ECNet: Early Attention and Local Convolution for Machine Comprehension

Abhishek Goswami
Microsoft
Redmond, WA 98052
agoswami@microsoft.com

Abstract

Current state-of-the-art machine comprehension models have some common tenets. One, the use of an embedding encoder layer over the input embeddings, to exploit contextual cues from surrounding words. Two, the use of neural attention mechanisms, to exploit matching cues between the context and the query. In this paper we propose ECNet, a hierarchical model for machine comprehension, which amplifies existing models by adding early attention and local convolution. It uses self-attention over the input embeddings as a form of early attention. For the embedding encoder layer, it models both sequential and local interactions between words, using recurrent and convolution layers respectively. On the SQuAD 2.0 dataset, ECNet outperforms the BiDAF model proposed recently in literature

1 Introduction

Machine comprehension (MC) and question answering (QA) tasks have gained significant interest in the past few years, with several end-to-end models showing promising results.

A key tenet of existing techniques is to use an embedding encoder layer. The input embedding layer first maps each word to a vector space. The embedding encoder layer then refines the embeddings using contextual cues from surrounding words. Approaches such as BiDAF [13] use a Long Short-Term Memory (LSTM) network at the embedding encoder layer to model temporal interactions between words, while more recent approaches such as QANet [17] use a combination of convolution and self-attention to model this layer. The embedding encoder layer plays a crucial in the model hierarchy with other layers stacked in a hierarchical fashion on top of it.

Another key factor in recent advancements has been the use of neural attention mechanisms, which extract useful signal by exploiting the notion of matching; either matching {context, query} sentences in QA tasks, or matching {source, target} language sentences in machine translation tasks. Several attention approaches have been proposed in literature. Chen et al [4] propose a uni-directional attention mechanism whereby the query attends to the context paragraph. In BiDAF, Seo at al [13] introduce bi-directional attention flow to obtain query-aware context representations. Wang et al [16] note that query-aware passage representations have limited knowledge of the context itself, and motivate self-matching attention to directly match the query-aware passage representation against itself.

In this paper we propose two novel extensions. First, we observe that given the input embeddings, most existing models dive straight into the embedding encoding layer. Attention layers come in later in the modeling stack, almost as an afterthought. We propose adding a embedding attention layer, as a form of early attention over the word and char embeddings. Second, we propose having a combination of recurrent and convolution layers in the embedding encoder layer. The motivation for these two novelties is to bring the benefits of *early attention* and *local convolution* into the model hierarchy. We show that adding these two extensions is indeed helpful, and helps outperform the BiDAF [13] model proposed recently in literature.

The remainder of the paper is organized as follows. In Section 2 we introduce our model. Section 3 presents the experimental results from our modeling techniques. In Section 4 we survey related work in the field of machine comprehension. Finally, we present our conclusions in Section 5

2 Model

In this section, we first formulate the machine comprehension problem. We then present our model, ECNet, for applying **E**arly attention and local **C**onvolution to deep neural **Net**works.

2.1 Problem Statement

The machine comprehension task considered in this paper is as follows. Given a context paragraph with T words, C = $\{c_1, c_2, ..., c_T\}$ and a query sentence with J words, Q = $\{q_1, q_2, ...\ q_J\}$, output a span S = $\{c_i, c_{i+1},...c_{i+j}\}$ from the original paragraph C that satisfactorily answers the question. Section 3.3 describes two metrics that are widely used in literature for evaluating this task. We use d to represent the hidden size used by several layers of the model.

Table 1: An example of a machine comprehension task.

Question	Economy, Energy and Tourism is one of the what?
Context	Subject Committees are established at the beginning of each parliamentary session, and again the members on each committee reflect the balance of parties across Parliament. Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment
Answer	current Subject Committees

2.2 Model Overview

Several state-of-the-art machine comprehension models have a similar structure. They have (a) an embedding layer (b) an embedding encoder layer (c) an attention flow layer (d) a model encoder layer and (e) an output layer.

We introduce two novel extensions to this structure. One, we add a Embedding Attention Layer between the input Embedding Layer and the Embedding Encoder Layer, with the goal of introducing early attention in the modeling process. Second, for the Embedding Encoder Layer we use a combination of recurrent and convolution operations to make it rich with both sequential and local interactions. Our machine comprehension model is thus a hierarchical multi-stage process consisting of six layers.

