Sequential Attention

Anonymous ACL submission

Abstract

In this paper we propose a neural network model with a novel *Sequential Attention* layer that extends soft attention by assigning weights to words in an input sequence in a way that takes into account not just how well that word matches a query, but how well surrounding words match. We evaluate this approach on the task of reading comprehension (*Who did What* and *CNN*) and show that it dramatically improves a strong baseline like the *Stanford Reader*. The resulting model is competitive with the state of the art model.

1 Introduction

Soft attention (Bahdanau et al., 2015), a differentiable method for selecting the inputs for a component of a model from a set of possibilities, has been crucial to the success of artificial neural network models for natural language understanding tasks like reading comprehension that take short passages as inputs. However, standard approaches to attention in NLP select words with only very indirect consideration of their context, limiting their effectiveness. This paper presents a method to address this by adding explicit context sensitivity into the soft attention scoring function.

We demonstrate the effectiveness of this approach on the task of *cloze*-style reading comprehension. A problem in the cloze style consists of a passage p, a question q and an answer a drawn from among the entities mentioned in the passage. In particular, we use the *CNN* dataset (Hermann et al., 2015), which introduced the task into widespread use in evaluating neural networks for language understanding, and the newer and more carefully quality-controlled *Who did What* dataset (Onishi et al., 2016).

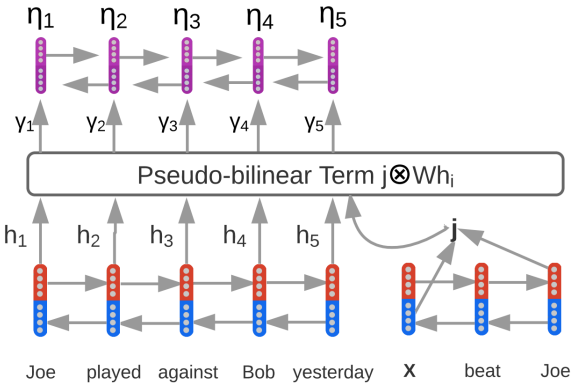

Figure 1: The Sequential Attention Model

In standard approaches to soft attention over passages, a *scoring function* is first applied to every word in the source text to evaluate how closely that word matches a locally-fixed *query vector* (here, a function of the question). The resulting scores are then normalized and used as the weights in a weighted sum which produces an *output* or *context* vector summarizing the most salient words of the input, which is then used in a downstream model (here, to select an answer).

In this work we propose a novel scoring function for soft attention that we call *sequential attention*, shown in Figure 1. In a *Sequential Attention* model, a modified version of the bilinear attention scoring function is used to produce a score *vector* for each word in the source text. A newly added bidirectional RNN then consumes those score vectors and uses them to produce a context-aware scalar score for each word. We evaluate this scoring function within the context of the Stanford Reader (Chen et al., 2016), and show that it yields dramatic improvements in performance. On both datasets, it is outperformed only by the *Gated Attention Reader* (Dhingra et al., 2016), which in some cases has access to features not explicitly seen by our model.

2 Related Work

In addition to Chen et al. (2016)'s *Stanford Reader* model, there have been several other modeling approaches developed to address these reading comprehension tasks. *ReasoNet* from Shen et al. (2016) uses a multi-turn and reinforcement learning approach; Cui et al. (2016) proposed an *attention-over-attention* mechanism; Seo et al. (2016) introduced the *Bi-Directional Attention Flow* which consists of a multi-stage hierarchical process to represent context at different levels of granularity; Munkhdalai and Yu (2016) introduces a memory augmented neural network approach with a recursive hypothesis; Dhingra et al. (2016) recently proposed a *Gated-Attention Reader* which integrates a multi-hop structure with a novel attention mechanism, essentially building query specific representations of the tokens in the document to improve prediction.

Outside the task of reading comprehension there is a more substantial body of work on soft attention over text, largely focusing on the problem of attending over single sentences. Luong et al. (2015) study several issues in the design of soft attention models in the context of translation, and propose the use of a bilinear scoring function. In work largely concurrent to our own, Kim et al. (2017) explore the use of conditional random fields (CRFs) to impose a variety of constraints on attention distributions. This work achieves strong results on several sentence level tasks, but at the cost of additional complexity stemming from the need to do inference over the embedded CRF model.

