding window approach does not need training and we evaluate its performance

Table 1: Sliding Window Baseline

Input:	Passage P, question Q and hypothesized answers A_i.
Output:	The correct answer A_s to the question Q.
1.	For each i
	$S = A_i \cup Q$
	$sw_i = \max_{j=1,\ldots,\lvert P\rvert} \sum_{w=1,\ldots,\lvert S\rvert} \left\{ \begin{array}{cc} IC(P_{j+w}) & \text{if } P_{j+w} \in S \\ 0 & \text{Otherwise} \end{array} \right\}$
	end for
2.	Return Arg $\max_i$ sw_i

on the training data itself. Due to large volume of the training set, we ran the program for different sections of the training and development data in parallel and reached the scores in Table 2.

3.2 Dynamic Recurrent Attention Network (DRAN)

Due to the number of members on the team, we decided to implement several variations of DRAN to suit our individual ideas/implementation style with respect to the underlying dynamic recurrent attention approach, denoted as DRAN Model 1 and DRAN Model 2.

3.2.1 DRAN Model 1

Three networks are investigated using DRAN Model 1. For word embeddings of questions and contexts, we used pre-trained 100-dimensional GloVe word embedding (Pennington et al., 2014), and fixed in the training.

- Three layers Bidirectional GRU network. We used 64, 96, and 128 units for the three layers GRU recurrent network, and bidirectional RNN is used for questions and contexts respectively. Then we concatenated the last states of questions and contexts to do pointer prediction.

- One layer Attention network. We used 128 and 160 units for two layers Bidirectional GRU network for questions, and 128 units for Bidirectional GRU network for context. The last state of the question GRU network was combined with a 320 units attention recurrent network for context. The last state of the question recurrent network was concatenated with the last state of the attention network. Here we used the same attention scheme as (Bahdanau et al., 2014). Please see the reference for the detail.

- Three layers Recurrent Attention network. We used 128 units for one layer Bidirectional GRU network for questions and contexts. The last states of the context GRU network was used to construct the attention layer of the context in light of the processed question. Similarly, we constructed an attention layer of the question in light of the processed context. These two attention layers were repeat for two times and then we merged the last state of question layer with each time state of context layer. Codes are available*.

After the above feature learning phases, we used time distributed prediction for the start pointer. Then we predicted the end pointer conditioned by the predicted start pointer. The three networks can be learned in an end-to-end manner. RMSProp was used for training with learning rate as 1×10^{-4} (Tieleman and Hinton, 2012). We used the early stopping criteria based on predicted accuracy. The training set was split into 80% training and 20% validation. The number of epochs is set as 5 and tolerance is 0.01.

Figure 1 through Figure 3 shows the learning curves for the previously described networks.

From the three figures, we can see our recurrent attention network achieved the smallest loss on the validation set, and was less likely to over-fitting. The training curve and the validation curve is very consistent. It shows the good generalization ability of our recurrent attention network.

3.3 DRAN Model 2

The DRAN Model 2 was inspired by the dynamic co-attention model of Xiong (Xiong et al., 2016). See Figure 4 for detail description of the architectures. The DRAN Model 2 simple attention

*https://github.com/wentaozhu/recurrent-attention-for-QA-SQUAD-based-on-keras.git