1. Embedding Layer.
2. Embedding Attention Layer.
3. Embedding Encoder Layer.
4. Attention Flow Layer.
5. Model Encoder Layer.
6. Output Layer.

The details of each of the layers are as follows.

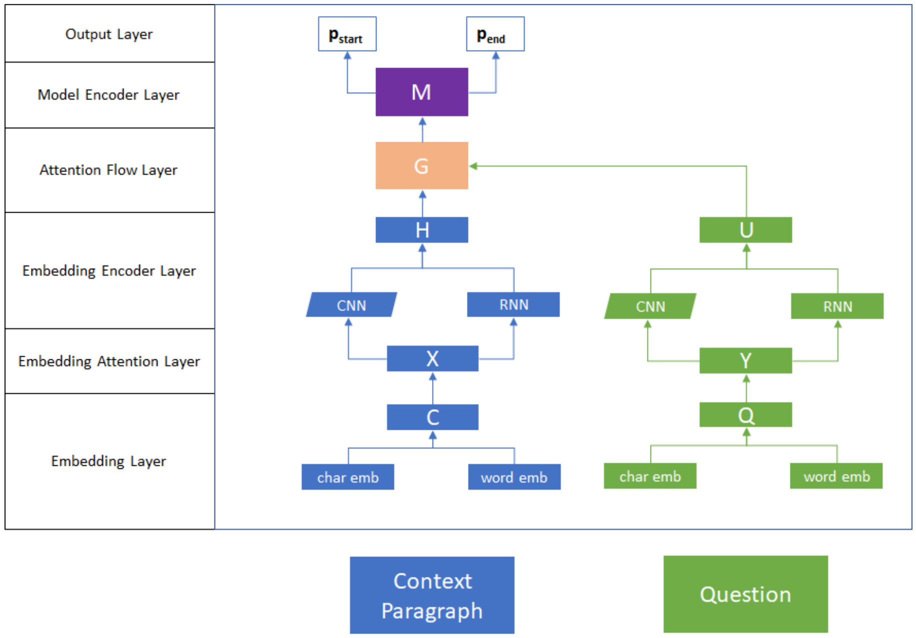

Figure 1: ECNet model architecture

1. Embedding Layer. In this layer we mix character embeddings with word embeddings.

For character embeddings, we use a method similar to that proposed by Kim et al [8]. We first convert a word to its character indices. We then pad (or truncate) each word so it has length m_{word}. For each of these characters we lookup a dense character embedding (which has shape e_{char}). To combine the character embeddings, we use 1-dimensional convolutions over m_{word} using e_{char} as the input channel size and e_{word} as the output channel size. The output of the CNN are max-pooled over the entire width to obtain a fixed-size vector of shape e_{word} for each word.

For word embeddings, we use pre-trained word vectors from GloVe [10] to obtain the fixed embedding for each word. The size of the word embeddings is e_{word} which is the same as the shape of the character-level embeddings for each word.

The concatenation of the character and word embeddings is passed to a Highway Network [14]. We do this for both the context sentence C and also the question Q. So now we have two matrices **C** $\in \mathbb{R}^{T,d}$ and **Q** $\in \mathbb{R}^{J,d}$ corresponding to the context and question respectively.

2. Embedding Attention Layer. The motivation for adding this layer is to attend to the embeddings provided by the previous layer. This layer starts early attention to the word and character character embeddings. As before, we do this for both the context sentence C and also the question Q , and concatenate the results with the embedding layer output giving us two matrices **X** $\in \mathbb{R}^{T,2d}$ and **Y** $\in \mathbb{R}^{J,2d}$ corresponding to the context and question respectively.

3. Embedding Encoder Layer. The purpose of this layer is to encode the relationships between the embeddings provided by the previous layers. On one hand we want to model the temporal interactions between words. For this we use a bi-directional LSTM. This results in two matrices of shape $(T, 2d)$ and $(J, 2d)$ corresponding to the context and question respectively.

We also model local interactions between the embeddings output by the embedding attention layer. We use 1-dimensional convolutions over the sequence length using $2d$ as both the input and output channel size. We do this using a kernel size 1, which results in two matrices of shape $(T, 2d)$ and $(J, 2d)$ corresponding to the context and question respectively.