3 Modeling

Given the tuple (passage, question, answer), our goal is to predict $\Pr(a|d,q)$ where a refers to answer, d to passage, and q to question. We define the words of each passage and question as $d = d_1,..,d_m$ and $q = q_1,...,q_l$, respectively, where exactly one q_i contains the token *@placeholder*, representing a blank that can be correctly filled in by the answer. With calibrated probabilities $\Pr(a|d,q)$, we take the $\operatorname{argmax}_a \Pr(a|d,q)$ where possible a are restricted to the subset of anonymized entity symbols present in d. In this section, we present two models for this distribution: the Stanford Reader and our extended *Sequential Attention* model.

3.1 Stanford Reader

Encoding Each word of the vocabulary, including anonymized entities, is mapped to a d-dimensional vector via embedding matrix $E \in \mathbb{R}^{d\times|V|}$. For simplicity, we denote the vectors of the passage and question as $\boldsymbol{d} = \boldsymbol{d_1},..,\boldsymbol{d_m}$ and $\boldsymbol{q} = \boldsymbol{q_1},...,\boldsymbol{q_l}$, respectively. The Stanford Reader (Chen et al., 2016) uses bidirectional GRUs (Cho et al., 2014) to encode the passage and questions. For the passage, the hidden state is defined

$$\boldsymbol{h_i} = \text{concat}(\overrightarrow{\boldsymbol{h_i}}, \overleftarrow{\boldsymbol{h_i}}) \tag{1}$$

Were contextual embeddings $\boldsymbol{d_i}$ of each word in the passage are encoded in both directions

$$\overleftarrow{\boldsymbol{h_i}} = \text{GRU}(\overleftarrow{\boldsymbol{h_{i+1}}}, \boldsymbol{d_i}); \overrightarrow{\boldsymbol{h_i}} = \text{GRU}(\overrightarrow{\boldsymbol{h_{i-1}}}, \boldsymbol{d_i}) \tag{2}$$

For the question, the last hidden representation of each direction is concatenated

$$\boldsymbol{j} = \text{concat}(\overrightarrow{\boldsymbol{j_l}}, \overleftarrow{\boldsymbol{j_1}}) \tag{3}$$

Attention and answer selection In this case, a bilinear attention layer (Luong et al., 2015) is used

$$\alpha_i = \text{softmax}_i(\boldsymbol{j} W_s \boldsymbol{h_i}) \tag{4}$$

$$\boldsymbol{o} = \sum \alpha_i \boldsymbol{h_i} \tag{5}$$

The prediction is

$$a = \underset{a\in p\cap entities}{\operatorname{argmax}} W_a^T \boldsymbol{o} \tag{6}$$

meaning the word with highest probabilities within the anonymized entities. Finally a softmax layer is added on top of $W_a^T \boldsymbol{o}$ with a negative log-likelihood objective for training.

3.2 Sequential Attention

Our model uses the idea of attention input-feeding (Luong et al., 2015) in a different way. Luong et al. (2015) feed their original attention vectors of the form[1]

$$\tilde{\boldsymbol{s}}_{\boldsymbol{i}} = \tanh(W_c[\boldsymbol{c_i}; \boldsymbol{h_i}]) \tag{7}$$

into the next RNN step by concatenating them with the previous hidden state. The goal is to make the model fully aware of the previous alignment choices. Seo et al. (2016) and Dhingra et al. (2016)

[1] Where h_i is the hidden representation and c_i is the context representation

use similar concepts with the *Bidirectional Attention Flow* and *Gated Attention Reader* models. We however take advantage of the bilinear attention and use it to build a *Sequential Attention* model with two layers and considerably lower architectural complexity and similar performance.

In attention, instead of getting a single value α_i for each word in the passage by using a bilinear term, we define the vectors $\boldsymbol{\gamma_i}$ with a pseudo-bilinear term[2]. Instead of doing the dot product as in the bilinear term, we conduct an element wise multiplication to end up with a vector instead of a scalar:

$$\boldsymbol{\gamma_i} = \boldsymbol{j} \otimes W_s \boldsymbol{h_i} \tag{8}$$

We then feed the γ_i vectors into a new bidirectional GRU layer to get the hidden attention $\boldsymbol{\eta_i}$ vector representation:

$$\overleftarrow{\boldsymbol{\eta_i}} = \text{GRU}(\overleftarrow{\boldsymbol{\eta_{i+1}}}, \boldsymbol{\gamma_i}); \overrightarrow{\boldsymbol{\eta_i}} = \text{GRU}(\overrightarrow{\boldsymbol{\eta_{i-1}}}, \boldsymbol{\gamma_i}) \tag{9}$$

We concatenated the directional vectors to be consistent with the structure of previous layers, but it is worth experimenting with other operations in future work.

