BI-DIRECTIONAL ATTENTION FLOW FOR MACHINE COMPREHENSION

Minjoon Seo[1]* **Aniruddha Kembhavi**[2] **Ali Farhadi**[1,2] **Hananneh Hajishirzi**[1]
University of Washington[1], Allen Institute for Artificial Intelligence[2]
{minjoon,ali,hannaneh}@cs.washington.edu, {anik}@allenai.org

arXiv:1611.01603v6 [cs.CL] 21 Jun 2018

ABSTRACT

Machine comprehension (MC), answering a query about a given context paragraph, requires modeling complex interactions between the context and the query. Recently, attention mechanisms have been successfully extended to MC. Typically these methods use attention to focus on a small portion of the context and summarize it with a fixed-size vector, couple attentions temporally, and/or often form a uni-directional attention. In this paper we introduce the Bi-Directional Attention Flow (BIDAF) network, a multi-stage hierarchical process that represents the context at different levels of granularity and uses bi-directional attention flow mechanism to obtain a query-aware context representation without early summarization. Our experimental evaluations show that our model achieves the state-of-the-art results in Stanford Question Answering Dataset (SQuAD) and CNN/DailyMail cloze test.

1 INTRODUCTION

The tasks of machine comprehension (MC) and question answering (QA) have gained significant popularity over the past few years within the natural language processing and computer vision communities. Systems trained end-to-end now achieve promising results on a variety of tasks in the text and image domains. One of the key factors to the advancement has been the use of neural attention mechanism, which enables the system to focus on a targeted area within a context paragraph (for MC) or within an image (for Visual QA), that is most relevant to answer the question (Weston et al., 2015; Antol et al., 2015; Xiong et al., 2016a). Attention mechanisms in previous works typically have one or more of the following characteristics. First, the computed attention weights are often used to extract the most relevant information from the context for answering the question by summarizing the context into a fixed-size vector. Second, in the text domain, they are often temporally dynamic, whereby the attention weights at the current time step are a function of the attended vector at the previous time step. Third, they are usually uni-directional, wherein the query attends on the context paragraph or the image.

In this paper, we introduce the Bi-Directional Attention Flow (BIDAF) network, a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity (Figure 1). BIDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. Our attention mechanism offers following improvements to the previously popular attention paradigms. First, our attention layer is not used to summarize the context paragraph into a fixed-size vector. Instead, the attention is computed for every time step, and the attended vector at each time step, along with the representations from previous layers, is allowed to *flow* through to the subsequent modeling layer. This reduces the information loss caused by early summarization. Second, we use a memory-*less* attention mechanism. That is, while we iteratively compute attention through time as in Bahdanau et al. (2015), the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. We hypothesize that this simplification leads to the division of labor between the attention layer and the modeling layer. It forces the attention layer to focus on learning the attention between the query and the context, and enables the modeling layer to focus on learning the interaction within the

*The majority of the work was done while the author was interning at the Allen Institute for AI.

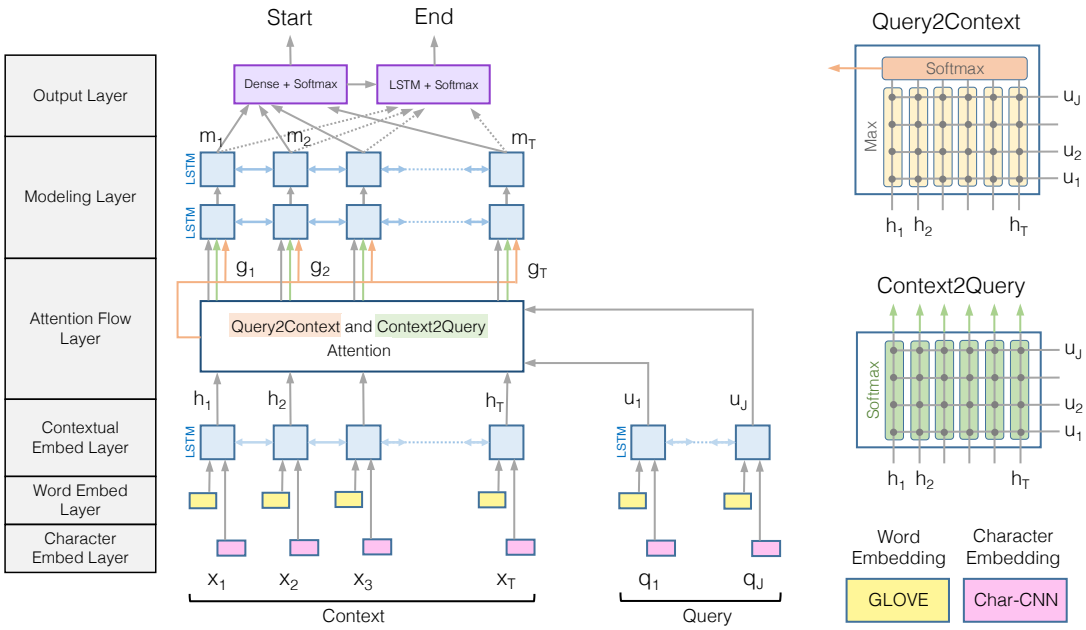

Figure 1: BiDirectional Attention Flow Model *(best viewed in color)*

query-aware context representation (the output of the attention layer). It also allows the attention at each time step to be unaffected from incorrect attendances at previous time steps. Our experiments show that memory-less attention gives a clear advantage over dynamic attention. Third, we use attention mechanisms in both directions, query-to-context and context-to-query, which provide complimentary information to each other.

