R-Net with Multiplicative Attention

Pierce Freeman
Department of Computer Science
Stanford University
Stanford, CA 94305
`piercef@stanford.edu`

Rooz Mahdavian
Department of Computer Science
Stanford University
Stanford, CA 94305
`rooz@stanford.edu`

Abstract

The R-Net model, originally published by Microsoft, is one of the highest performing models measured against the SQUAD dataset. Various permutations of the original architecture (both single and ensemble) remain as top EM/F1 scores for prediction accuracy. Our model extends the original R-Net implementation with multiplicative attention instead of additive attention, which achieves comparable results in far less training time. We also enrich the current literature with a thorough analysis of our design choices.

1 Introduction

The SQUAD challenge is a well-defined supervised classification problem. It bundles a pair of 100,000 Wikipedia entries with questions asking about the original paragraph. When the set was constructed, crowd-sourcing was used to annotate the answers. These individuals were presented the passage, the question, and allowed to select a span of text to form an answer. Crucially, this span must come from the paragraph itself. A sample SQUAD entry is included below -

Traditional architecture is distinctive and include the Manueline, also known as Portuguese late Gothic, a sumptuous, composite Portuguese style of architectural ornamentation of the first decades of the 16th century.

Context

What is the Manueline style also known as?

Question

Portuguese late Gothic

Answer

Figure 1: Sample of a SQUAD pairing

By defining the problem in this way, SQUAD presents a proxy to test a machine's ability to understand unstructured content. It's a subset of more general reading compression, where a party can read a passage and respond to a question in freetext form. This sequence-to-sequence task remains one of the pinnacles of machine learning research, but due to its unstructured nature it's difficult to model adequately. The power of SQUAD rests in the answer span residing in the input passage. The task reduces to predicting the start and end tokens of the answer.

We extend the research done by Microsoft Research Asia in their *R-Net: Machine Reading Comprehension with Self Matching Networks*.[1] A core part of their implementation relied on the Gated Attention Unit, which adds additive attention with a localized gate to vanilla GRUs. This gate allows the model to selectively ignore input information at each time-step. Within our model, we replace the additive attention with a multiplicative attention. We also split the intertwined GRU into its own layer, which removes a recursive dependency. This yields similar performance while greatly reducing memory complexity and training time.

2 Background

2.1 Attention Methods

Within *Attention is All You Need*, Vaswani et al. propose a multiplicative attention that has better performance over previous methods.[2] They define the "attention" tensor as having keys and values K, V that can be different dimensions. The query Q interacts with these keys to create the attention distribution, and the actual weighted average is computed over V. Whereas additive attention doesn't allow for a direct interplay between the query and the attended tensor, directly applying a dot product can allow for a richer relationship to be learned. This creates a distribution for a given query. They explicitly define this relationship to be -

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

They also propose a permutation of this model, where they linearly weighted the query, keys, and values. This increased the predictive power of their sampled models -

$$\text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

2.2 Stable Dropout

Gal, et al. introduce an interpretation of RNN dropout with a Bayesian justification.[3] Traditionally, a deep RNN with multiple layers uses a different dropout mask for each layer and timestep. When applied in a RNN of depth $d = D$ to a sequence of length $t = T$, this results in DT separate dropout masks. For a given hidden unit h, we define this behavior as follows -

$$\text{Dropout}_{dth} = \text{Uniform}[0, 1]$$

Intuitively, this approach could negatively impact training and inference because the model is ignoring different information on each time step. Indeed, Gal, et al. similarly found that this dropout setup is unstable when compared to other techniques. Within their variational RNN, they use the same mask for every layer. The output of the first RNN is masked identically on every timestep, as is the second, and so on. The dropout mask thus becomes -

$$\text{Dropout}_{dh} = \text{Uniform}[0, 1] \tag{2}$$

3 Approach

3.1 Original R-Net Architecture

The original R-Net paper makes heavy use of attention modules with bidirectional RNNs. These RNNs have natural motivation because they can iterate over single-dimensional sequence data, like passages of text. Indeed, when RNNs forward propagate they roughly approximate rightward reading like English speakers do. R-Net makes use of bidirectional RNNs to read forward and backwards, which allows each timestep to have a better sense of surrounding context.

