Machine Comprehension with Modified Bi-Directional Attention Flow Network

Sijun He
Stanford University
sijunhe@stanford.edu

Jiajun Sun
Stanford University
jiajuns@stanford.edu

Mingxiang Chen
Stanford University
ming1993@stanford.edu

Abstract

The advancement of of Machine Comprehension (MC) in recent years can be largely attributed to the attention mechanism. We proposed a filtering layer in the multi-layer hierarchical neural network architecture that can potentially enhance the performance of the attention mechanism. Our preliminary experimentation on the Stanford Question Answering Dataset (SQuAD) showed that the filtering layer can effectively assist the attention flow layer in Bi-Directional Attention Flow (BiDAF) model.

1 Introduction

Almost all human knowledge are recorded as text. The ability to read, understand and extract information from text is natural for human beings, but very challenging for machines. In recent years, the field of Machine Comprehension (MC) has made significant progress. Many end-to-end neural network systems have gained tremendous success on the task of Question Answering (QA), where the machines answer questions, using words contained within the given related context. Many of the success can be attributed to the invention of the attention mechanism, which allows the system to focus on specific area, may it be a sentence in a paragraph, or a selected region of pixels in an image. The leading models on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016) , such as Dynamic Coattention Networks (Xiong et al. 2017) and Bi-Directional Attention Flow (Seo et al. 2017), all involves innovation on their attention mechanisms in the architecture.

The objective of our project is to propose a filtering layer in the multi-layer hierarchical neural network architecture that can potentially enhance the performance of the attention mechanism. The filtering layer is inspired by the concept of similarity from collaborative filtering in recommendation systems and aims to amplify the hidden signal of relevant information while filtering out the redundant information. Our preliminary experimentation on the Stanford Question Answering Dataset (SQuAD) shows that the filtering layer can effectively assist the attention flow layer in Bi-Directional Attention Flow (BiDAF) model.

2 Models

As shown in Figure 1, our neural network architecture consists of six layers, of which three are identical to the BiDAF model and two are slightly modified. The remaining one is the filtering layer which we proposed.

1. **Word Embedding Layer** maps each word to a high-dimensional representation using a pre-trained GloVe (Pennington et al.) word embedding model.
2. **Filtering Layer** filters the redundant information and emphasizes on the most relevant context
3. **Contextual Embedding Layer** models the contextual representation of words in its surrounding neighborhood
4. **Attention Flow Layer** intertwine the contextual representation of words in the question and the context into a question-aware representation for each word in the context
5. **Modeling Layer** captures the interaction among the question-aware representation of words in the context
6. **Output Layer** outputs the answer

2.1 Word Embedding Layer

The word embedding layer maps each individual word into a high-dimension vector space. It is a set of pre-trained GloVe (Pennington et al.) word vectors using datasets provided by Wikipedia 2014 and Gigaword 5. The dimensionality of the vectors d is 100. The word embedding layer outputs the context representation $\mathbf{X} \in \mathcal{R}^{d \times T}$, where T is the maximum number of context length and $\mathbf{Q} \in \mathcal{R}^{d \times J}$, where J is the maximum number of question length.

2.2 Filtering Layer

The filtering layer is inspired by Wang et al. (2016) and the concept of similarity of collaborative filtering in Recommendation Systems. The idea is that only a small portion of context is needed to answer the question so we use the cosine similarity to dampen the signal of redundant information. For each word pair with context word $x_i \in \mathbf{X}$ and $q_j \in \mathbf{Q}$, we compute the cosine similarity

$$\text{sim}_{i,j} = \frac{x_i^T q_j}{|x_i||q_j|}$$

And the relevancy is computed as a weighted average of the cosine similarity, with W_{sim} as a trainable weights of $\mathcal{R}^J$.