Our BIDAF model[1] outperforms all previous approaches on the highly-competitive Stanford Question Answering Dataset (SQuAD) test set leaderboard at the time of submission. With a modification to only the output layer, BIDAF achieves the state-of-the-art results on the CNN/DailyMail cloze test. We also provide an in-depth ablation study of our model on the SQuAD development set, visualize the intermediate feature spaces in our model, and analyse its performance as compared to a more traditional language model for machine comprehension (Rajpurkar et al., 2016).

2 MODEL

Our machine comprehension model is a hierarchical multi-stage process and consists of six layers (Figure 1):

1. **Character Embedding Layer** maps each word to a vector space using character-level CNNs.
2. **Word Embedding Layer** maps each word to a vector space using a pre-trained word embedding model.
3. **Contextual Embedding Layer** utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.
4. **Attention Flow Layer** couples the query and context vectors and produces a set of query-aware feature vectors for each word in the context.
5. **Modeling Layer** employs a Recurrent Neural Network to scan the context.
6. **Output Layer** provides an answer to the query.

[1]Our code and interactive demo are available at: `allenai.github.io/bi-att-flow/`

1. Character Embedding Layer. Character embedding layer is responsible for mapping each word to a high-dimensional vector space. Let $\{\boldsymbol{x}_1, \dots \boldsymbol{x}_T\}$ and $\{\boldsymbol{q}_1, \dots \boldsymbol{q}_J\}$ represent the words in the input context paragraph and query, respectively. Following Kim (2014), we obtain the character-level embedding of each word using Convolutional Neural Networks (CNN). Characters are embedded into vectors, which can be considered as 1D inputs to the CNN, and whose size is the input channel size of the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word.

2. Word Embedding Layer. Word embedding layer also maps each word to a high-dimensional vector space. We use pre-trained word vectors, GloVe (Pennington et al., 2014), to obtain the fixed word embedding of each word.

The concatenation of the character and word embedding vectors is passed to a two-layer Highway Network (Srivastava et al., 2015). The outputs of the Highway Network are two sequences of d-dimensional vectors, or more conveniently, two matrices: $\mathbf{X} \in \mathbb{R}^{d \times T}$ for the context and $\mathbf{Q} \in \mathbb{R}^{d \times J}$ for the query.

3. Contextual Embedding Layer. We use a Long Short-Term Memory Network (LSTM) (Hochreiter & Schmidhuber, 1997) on top of the embeddings provided by the previous layers to model the temporal interactions between words. We place an LSTM in both directions, and concatenate the outputs of the two LSTMs. Hence we obtain $\mathbf{H} \in \mathbb{R}^{2d \times T}$ from the context word vectors $\mathbf{X}$, and $\mathbf{U} \in \mathbb{R}^{2d \times J}$ from query word vectors $\mathbf{Q}$. Note that each column vector of $\mathbf{H}$ and $\mathbf{U}$ is $2d$-dimensional because of the concatenation of the outputs of the forward and backward LSTMs, each with d-dimensional output.

It is worth noting that the first three layers of the model are computing features from the query and context at different levels of granularity, akin to the multi-stage feature computation of convolutional neural networks in the computer vision field.

4. Attention Flow Layer. Attention flow layer is responsible for linking and fusing information from the context and the query words. Unlike previously popular attention mechanisms (Weston et al., 2015; Hill et al., 2016; Sordoni et al., 2016; Shen et al., 2016), the attention flow layer is not used to summarize the query and context into single feature vectors. Instead, the attention vector at each time step, along with the embeddings from previous layers, are allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization.

The inputs to the layer are contextual vector representations of the context $\mathbf{H}$ and the query $\mathbf{U}$. The outputs of the layer are the query-aware vector representations of the context words, $\mathbf{G}$, along with the contextual embeddings from the previous layer.

In this layer, we compute attentions in two directions: from context to query as well as from query to context. Both of these attentions, which will be discussed below, are derived from a shared similarity matrix, $\mathbf{S} \in \mathbb{R}^{T \times J}$, between the contextual embeddings of the context ($\mathbf{H}$) and the query ($\mathbf{U}$), where $\mathbf{S}_{tj}$ indicates the similarity between t-th context word and j-th query word. The similarity matrix is computed by

$$\mathbf{S}_{tj} = \alpha(\mathbf{H}_{:t}, \mathbf{U}_{:j}) \in \mathbb{R} \tag{1}$$

where α is a trainable scalar function that encodes the similarity between its two input vectors, $\mathbf{H}_{:t}$ is t-th column vector of $\mathbf{H}$, and $\mathbf{U}_{:j}$ is j-th column vector of $\mathbf{U}$, We choose $\alpha(\mathbf{h}, \mathbf{u}) = \mathbf{w}_{(\mathbf{S})}^\top [\mathbf{h}; \mathbf{u}; \mathbf{h} \circ \mathbf{u}]$, where $\mathbf{w}_{(\mathbf{S})} \in \mathbb{R}^{6d}$ is a trainable weight vector, $\circ$ is elementwise multiplication, $[;]$ is vector concatenation across row, and implicit multiplication is matrix multiplication. Now we use $\mathbf{S}$ to obtain the attentions and the attended vectors in both directions.