The original authors use additive attention, which we can generalize as a non-linearity applied to a group of vectors (where V includes the query and attention vectors).

$$\text{Attention}(V) = v^T \tanh(\sum_i^V W_i V_i) \tag{3}$$

If we have hidden size d_h and V vector i is of dimensionality d_{V_i}, then the weight matrices are of size: $W : [d_h, d_{V_i}]$.

We describe the forward propagation of this network as follows, following the notation of the original paper. We have also included diagrams of our own model, which should be similar to the original but may vary in specific details.

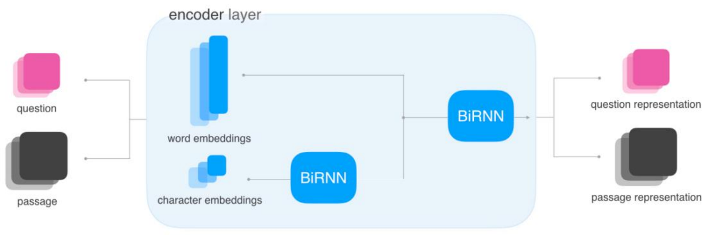


Figure 2: Encoder Layer

Define a question $Q = \{w_t^Q\}_{t=1}^m$ and passage $P = \{w_t^P\}_{t=1}^n$. For each word w_t, we have the associated Glove word-embedding e_t and character-level embedding c_t. For each question and passage in turn, we concatenate these word and character representations. The encoder layer runs these two values through a vanilla Bidirectional GRU. These hidden layers now incorporate some local context of neighboring words.

$$u_t^Q = \text{BiRNN}_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$
$$u_t^P = \text{BiRNN}_P(u_{t-1}^P, [e_t^P, c_t^P])$$

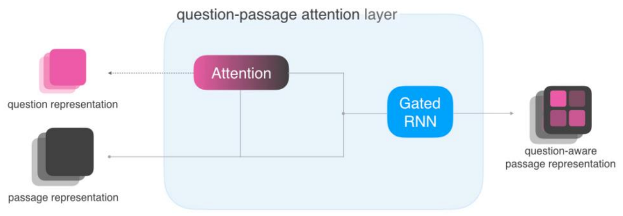

question-passage attention layer
Attention
Gated RNN
question representation
passage representation
question-aware passage representation

Figure 3: Question-Passage Attention Layer

Once we have the separate encoded representations, each word of the passage attends to the entire question to create an attention distribution. This tensor is then then piped through an RNN. We can motivate this choice by recognizing that most questions allude to a portion of the text, using similar word choice. If a passage word is also mentioned in the question, it may be more likely to surround the answer span. This 'sentence-pair' representation $\{v_t^P\}_{t=1}^n$ was first proposed by Rocktaschel et al. [4] R-Net uses this basic paradigm, with the additional improvements of Wang et al. [5] These two advances result in the following RNN -

$$v_t^P = \text{BiRNN}(v_{t-1}^P, *[u_t^P, c_t]) \tag{4}$$

$$s_j^t = \text{Attention}([u_j^Q, u_t^P, v_{t-1}^P]) \tag{5}$$

$$a_i^t = \frac{\exp(s_i^t)}{\sum_{j=1}^m \exp(s_j^t)}$$

$$c_t = \sum_{i=1}^m a_i^t u_i^Q$$

Here v^T is a $[1, d_{\text{hidden}}]$ vector, and the weights W are $[d_{\text{hidden}}, d_{\text{input}}]$. Additionally, the R-Net authors also define $*[u_t^P, c_t]$ to be a gated attention unit -

$$g_t = \text{sigmoid}(W_g[u_t^P, c_t])$$

$$*[u_t^P, c_t] = g_t * [u_t^P, c_t] \tag{6}$$

We recognize that this definition is recursive. It expects c_t to be passed to every timestep, but the c_t computation itself relies on the previous timestep's hidden state v_{t-1}^P. It's therefore impossible to calculate a complete c_t prior to running this BiRNN. Instead, it proved useful to instead write a replacement to the GRU that can calculate this attention distribute during its BiRNN propagation