$$r_i = W_{\text{sim}}^T \left[\text{sim}_{i,:}\right]$$

2.3 Contextual Embedding Layer

We use a Bi-directional Long Short-Term Memory (LSTM) (Hochreiter et al.) network as our encoding layer. Contexts and questions will be fed into the network respectively, while the output from each hidden state are all be recorded. Note that the output of which will be $2d$-dimension vectors (where d is 100 in our model) due to the concatenation of two output for both direction of the network.

$$\mathbf{H} = \left[\overrightarrow{\mathbf{H}}, \overleftarrow{\mathbf{H}}\right] = \text{BiLSTM}(\mathbf{X})$$

$$\mathbf{U} = \left[\overrightarrow{\mathbf{U}}, \overleftarrow{\mathbf{U}}\right] = \text{BiLSTM}(\mathbf{Q})$$

2.4 Attention Flow Layer

The attention layer is used to couple the information from the question and the information given by the context. It categories the importance of the attention to each word vector in the text (both contexts and questions). The input of this layer is contextual vector of the context $\mathbf{H}$ and the question $\mathbf{U}$.

In this layer, the attention is calculated from the similarity matrix of $\mathbf{H}$ and $\mathbf{U}$ as follow:

$$\mathbf{S}_{ij} = \alpha(\mathbf{H}_{:,i}, \mathbf{U}(:, j))$$

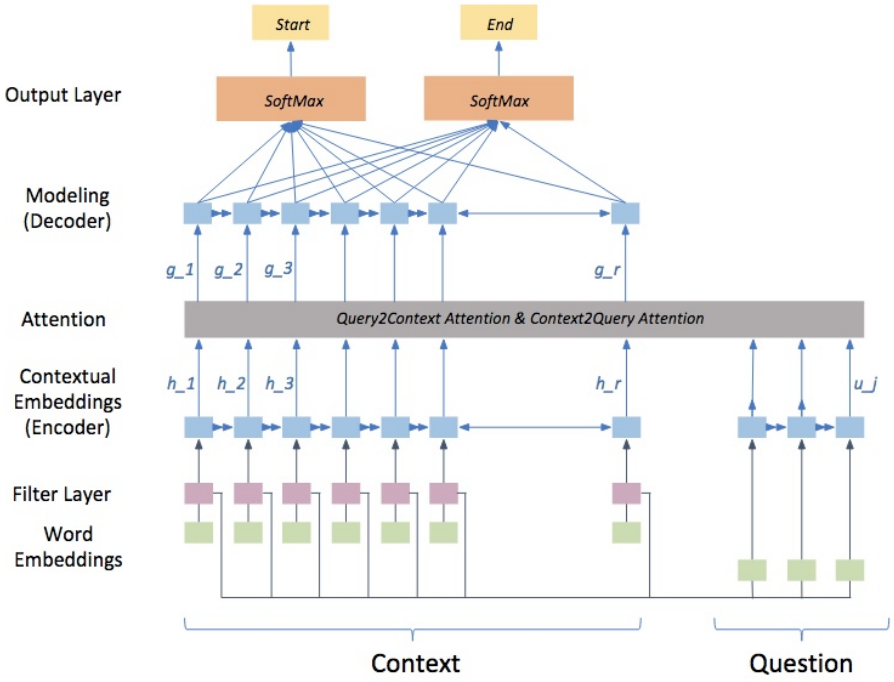


Figure 1: Bi-Directional Attention Flow Model

where $\mathbf{S}_{ij}$ indicates the similarity of the word i in the context and the word j in the query, and α is the bilinear factor introduced by Chen et al. (2016) which is defined as

$$\alpha(h, u) = h^T \mathbf{W}_\alpha u$$

$\mathbf{W}_\alpha$ is a trainable matrix of $\mathcal{R}^{2d \times 2d}$. The similarity matrix is different from the original BiDAF model where

$$\alpha(h, u) = \mathbf{W}_\alpha^T [h; u; h \circ u]$$

2.4.1 Context-to-query Attention

The context-to-query attention indicates the weight of each word in the query considering the given context. The attention weight is computed as follow