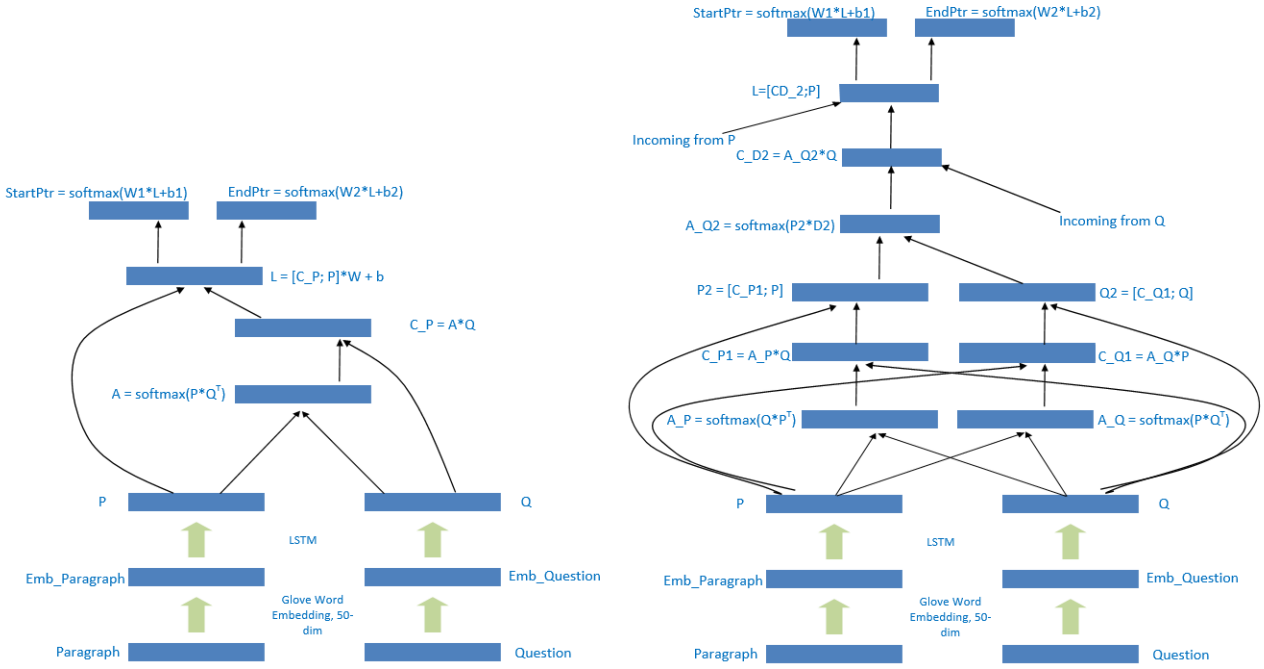

Figure 4: DRAN Model 2 simple attention (on left) and advanced attention (on right) networks

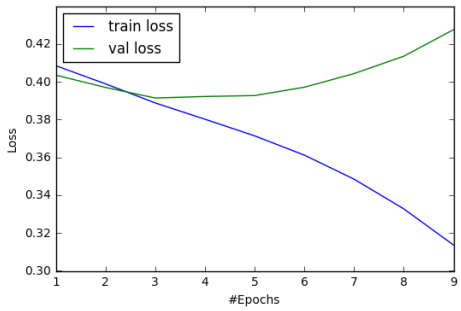

Figure 1: The learning curve of three layers bidirection GRU network.

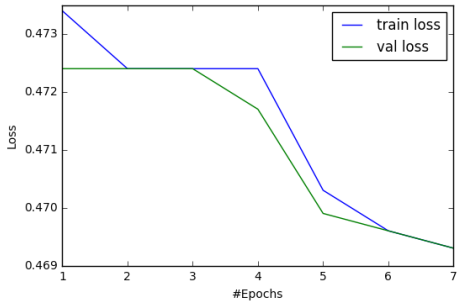

Figure 2: The learning curve of one attention network.

network utilized an affinity-like attention matrix to establish a baseline performance for this type of network. DRAN Model 2 advanced network expands on the simple attention model by implementing the proposed dynamic recurrent attention mechanism. Once context and paragraph information had been fused, the starting and end pointer of the answer span was predicted by two softmax layer independently of each other.

Much like DRAN Model 1, we used pre-trained 50-dimensional GloVe word embedding to embed the context paragraphs and questions. All unidirectional LSTM and dense layers used 100 units for computational time considerations. Adam optimizer was used with a learning rate of 0.004. Based on Figure 1 through Figure 3, 4 epoch was used with dropout included before the output softmax layers with a 50% dropout rate to help control for overfitting.