A full vectorization of this new GRU requires careful attention to the shape of higher dimensional tensors. More specifically, at each timestep t we must accept a passage sequence u_t^P = [batch_size, embedding_size] and attend to the question embedding u^Q = [batch_size, question_length, embedding_size].

$$a = u_t^P (W_u^P)^T$$

$$b = ik, jmk \rightarrow jmi : W_u^Q, u^Q$$

Where the second quantity is expressed in Einstein summation notation. W_u^P and W_u^Q are of size [hidden_size, embedding_size]. The full attention tensor can then be computed as follows (making use of broadcasting along the third dimension) -

$$c = tanh(a + b)$$

$$s^t = ik, jmk \rightarrow jmi : v^T, c$$

Where v^T = [1, hidden_size]. This produces a final output tensor of s^t = [batch_size, hidden_size], which acts as the matrix of attention scores for each attended-to timestep.

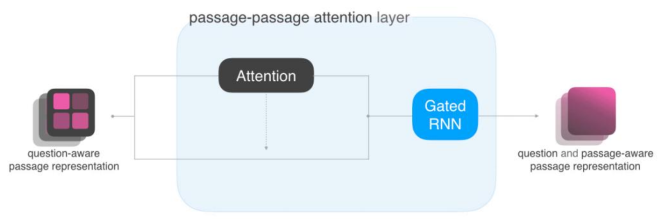

passage-passage attention layer
Attention
Gated RNN
question-aware passage representation
question and passage-aware passage representation

Figure 4: Passage-Passage Attention Layer

As a result of being fed through a RNN, the resulting passage has some ingrained notion of its context. However, due to vanishing gradients, this context is limited to around five words around each timestep t. Therefore we're motivated to create a contextually aware passage output. We facilitate this by having each word of the passage attend to the entire passage. We follow the same GRU definition from in equations 4-6.

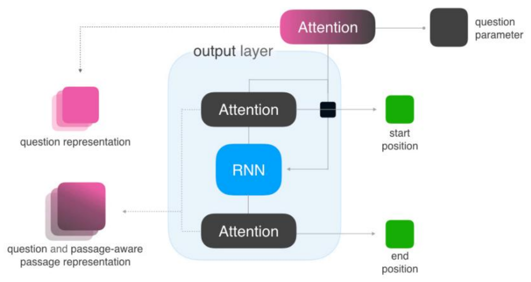


Figure 5: Pointer Network Layer

We define our pointer network generally as a layer that accepts a vector input, then outputs a softmax probability a over all the words of the passage. When run with different inputs, the argmax of this probability defines the 'start' and 'end' tokens of the span respectively p. This output layer is defined as an attention over some input $\hat{h}$ as such.

$$s_j = \text{Attention}([h_j^P, \hat{h}])$$

$$a_i = \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)}$$

$$c_i = \sum_{i=1}^{n} a_i^t h_i^P$$

$$p = \text{argmax}(a_1, ..., a_n)$$

Now we consider the values $\hat{h}$ that are inputted into this output layer. The "start" input is an attention distribution r^Q over the question. Heuristically, this serves to initialize the pointer network with a notion of the answer we're looking for.

$$s_j = \text{Attention}([u_j^Q, V_r^Q])$$

$$a_i = \frac{\exp(s_i)}{\sum_{j=1}^{m} \exp(s_j)}$$

$$r^Q = \sum_{i=1}^{m} a_i u_i^Q \tag{7}$$

Here, V_r^Q is a learned vector which is applied for every batch. The "end" input is generated by passing r^Q and its attended-to c through a single GRU cell -

$$\hat{h}^{\text{end}} = GRU(r^Q, c_{r^Q})$$

3.2 Character Embeddings

The method of generating character embeddings is not covered within the original R-Net paper. Since there's no industry standard method to generate character embeddings, we followed the methodology of Woolf et al. [6]