$$a_i = \text{softmax}(S_{i:})$$

The attended query vector for the entire context is computed as

$$\hat{\mathbf{U}} = \sum_j a_{ij} \mathbf{U}_{:j}$$

2.4.2 Query-to-context Attention

The query-to-context attention indicates the weight of each context word considering the given query. Similar to what we did in context-to-query attention, the attention weights are calculated from

$$b = \text{softmax}(max_{col}(\mathbf{S}))$$

and the attended context vector matrix can be computed as

$$\hat{\mathbf{H}} = \sum_t b_t \mathbf{U}_{:t}$$

Finally, the bi-direction attention and the contextual embeddings are combined to yield $\mathbf{G}$.

$$\mathbf{G}_{:t} = \beta(\mathbf{H}_{:t}, \hat{\mathbf{U}}_{:t}, \hat{\mathbf{H}_{:t}})$$

where the β function is the concatenation of the vectors and their element-wise products. The simple concatenation has shown good performance, as described by Seo et al.

$$\beta(h, \tilde{u}, \tilde{h}) = [h, \tilde{u}, h \circ \tilde{u}, h \circ \tilde{h}]$$

2.5 Modeling Layer

The input of the modeling layer (decoding layer) encrypts the information of the context together with some information from the query. So that when we pass the tensor to a bi-LSTM network, similar to what we did in the contextual embedding layer, the output would be a matrix of $\mathcal{R}^{2d \times T}$, where T is maximum number of context length.

$$\mathbf{M} = \left[\overrightarrow{\mathbf{M}}, \overleftarrow{\mathbf{M}}\right] = \text{BiLSTM}(\mathbf{G})$$

2.6 Output Layer

In this layer, we output the answer to questions by calculating the probability for each position in the context as the start and the end indices of the answer with softmax. However, in some situations, though very rare, we would flip over the start and ending labels if the ending predicted maybe even appears earlier than the start in the context.

$$a_s = \text{softmax}(\mathbf{W}_s^T [\mathbf{G}, \mathbf{M}])$$

$$a_e = \text{softmax}(\mathbf{W}_e^T [\mathbf{G}, \mathbf{M}])$$

The prediction of the end index in our model is simpler than the BiDAF model, where $\mathbf{M}$ is passed through another bi-directional LSTM to obtain $\mathbf{M}^2$. The BiDAF model predicts the end index of the answer as

$$a_e = \text{softmax}(\mathbf{W}_e^T \left[\mathbf{G}, \mathbf{M}^2\right])$$

We had the chance to briefly experiment with the extra bi-directional LSTM layer, but didn't see any immediate improvement over the our simpler model (section 4). It is likely that with adequate hyper-parameter tuning, the BiDAF model would outperform.

We define the loss criterion as the sum of the cross entropy error of the true start and end indices by the predicted start and end indices, averaged by the batch size.

$$L(\theta) = \frac{1}{N} \left[\text{CE}(a_s, y_s) + \text{CE}(a_e, y_e)\right]$$

3 Experiment

3.1 Dataset

Our model was evaluated using the recently released SQuAD (Rajpurkar et al.) dataset. The dataset is split into training, validation and test set, which respectively has roughly 81k, 4k and 10k samples. Each sample in the training set contains a context paragraph, question and one candidate answer, while the validation set and the test set have three candidate answers for each sample.

3.2 Evaluation

SQuAD provides two metrics to evaluate the performance of a model: ExactMatch (EM) and F1 score. Exact match measures the percentage of predictions that exactly match one of the ground truth answers, while F1 score is calculated as the harmonic mean of the token-level precision and recall of the predictions. The F1 score can also be loosely interpreted as the average overlap between the prediction and ground truth answer. When measuring the prediction against multiple ground truth answers in the validation set and the test set, the maximum F1 is taken over all of the ground truth answers for a given question, and then average over all of the questions.