The concatenation of the RNN and CNN layers gives us two matrices **H** $\in \mathbb{R}^{T,4d}$ and **U** $\in \mathbb{R}^{J,4d}$ respectively.

4. Attention Flow Layer. We also add a bi-directional attentional flow layer introduced by Seo et al [13]. The main idea is that attention should flow both ways - from the context to the question and from the question to the context. The attention flow layer also fuses the information between the context and the query words.

The inputs to the layer are contextual vector representations of the context **H** and the query **U**. The outputs of the layer is $\mathbf{G} \in \mathbb{R}^{T,16d}$ which is a query-aware vector representations of the context words, along with the embeddings from the previous layer.

5. Model Encoder Layer. This layer encodes the query-aware representations of the context words. The input is **G**, and the output is matrix **M**, which captures the interaction among the context words conditioned on the query. We use two layers of bi-directional LSTM, with hidden size d for each direction. Matrix $\mathbf{M} \in \mathbb{R}^{T,2d}$ is then passed to the Output Layer.

6. Output Layer. This layer is application specific. For the QA task being explored in this project, we need to find a sub-phrase of the context to answer the query. The phrase is derived by predicting the start and end indices of the phrase in the paragraph. The output layer produces two probability distribution $\boldsymbol{p}_{start}, \boldsymbol{p}_{end} \in \mathbb{R}^N$ corresponding to each position in the context.

$$\boldsymbol{p}_{start} = \text{softmax}(\boldsymbol{W}_{start}[\boldsymbol{G}; \boldsymbol{M}]). \tag{1}$$

$$\boldsymbol{p}_{end} = \text{softmax}(\boldsymbol{W}_{end}[\boldsymbol{G}; \boldsymbol{M'}]). \tag{2}$$

where $\boldsymbol{M'} \in \mathbb{R}^{T,2d}$ is a matrix obtained by applying a bi-directional LSTM to $\boldsymbol{M}$.

2.3 Model Training and Scoring

Training. We define the training loss as the sum of the negative log-likelihood (cross-entropy) loss for the start and end locations. So for a (context, question) pair with *start* index i $\in \{1, 2, ..., T\}$ and *end* index j $\in \{1, 2, ..., T\}$

$$\text{loss} = -\log \boldsymbol{p}_{start}(i) - \log \boldsymbol{p}_{end}(j). \tag{3}$$

During training, we average across the batch and use the Adadelta optimizer [18] to minimize the loss.

Scoring. At test time, we chose the pair (i,j) of indices that maximizes $\boldsymbol{p}_{start}(i).\boldsymbol{p}_{end}(j)$ subject to $i \leq j$ and $j - i + 1 \leq L_{max}$, where L_{max} is a hyperparameter which sets the maximum length of a predicted answer.

No Answer. We adopt the approach proposed by Levy et al [9]. We prepend a OOV token to the beginning of each sequence. The model outputs $\boldsymbol{p}_{start}$ and $\boldsymbol{p}_{end}$ soft-predictions as usual. When discretizing a prediction, if $\boldsymbol{p}_{start}(0) \cdot \boldsymbol{p}_{end}(0)$ is greater than any predicted answer span, the model predicts no-answer. Otherwise the model predicts the highest probability span. Note, this approach also allows us to predict a per-example confidence score that the question is unanswerable.

3 Experiment

In this section, we conduct experiments to study the performance of our models. We will benchmark our models on the Stanford Question Answering Dataset (SQuAD) 2.0 [12], considered to be one of the most competitive datasets in QA tasks. We also provide some implementation details for our models and present the main results.