For every character i, we find all words in our corpus that contain i. Denote this set W. We then refer to the embeddings of each $w_j \in W$, such that we have embeddings e_j. With the intuition that characters have some semantic influence on word usage (eg. verbs or tenses) we define a character embedding as the average of word vectors. Thus $c_i = \frac{1}{\|W\|} \sum_i^W e_i$.

3.3 GRU Cell Bottleneck

During the unrolling of our custom GRU cells defined in equation 4, we have to calculate attention distributions during every timestep. This process can't be parallelized due to previous hidden state dependencies. It therefore poses a significant computational bottleneck during the training of our model.

Instead of performing this logic within the GRU, we instead propose a slight permutation of the model architecture. We perform the attention prior to passing the result to the BiRNN, so we modify equation 4 from $s_j^t = \text{Attention}([u_j^Q, u_t^P, v_{t-1}^P])$ into -

$$s_j^t = \text{Attention}([u_j^Q, u_t^P])$$

This decision does lose the ability to blend the passage context within the current attention store. However, we justify this choice on a few grounds. One, the post-attention RNN should provide some of this context awareness due to the GRU's internal weighting. Two, empirically we found this adjustment to only marginally affect performance while significantly decreasing training time.

3.4 Multiplicative Attention

We present an alternative to additive attention, using a non-linear multiplicative attention similar to that proposed by Vaswani et al. We take the dot product of two weighted vectors - V_1 and V_2 to substitute for our equation 3.

$$\text{Attention}(V) = \tanh(W_1 V_1 \cdot W_2 V_2)$$

This approach only supports up to two component vectors - usually one for the keys and one for the query. However, once we make the modeling choice described in section 3.3, all our attention layers fulfill this requirement.

4 Experiments

A majority of our initial experiments were focused on getting the R-Net model implementation as accurate as possible. In their paper, Microsoft publishes results with $71.1EM$ and $79F1$. Our best results were $61.82F1$ and $46.71EM$ on a dev set with a single ground truth label.

4.1 Loss Curves

Through tens of separate trials, some with sweeping architectural challenges, we found that our model faced a similar loss function and F1 curves. As such, it seems there may be an illusive upstream issue that we were unable to adequately discover at the time of submission.

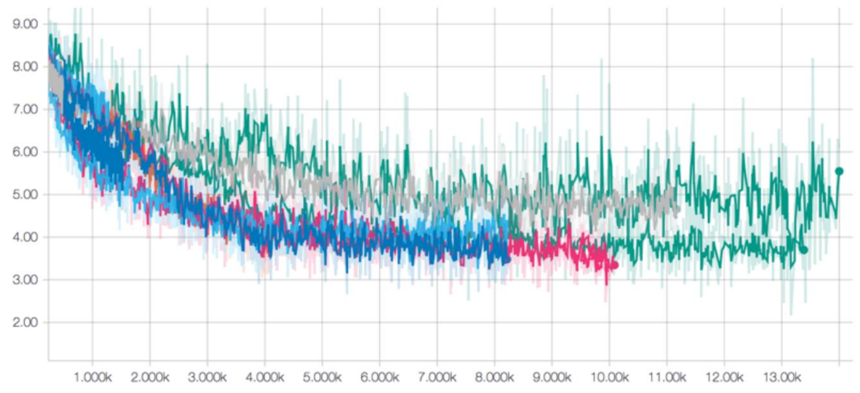
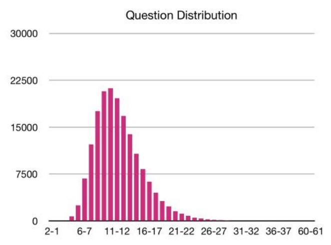


Figure 6: Training Loss Curves

4.2 Sequence Lengths

Our model pads questions and passages to a predefined length for their type. Larger sequence lengths require more GPU space to store in memory, so they limit our batch size and hidden layers. As such, it makes sense to prune this dataset if we are able to retain the majority of information.