3.3 Data Preprocessing

3.3.1 Maximum context and question length

The longest context paragraph and question have 2834 and 281 words respectively. The overwhelming majority of the contexts are shorter than 1000 words and the questions shorter than 100 words, as shown in Figure 2. Moreover, based on the training data, the longest answer ends at the 605-th word of the context. By limiting the maximum context length to 605 and the maximum question length to 100, we speed up the computation significantly while avoid losing any answers.

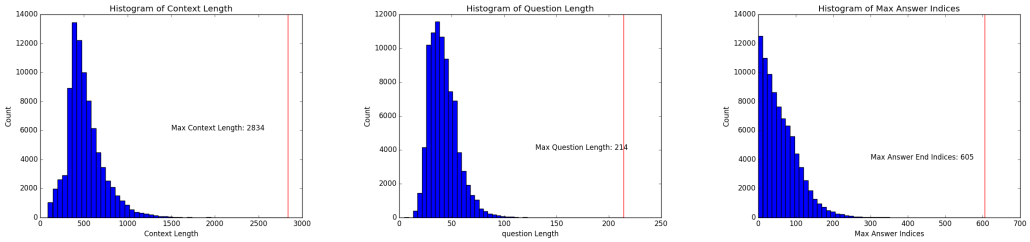


Figure 2: Distribution of Context, Question Length and Answer End Indices

3.3.2 Padding & Masking

The context paragraphs and questions shorter than the their maximum lengths is padded with the special token $\langle pad \rangle$. When computing the cost, the padded portion are masked with exponential masking before applying the softmax function.

3.3.3 Word Embedding and Vocabulary

Following the starter code provided by the teaching staff of CS224N, we constructed vocabulary list of all words in the training and validation set and trimmed the GloVe word vectors accordingly. This was proven to be a mistake, as we later observe inconsistency in our validation set and test set, possibly due to Out-Of-Vocabulary(OOV) error. Due to time constraints of the project, we were not able to fix the issue.

3.4 Model Training Details

The model is trained for 12 epochs with a batch size of 150 and a learning rate 0f 0.001 using the adaptive Adam optimizer (Kingma et al.). The learning rate was annealed with an exponential decays at 0.9 per epoch using the staircase exponential decay API provided by Tensorflow. A dropout rate of 0.2 is applied to all LSTM layer and the weighted output of the output layer. The training process averages roughly 30 minutes per epoch on a single Tesla M60.

4 Result

As shown in Table 1, our results provide some preliminary evidence that the filtering layer can effectively assist the attention layer in focusing on a specific area for the answer to the question. With the task of analyzing the effect of the filtering layer in mind, we were able to do a reasonable amount of hyper-parameter tuning and produce a 3-model average of metrics.

The table also includes various models we have attempted. We had reasonable belief that these models would work well, but have not had the chance to do enough hyper-parameter tuning. As shown in figure 3, model with glove 300 dimension has significantly lower training loss but its F1 score is pretty low. This is due to over-fitting. Due to various mistakes made at the earlier stage of the project, we do not have enough time to turn our model to reach its full capability and we were not able to run as many models as we wanted.