4 Experiments

4.1 Dataset Description

The dataset used for this task is the SQuAD dataset. SQuAD is a new reading comprehension dataset. It consists of questions posed by crowd workers on a set of Wikipedia articles where the answer to every question is a segment, or span, of text from the corresponding context paragraphs.

Table 2: Empirical Results

	Exact Match {Train — Dev }	F1 Score {Train — Dev }
Sliding window baseline (our implementation)	12.3% — 12.7%	18.4% — 19.5%
Sliding window baseline (original paper)	——— — 13.2%	——— — 20.2%
DRAN Model 1 (3 layer recurrent attention)	——— — 2.4%	——— — 8.2%
DRAN Model 2 Simple	60.8 — 2.6%	55.6 — 7.3%
DRAN Model 2 Advanced	63.1 — 4.2%	56.3 — 9.2%

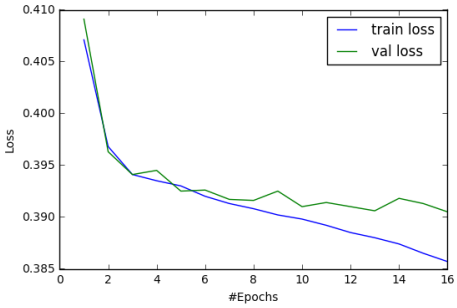

Figure 3: The learning curve of three layers recurrent attention network.

With 100,000+ question-answer pairs on 500+ articles, SQuAD is significantly larger than previous reading comprehension datasets.

4.2 Evaluation Criteria

We evaluate our approaches based on two metrics: Exact Match (EM) and F1 scores. EM is used to assess whether the generated answer is exactly correct, i.e. a generated answer is considered correct if it is identical to one of the three provided ground truth answers for each question. For example, if the ground truth is Seattle, both seattle and Seattle are correct, but in seattle or at seattle is not. The F1 score is commonly defined as the weighted harmonic mean of Precision and Recall. For this particular task set by the organizers of the dataset, precision and recall are treated equally.

4.3 Results

Empirical results for the sliding window baseline model and both DRAN Models are shown in Table 2. As can be seen, the baseline sliding window model outperforms our DRAN models on the dev set. Our DRAN models offer superior performance on the training set. What is interesting is that addition of the dynamic recurrent attention network over the simple attention network does improve performances by a few percentage points.

What we noticed is that our decoders, whether they predict the start and end points independently or they predict the end point conditioned on start point, we see around 34% where the end point is actually predicted to start before the start point. The other major finding producing the lower scores for our DRAN models is that there is a significant number of questions where prediction of the answer start point was outside of the actual length of its corresponding context paragraph length. In fact, we hypothesize there may be some co-adaptation going on where the same context paragraph is used but with different questions that leads to identical wrong prediction. Although adding dropout before the predictors with high dropout percentage did help improve performance, there may still be some components closer towards the start of the network that is causing this system to overfit and not generalize well.

5 Conclusion and Future Work

We proposed the Dyanmic Recurrent Attention Network, an end-to-end neural network architecture for question and answering. DRAN consists of stacking multiple layers of attentions, first with context in light of question and then question in light of context, followed by simple softmax predictors for the start and end point of the answer span, either independently or conditioned on one another. On the SQuAD dataset, the DCN achieves results wors than the baseline sliding window approach.

For future work, we believe while the encoding portion of the network is important, the decoding portion could be just as important. We proposed implementing a more sophisticated decoder such as those using highway maxout network or even fitting a sequence model over the hidden states that feeds into the softmax predictors to ensure that end point follows the starting point of the answer span.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR, abs/1506.07285* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, 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 *EMNLP*. volume 3, page 4.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2).

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Chohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724* .

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* .