We ran a manual analysis of our dataset to find where the text lengths appear to begin their long-tail. For questions, we found this was after character 20. For passages, we found this was after character 300. We discarded training examples longer than these values.



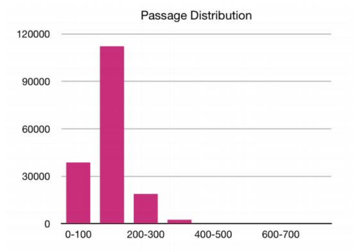


Figure 7: Length distributions in dataset

4.3 Trainable Character Embeddings

As part of diagnosing our model loss function, we disabled the training of character embeddings, relying on their default values. As described in section 3.2, these are averaged vectors over matching Glove words. Disabling training for this layer seemed to positively increase F1 and EM.

Table 1: Trainable Character Embeddings

	F1 @ iteration 2.5k	EM @ iteration 2.5k
Trainable	0.32	0.23
Non-Trainable	0.47	0.33

4.4 Multiplicative Attention

We note that the multiplicative layer achieves slightly worse performance compared to the vanilla additive layer. With that said, in some constrained cases it is worth the trade-off. First, it has a lower memory footprint. This allowed us to simultaneously increase the batch size and hidden layer size, while still operating on only one GPU. Second, matrix multiplications can be optimized much faster than additions. This can decrease training time by a significant margin.

Table 2: Multiplicative Attention

	F1 @ iteration 6.2k	EM @ iteration 6.2k
Additive	0.50	0.37
Multiplicative	0.47	0.34

4.5 CUDA RNNs

We also attempted to rewrite our RNN layer using the CudnnGRU, a highly optimized implementation of the GRU that works on Nvidia's GPUs.

The speed optimization that CUDA brings is nearly a factor of 4.5. That said, when combined with the multiplicative attention layer it appears to suffer in performance rather significantly.

Table 3: Cuda RNN Runtime

	Execution Time	Dev F1 @ 8.2k	Dev EM @ 8.2k
Vanilla GRU + Gated Attention	12h 4min	0.57	0.43
Cuda GRU + Attention Layer	2hr 41min	0.48	0.34

4.6 Attention Analysis

We have visualized the attention distributions for each of the layers in our model discussed in **Section 3.1** in Figure 8 below, for a successful example question/answer pair from the dev set.

Within each distribution, we highlight all words which have a probability within the highest relative order of magnitude (where the strength of the highlight reflects the relative probability mass within this order of magnitude), and annotate them with the corresponding probability.

For the passage-question layer and the passage-passage layer, the distributions are provided specifically for the two words in the ground-truth answer within the passage (over the question and passage, respectively); the model accurately predicts the ground-truth span for this example, which makes it likely the information provided in the attention layers for the two words in the answer span would be relevant. Note that that each distribution was computed with multiplicative attention (as discussed in **Section 4.2**).