$$\boldsymbol{\eta_i} = \text{concat}(\overrightarrow{\boldsymbol{\eta_i}}, \overleftarrow{\boldsymbol{\eta_i}}) \tag{10}$$

Finally, we compute the α weights as below (where $sum[]$ is the sum of the elements of a vector) and proceed as before.

$$\alpha_i = \text{softmax}_i(sum[\boldsymbol{\eta_i}]) \tag{11}$$

4 Experiments and Results

We evaluate our model on two tasks, *CNN* and *WDW*. For CNN, we used the anonymized version of the dataset released by Hermann et al. (2015), containing $380,298$ training, $3,924$ dev, and $3,198$ test examples. For *WDW* we used Onishi et al. (2016)'s data generation script to reproduce their *WDW* data, yielding $127,786$ training, $10,000$ dev, and $10,000$ test examples[3]. We used the strict version of *WDW* which is a more difficult task and is used for their leaderboard [4].

[2]Note that doing softmax over the sum of the terms of the γ_i vectors would lead to the same α_i of the Stanford Reader

[3]We found 340 examples in the strict training data, 545 examples in the relaxed training data, 20 examples in the test set, and 30 examples in the validation set that were not answerable because the anonymized answer entity did not exist in the passage. We removed these examples, reducing the size of the test set by 0.2%, to $9,980$ examples.

[4]https://tticnlp.github.io/who_did_what/leaderBoard.html

Model	*WDW* Strict	*CNN*
Attentive Reader	53%	63%
Stanford Reader	65.6%	73.4%
+ Sequential Attention	67.2%	77.1%
Gated Att. Reader	**71.2**%	**77.9**%

Table 1: Accuracy on *WDW* and CNN test sets

Training We implemented all our models in Theano (Theano Development Team, 2016) and Lasagne (Dieleman et al., 2015) and used Stanford Reader (Chen et al., 2016) open source implementation as a reference. We largely used the same hyperparameters as Chen et al. (2016) in the Stanford Reader: $|V| = 50K$, embedding size $d = 100$, 100 dimensional *GloVe* (Pennington et al., 2014) word embeddings for initialization, hidden size $h = 128$. Attention and output parameters were initialized from a $U \sim (-0.01, 0.01)$ while GRU weights were intialized from a $N \sim (0, 0.1)$. Learning was carried out with SGD with a learning rate of 0.1, batch size of 32, gradient clipping of norm 10 and dropout of 0.2 in all the vertical layers[5] (including the *Sequential Attention* layer). Also, all the anonymized entities were relabeled according to the order of occurrence, as in the *Stanford Reader*. We trained all models for 30 epochs.

4.1 Results

Who did What In our experiments, *Stanford Reader* got a 65.6% accuracy on the strict *WDW* dataset compared to the 64% that Onishi et al. (2016) they reported. *Sequential Attention Reader* got 67.21%, which achieves the second position in the leaderboard, only surpassed by the 71.2% from the *Gated Attention Reader* with *qe-comm* (Peng Li and Xu, 2016) features and fixed GloVe embeddings. However, GA reader without features and fixed embeddings performs significantly lower at 67%. It is important to mention that we did not use these kind of features or fixed embeddings in our experiments, so it is likely that our model performs even better with those in future work.

Another experiment we conducted was to add

[5]We also tried increasing the hidden size to 200, using 200 sized GloVe word representations and increasing the dropout rate to 0.3. Finally we increased the number of hidden encoding layers to two. None of these changes resulted in significant performance improvements in accordance to (Chen et al., 2016).

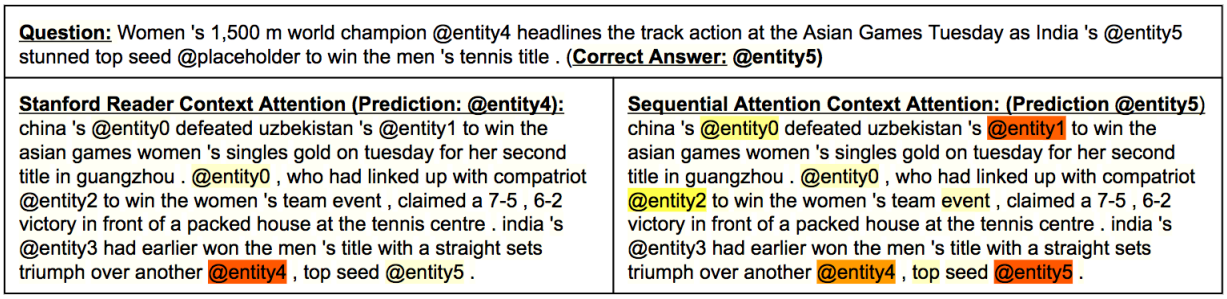

Figure 2: Representative sample output for the Stanford Reader and our model.