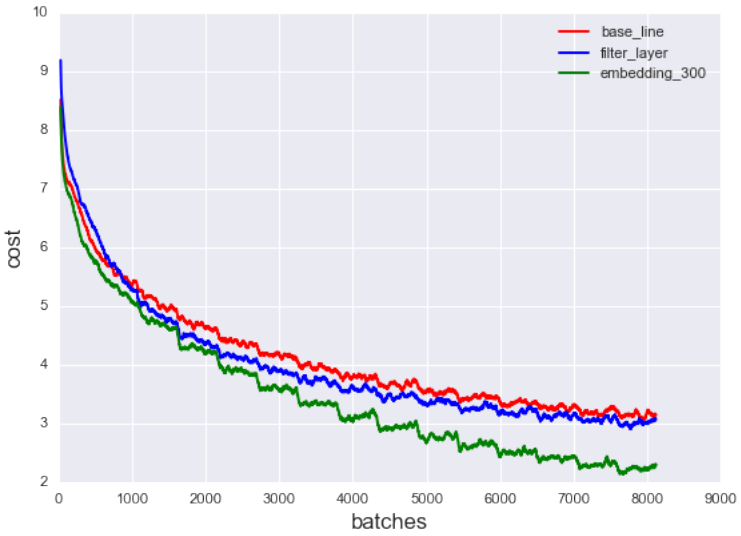


Figure 3: converge of three models

Model	Description	Val F1	Val EM	Test F1	Test EM
Baseline	BiDAF	57.2*	43.1*		
Filtered	BiDAF w/ filter	**61.8***	**47.2***	**56.5**	**42.4**
Attempted but Lacked Tuning Time					
150 Hidden	BiDAF w/ 150 hidden	46.2	29.3		
150 Hidden Filtered	BiDAF w/ 150 hidden & filter	59.2	44.3		
300d GloVe	BiDAF w/ 300d GloVe	52.1	38.6		
end LSTM	extra LSTM for answer end	51.5	35.2		

$*$ denote average of 3 models

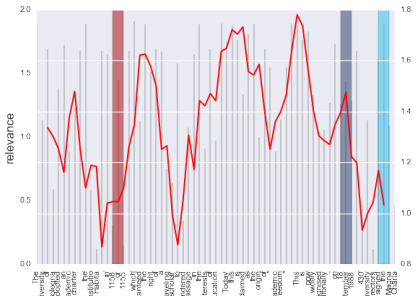
Table 1: Results Summary

4.1 Error Analysis

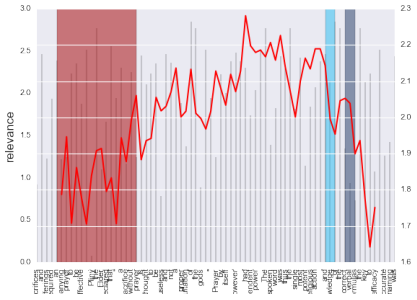
The filter layer shows some interesting results. Three examples are selected to demonstrate what is the mechanism behind the filter layer that helps improve model performance. As shown in figure 4, these are two examples that filter layer outperforms the baseline model. Figure on the left side demonstrates the norm of the filter layer at each word position. The bar graph in the background shows the absolute value and the red line is a windowed moving average. Three colored sections show the matched words.

It is found that the filter layer is highly likely to increase the signal for the corrected answer. In many cases, words around correct answers tend to be more similar to the context than other regions. As shown in figure 4, the light blue regions which are the signal from question are very close to the true answer. This helps the model to predict the correct answer. On the other hand, the right answer themselves are more similar to the context than the wrong one. This increase the probability for predicting correct answer.

Figure 5 demonstrates a case that filter layer is not helpful at all. In this case, the model with filter layer dose not produce correct answer. The question signal (marked in light blue) shows up twice in the context; therefore, filter layer gets confused.

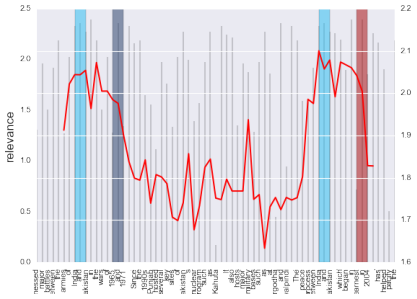


Question: On what date was the Magna Carta Universitatum signed?
Context: The University of Bologna adopted an academic charter , the Constitutio Habita , in 1158 or 1155, which guaranteed the right of a traveling scholar to unhindered passage in the interests of education. Today this is claimed as the origin of "academic freedom". This is now widely recognised internationally, on 18 September 1988, 430 university rectors signed the Magna Charta Universitatum



Question: What knowledge was of importance in the potency of prayer?
Context: All sacrifices and offerings required an accompanying prayer to be effective. Pliny the Elder declared that" a sacrifice without prayer is thought to be useless and not a proper consultation of the gods." Prayer by itself, however, had independent power. The spoken word was thus the single most potent religious action, and knowledge of the correct verbal formulas the key to efficacy