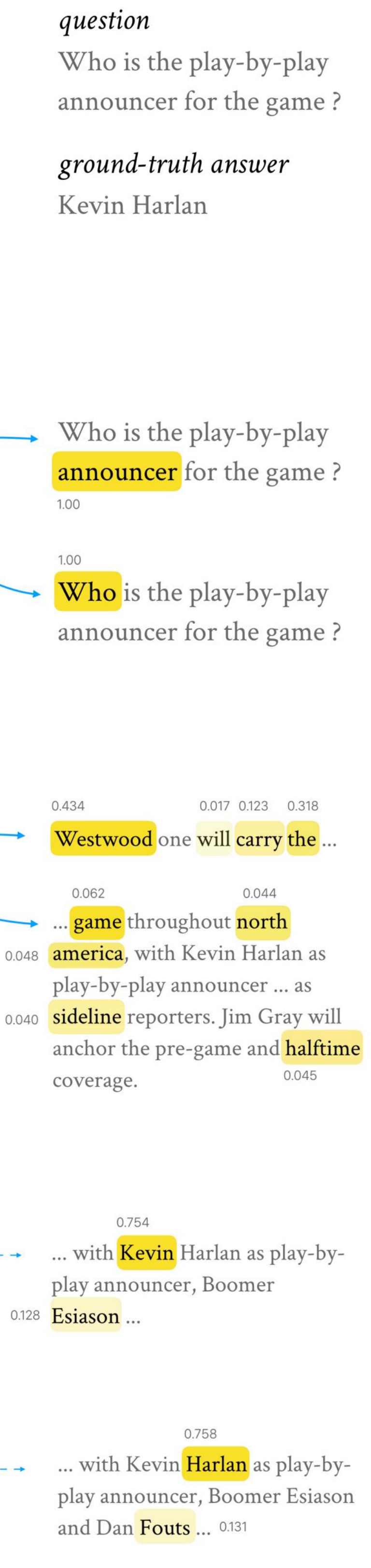
Interestingly, we see that, within the passage-question layer, the token ”Kevin” from the question matches directly with the token ”announcer” from the question, and the following token ”Harlan” from the question matches directly with the token ”Who” from the question. Together, we interpret this to enrich the passage representation at these time-steps with information about the nature of the question as one of identity (”Who” matching to ”Harlan”) and role (”Kevin” matching to ”announcer”). The conjunction of these two may have been crucial in the disambiguation between the multitude of names and roles within the passage, as in the pointer-output layer we can see that the start distribution almost selected ”Esiason” instead of ”Kevin”, and the end distribution almost selected ”Fouts” instead of ”Harlan”. The distributions within the passage-passage layer are less easily interpretable, but appear to integrate the context surrounding both tokens.

5 Conclusion

We have demonstrated that CUDA RNNs and independent attention gates show promise to significantly improve training times of R-Net. As part of this process, we also robustly analyzed the design choices and enumerated some reasonable assumptions that weren't specified in the original design.

We hope this analysis of R-Net and supplementary choices can enable further experimentation within this model. Despite the potential implementation or hyperparameter choice that is keeping our F1/EM scores lower than the original paper, we have shown R-Net to be relatively robust to differences in modeling choices. This proves promising for the overall resilience of the underling attention dynamics as they are applied to other domains.

4.6 Attention Analysis



passage
Westwood one will carry the game throughout north america, with Kevin Harlan as play-by-play announcer, Boomer Esiason and Dan Fouts as color analysts, and James Lofton and Mark Malone as sideline reporters. Jim Gray will anchor the pre-game and halftime coverage.
question
Who is the play-by-play announcer for the game ?
ground-truth answer
Kevin Harlan
passage-question attention
Who is the play-by-play announcer for the game ?
1.00
1.00
Who is the play-by-play announcer for the game ?
passage-passage attention
0.434 0.017 0.123 0.318
Westwood one will carry the ...
0.062 0.044
... game throughout north
0.048 america, with Kevin Harlan as play-by-play announcer ... as
0.040 sideline reporters. Jim Gray will anchor the pre-game and halftime
0.045
coverage.
start-pointer
0.754
... with Kevin Harlan as play-by-play announcer, Boomer
0.128 Esiason ...
end-pointer
0.758
... with Kevin Harlan as play-by-play announcer, Boomer Esiason and Dan Fouts ... 0.131

Figure 8: Attention Distributions on a Successful Prediction

6 References

[1] N. L. C. G. Microsoft Research Asia, "R-net: Machine reading comprehension with self-matching networks,"

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[3] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[4] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom, "Reasoning about entailment with neural attention," *arXiv preprint arXiv:1509.06664*, 2015.

[5] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 189–198.

[6] Woolf, *Pretrained character embeddings for deep learning and automatic text generation*. [Online]. Available: http://minimaxir.com/2017/04/char-embeddings/.