100K training samples from *CNN* to the *WDW* data. This increase in the training data size boosted accuracy by 1.4% with the SR and 1.8% with the *Sequential Attention* model reaching a 69% accuracy. This improvement strongly suggests that the gap in performance/difficulty between the CNN and the *WDW* datasets is partially related to the difference in the training sizes which generates overfitting. Furthermore, it suggests that the *CNN* data is relevant for the *WDW* task.

CNN For a final sanity check and a fair comparison against a well known benchmark, we ran our model with the *Sequential Attention* mechanism on exactly the same *CNN* data used by Chen et al. (2016).

Our *Sequential Attention* model took an average of 2X more time per epoch to train vs. the *Stanford Reader*. However, our model converged in only 17 epochs vs. 30 for the *SR*. The results of training the *SR* on *CNN* where slightly lower than the 73.6% reported by Chen et al. (2016). Our model achieved an accuracy of 77.1% which, to the best of our knowledge, has only been surpassed by Dhingra et al. (2016) with their *Gated-Attention Reader* model. The *Bidirectional Attention Flow* model is also very close in performance to ours and uses similar concepts, but its architecture is considerably more complicated.

4.2 Discussion

The difference between our *Sequential Attention* and the classical Bilinear Attention is that we are conserving the distributed representation of the Bilinear similarity at each component and propagating that contextual information to attention over other words. In other words, when the Bilinear attention layer is computing: $\alpha_i = \text{softmax}_i(\boldsymbol{j} W_s \boldsymbol{h_i})$, it only cares about the absolute value of the resulting α_i, meaning the amount of attention that it gives to that word. Whereas if we keep the vector $\boldsymbol{\gamma_i}$ we can also know which were the dimensions of the distributed representation of the attention that weighted in that decision. Furthermore, if we use that information to feed a new GRU, it helps the model to learn how to assign attention to surrounding words.

Figure 2 shows some sample model behavior. In this example and elsewhere, *Sequential Attention* results in less sparse attention vectors, and this helps the model assign attention not only to potential target strings (anonymized entities) but also to relevant contextual words that are related to those entities. This ultimately leads to richer semantic representations $\boldsymbol{o} = \sum \alpha_i \boldsymbol{h_i}$ of the passage.

5 Conclusion and Discussion

In this this paper we created a novel and simple model with a *Sequential Attention* mechanism that performs near the state-of-the-art on the *CNN* and *WDW* datasets by improving the bilinear attention mechanism with an additional bi-directional RNN layer. This additional layer allows the flow of attentional information of context to compute the attentional weight for each token.

For future work we would like to run our model on some of the recently launched datasets such as *SQuAD* and *Marco*. Additionally, we think that some of the ideas implemented in the *Sequential Attention* model could be mixed with ideas applied in recent research. We find the work of Dhingra et al. (2016) and Seo et al. (2016) particularly interesting. For example the use of *qe-comm* (Peng Li and Xu, 2016) features, fixed word embeddings and character-level embeddings have proved to boost performance of the GA Reader in *WDW*.

Our work focused on cloze-style machine comprehension, but we believe that the *Sequential Attention* mechanism may benefit other tasks as well, such as neural machine translation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

D. Chen, J. Bolton, and C. D. Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. *ArXiv e-prints* .

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* .

Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu. 2016. Attention-over-Attention Neural Networks for Reading Comprehension. *arXiv preprint arXiv:1607.04423* .

Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *CoRR* abs/1606.01549. http://arxiv.org/abs/1606.01549.

Sander Dieleman, Jan Schlter, Colin Raffel, Eben Olson, Sren Kaae Snderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gbor Takcs, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. 2015. Lasagne: First release. https://doi.org/10.5281/zenodo.27878.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 1693–1701. http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend.pdf.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Tsendsuren Munkhdalai and Hong Yu. 2016. Reasoning with memory augmented neural networks for language comprehension. *arXiv preprint arXiv:1610.06454* .

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2016. Who did What: A Large-Scale Person-Centered Cloze Dataset. *ArXiv e-prints* .

Wei Li Zhengyan He Xuguang Wang Ying Cao Jie Zhou Peng Li and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *CoRR* abs/1607.06275. http://arxiv.org/abs/1607.06275.

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. http://www.aclweb.org/anthology/D14-1162.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603. http://arxiv.org/abs/1611.01603.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.