Figure 4: figure on the left shows the norm of filter layer at each position. red color labels the wrong prediction from baseline model and dark blue labels the correct prediction from model with filter layer and light blue label the signal from question



Question: When did India and Pakistan fight in Punjab?
Context: Punjab witnessed major battles between the armies of India and Pakistan in the wars of 1965 and 1971. Since the 1990s Punjab hosted several key sites of Pakistan's nuclear program such as Kahuta. It also hosts major military bases such as at Sargodha and Rawalpindi. The peace process between India and Pakistan, which began in earnest in 2004, has helped pacify the situation

Figure 5: figure on the left shows the norm of filter layer at each position. red color labels the wrong prediction from model with filter layer and dark blue labels the correct prediction from baseline model and light blue label the signal from question

4.2 Future Improvement

- Further and more structured hyper-parameter (grid search, etc.) tuning in order to reproduce the performance of the original BiDAF model
- Comparison of different variant of filtering layer: use max instead of weighted average,etc
- Analysis on the effect of attention weights, with and without the filtering layer

5 Conclusion

With this project, we proposed a filtering layer sitting on the word embedding layer that can potentially enhance the performance of the attention mechanism. Though not comprehensive, our preliminary experimentation on the SQuAD dataset showed that the filtering layer we proposed can effectively assist the attention flow layer in the BiDAF model. We also had the opportunity to closely examined and implemented the multi-stage hierarchical process of the BiDAF model, though we were

not able attempted to fully reproduce its performance. The hands-on experience on deep learning from this project has been both humbling and rewarding.

6 Contributions

Sijun He

- Responsible for the structuring of the code and turning the start code into a minimal working model
- Major contribution to the implementation of the model, with emphasis on preprocessing utility scripts, word embedding layer, contextual embedding layer, attention flow layer, answer formulation functions
- Running models on GPU

Jiajun Sun

- Lead the effort in expanding the baseline BiDAF model, including hyper-parameter exploration and devising model modifications
- Major contribution to the implementation of the model, with filtering layer, attention flow layer, decoder modeling layer, decoder output layer, model prediction scripts
- Running models on GPU

Mingxiang Chen

- Responsible for error analysis and various visualization prospect of the project
- Major contribution to performance evaluation scripts on validation set and CodaLab submission scripts
- Initial drafter and major contributor to the paper and the poster

7 Acknowledgments

We would like to thank the teaching staff of CS224N for their support and the generosity of Microsoft for the GPU credits they provided on Microsoft Azure.

References

[1] Rajpurkar P., Zhang J., Lopyrev K., Liang P. (2016) SQuAD: 100,000+ Questions for Machine Comprehension of Text, *Empirical Methods in Natural Language Processing (EMNLP 2016)*

[2] Xiong C., Zhong V., Socher R., (2017) Dynamic Coattention Networks for Question Answering, *International Conference on Learning Representations (ICLR 2017)*

[3] Seo M., Kembhavi A., Farhadi A., Hajishirzi H. (2017) Bi-Directional Attention Flow for Machine Comprehension, *International Conference on Learning Representations (ICLR 2017)*

[4] Pennington J., Socher R., Manning C.D. (2014) Glove: Global Vectors for Word Representation, *Empirical Methods in Natural Language Processing (EMNLP 2014)*

[5] Hochreiter S., Schmidhuber J., Long short-term memory, *Neural Computation, 1997*

[6] Wang Z., Mi H., Hamza W., Florian R. (2016) Multi-Perspective Context Matching for Machine Comprehension, *In eprint arXiv:1612.04211*

[7] Chen D., Bolton J., Manning, C.D. (2016) A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task, *Conference Proceedings of Association for Computational Linguistics (ACL 2017)*

[8] Kingma D.P., Ba J. (2014) Adam: A Method for Stochastic Optimization *International Conference for Learning Representations (ICLR 2015)*