3.1 Dataset

We consider the Stanford Question Answering Dataset (SQuAD) 2.0 [12] for machine comprehension. Our model is given a paragraph, and a question about that paragraph, as input. The goal is to answer

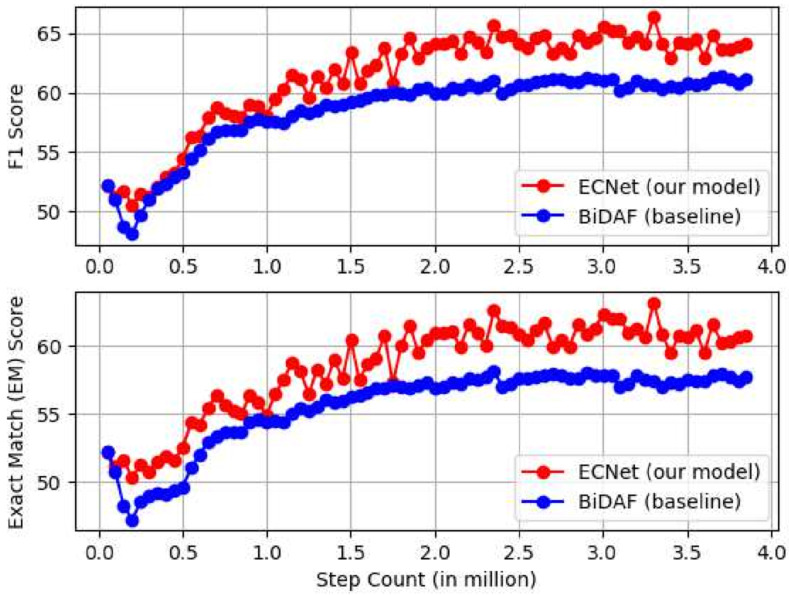

Figure 2: Comparison of F1 and EM scores while training.

the question correctly. There are around 150k questions, and roughly half of the questions cannot be answered using the provided paragraph.

3.1.1 Data splits

The official SQuAD dataset has three splits: train, dev and test. The train and dev sets are publicly available and the test set is entirely secret. For this project we use a custom dev and test set obtained by splitting the official dev set in half.

To summarize we have the following data splits:

- **Train**. 129,941 examples. All taken from the official SQuAD 2.0 training set.
- **Dev**. 6,078 examples. Roughly half of the official dev set, randomly selected.
- **Test**. 5,915 examples. The remaining examples from the official dev set.

From now on we refer to these splits as the train set, dev set and test set respectively. We will use the train set to train the model. We report the performance metrics on the dev set.

3.2 Training Details

The model architecture used for this task is shown in Figure 1.

For the Embedding Layer, m_{word} is set to 16. e_{char} and e_{word} are set to 64 and 300 respectively. We use one 1D filter for the CNN char embedding with a kernel size of 5. The hidden state d of the model is 100. For the convolutions in the Embedding Encoder Layer, we use a set of 4 stacked CNN layers, each with input/output channels as $2d$ with a kernel size of 1 . We uses dropout as a form of regularization across all the six layers in our model. Table 4 shows the effect of dropout on our model performance. We use the Adadelta optimizer [18] with a learning rate of 0.5 which is kept fixed. While training we use a batch size of 64. When scoring, L_{max} is set to 15.

We implement our model in Python using PyTorch [2]. The experiments are carried out on a Azure Data Science Virtual Machine (DSVM) [1] which has a NVIDIA Tesla K80 GPU.

Table 2: Comparing ECNet with BiDAF

	EM	F1
BiDAF with character embedding (baseline)	59.47	62.46
ECNet (our model)	63.17	66.38

Table 3: Results from Ablation Study

	EM	F1
No Embedding Attention Layer	60.93	64.34
No CNN layers inside the Embedding Encoder Layer	59.64	63.10
No character embedding in the Embedding Layer	59.47	62.46
Un-freezing the character and word embeddings in the Embedding Layer	62.19	65.39

3.3 Metric Details

We measure performance via two metrics: Exact Match (EM) and the F1 score.

- **Exact Match** is a binary measure (i.e. true/false) of whether the system output matches the ground truth answer exactly.
- **F1** is the harmonic mean of precision and recall.
- When a question has no answer, both the F1 and EM score are 1 if the model predicts no-answer, and 0 otherwise.
- For questions that do have answers, when evaluating on the dev or test sets, we take the maximum F1 and EM scores across the three human-provided answers for that question.

3.4 Results

Table 2 shows the comparison between our model and the baseline. As per the original BiDAF model, we include a character-level embedding layer using character-level convnets. This gives us a very strong baseline to compare with. We see ECNet outperforms BiDAF on both the F1 and EM metrics. Figure 2 shows a comparison of the metrics while training the two models.

3.5 Ablation Study

Table 3 shows the performance of the model and its ablations on the dev set. Having an embedding attention layer helps model performance. This validates our hypothesis that adding attention layers early in the model stack should help performance. For ablating the effect of the CNN layers, we experiment by removing the CNN layers from the embedding encoder layer. CNN layers prove to be critical, with a drop of 3 points on both metrics when absent. Char embeddings in the embedding layer also play a crucial role, whereby word-level embeddings represent the semantics of each word as a whole, while char-level embeddings better handle out-of-vocab (OOV) or rare words. Interestingly we also see that un-freezing the char-level and word-level embeddings gives us slightly lower performance. It seems the model gives slightly worse results if we allow the backpropagation to happen all the way through the embedding layer. This shows the pre-loaded embeddings are quite good, fine-tuning these embeddings do not generalize well to unseen data in the dev set.

Table 4: Effect of dropoput

	EM	F1
No Dropout	60.41	63.46
Dropout = 0.1	62.06	65.39
Dropout = 0.2 (chosen)	63.17	66.38
Dropout = 0.3	59.84	63.81
Dropout = 0.4	61.10	64.09

Table 3 shows the effect of dropout rates. The model overfits with low dropout rates. High drop-out rates help in preventing overfitting, but lead to lower EM/F1 scores. We settle on 0.2 as the dropout rate since it gives the best results.

4 Related Work

Machine comprehension (MC) and question answering (QA) tasks have gained significant interest in the past few years. In this section we provide a brief overview of some of the fundamental techniques in this field.

Use of Attention mechanisms. The use of attention mechanisms is one of the key tenets of existing MC literature. Several techniques [3, 6] use a dynamic attention mechanism, in which the attention weights are updated dynamically using the query, context and previous attention. Chen at al [4] show that simply using a bi-linear term for computing the attention weights improves model accuracy. In this paper, we use a memory-less attention mechanism similar to BiDAF [13]. We introduce the notion of early attention, and show how it helps model performance.

RNN, CNN and Transformer architectures. Recurrent Neural Networks have traditionally been the model of choice to capture the sequential nature of text. While common, RNNs are slow because of their recursive definition, which prevents parallel computation. Another drawback of RNNs is difficulty in modeling long dependencies, an issue which has prompted the use of Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) architectures for NLP tasks. Attempts have also been made to replace recurrent networks by Convolution Neural Networks (CNN) [7]. More recently, full attention architectures [15] have been shown to be effective across a wide variety of tasks, such as text classification, machine translation, question answering and sentiment analysis. A combination of convolutions and self-attention [17] has also been shown to have promising results. In this paper, we use the best of all worlds using a combination of RNN, CNN and attention mechanisms in a hierarchical model.

Pre-trained Contextual Embeddings. Several state of the art techniques leverage pre-trained contextual embeddings (PCE). Examples of such PCE-based techniques are ELMo [11] and BERT [5]. The core idea of such techniques is to represent a piece of text using word embeddings that depend on the context in which the word appears in the text. This is typically achieved by pretraining the weights on a large-scale language modeling dataset, and using the pre-trained weights for the initial model layers. In this paper we do not use PCE-based techniques. As mentioned in Section 2.2 we use GloVe embeddings to represent our word vectors. We can extend our model to incorporate PCE-based techniques.

5 Conclusion

In this paper we propose ECNet, a hierarchical model for machine comprehension, with focus on early attention and local convolution. The goal is to use attention early in the modeling stack, and to capture local interactions using convolution layers. On the SQuAD 2.0 dataset, ECNet outperforms the BiDAF [13] model proposed recently in literature. Ablation analyses demonstrate the effect of the novelties proposed in the model. Future work involves extending ECNet to incorporate pre-trained contextual embeddings (PCE) in the embedding layer.

References

[1] Azure data science virtual machines. https://azure.microsoft.com/en-us/services/virtual-machines/data-science-virtual-machines/.

[2] Pytorch. https://pytorch.org/.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] D. Chen, J. Bolton, and C. D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*, 2016.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[7] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[8] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[9] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

[10] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[12] P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[13] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[14] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[15] 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*, pages 5998–6008, 2017.

[16] 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)*, volume 1, pages 189–198, 2017.

[17] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[18] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.