Context-to-query Attention. Context-to-query (C2Q) attention signifies which query words are most relevant to each context word. Let $\mathbf{a}_t \in \mathbb{R}^J$ represent the attention weights on the query words by t-th context word, $\sum \mathbf{a}_{tj} = 1$ for all t. The attention weight is computed by $\mathbf{a}_t = \mathrm{softmax}(\mathbf{S}_{t:}) \in \mathbb{R}^J$, and subsequently each attended query vector is $\tilde{\mathbf{U}}_{:t} = \sum_j \mathbf{a}_{tj} \mathbf{U}_{:j}$. Hence $\tilde{\mathbf{U}}$ is a $2d$-by-T matrix containing the attended query vectors for the entire context.

Query-to-context Attention. Query-to-context (Q2C) attention signifies which context words have the closest similarity to one of the query words and are hence critical for answering the query.

We obtain the attention weights on the context words by $\mathbf{b} = \text{softmax}(\max_{col}(\mathbf{S})) \in \mathbb{R}^T$, where the maximum function ($\max_{col}$) is performed across the column. Then the attended context vector is $\tilde{\mathbf{h}} = \sum_t \mathbf{b}_t \mathbf{H}_{:t} \in \mathbb{R}^{2d}$. This vector indicates the weighted sum of the most important words in the context with respect to the query. $\tilde{\mathbf{h}}$ is tiled T times across the column, thus giving $\tilde{\mathbf{H}} \in \mathbb{R}^{2d \times T}$.

Finally, the contextual embeddings and the attention vectors are combined together to yield $\mathbf{G}$, where each column vector can be considered as the query-aware representation of each context word. We define $\mathbf{G}$ by

$$\mathbf{G}_{:t} = \boldsymbol{\beta}(\mathbf{H}_{:t}, \tilde{\mathbf{U}}_{:t}, \tilde{\mathbf{H}}_{:t}) \in \mathbb{R}^{d_{\mathbf{G}}} \tag{2}$$

where $\mathbf{G}_{:t}$ is the t-th column vector (corresponding to t-th context word), $\boldsymbol{\beta}$ is a trainable vector function that fuses its (three) input vectors, and $d_{\mathbf{G}}$ is the output dimension of the $\boldsymbol{\beta}$ function. While the $\boldsymbol{\beta}$ function can be an arbitrary trainable neural network, such as multi-layer perceptron, a simple concatenation as following still shows good performance in our experiments: $\boldsymbol{\beta}(\mathbf{h}, \tilde{\mathbf{u}}, \tilde{\mathbf{h}}) = [\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{h}}] \in \mathbb{R}^{8d \times T}$ (i.e., $d_{\mathbf{G}} = 8d$).

5. Modeling Layer. The input to the modeling layer is $\mathbf{G}$, which encodes the query-aware representations of context words. The output of the modeling layer captures the interaction among the context words conditioned on the query. This is different from the contextual embedding layer, which captures the interaction among context words independent of the query. We use two layers of bi-directional LSTM, with the output size of d for each direction. Hence we obtain a matrix $\mathbf{M} \in \mathbb{R}^{2d \times T}$, which is passed onto the output layer to predict the answer. Each column vector of $\mathbf{M}$ is expected to contain contextual information about the word with respect to the entire context paragraph and the query.

6. Output Layer. The output layer is application-specific. The modular nature of BIDAF allows us to easily swap out the output layer based on the task, with the rest of the architecture remaining exactly the same. Here, we describe the output layer for the QA task. In section 5, we use a slight modification of this output layer for cloze-style comprehension.

The QA task requires the model to find a sub-phrase of the paragraph to answer the query. The phrase is derived by predicting the start and the end indices of the phrase in the paragraph. We obtain the probability distribution of the start index over the entire paragraph by

$$\mathbf{p}^1 = \text{softmax}(\mathbf{w}_{(\mathbf{p}^1)}^\top [\mathbf{G}; \mathbf{M}]), \tag{3}$$

where $\mathbf{w}_{(\mathbf{p}^1)} \in \mathbb{R}^{10d}$ is a trainable weight vector. For the end index of the answer phrase, we pass $\mathbf{M}$ to another bidirectional LSTM layer and obtain $\mathbf{M}^2 \in \mathbb{R}^{2d \times T}$. Then we use $\mathbf{M}^2$ to obtain the probability distribution of the end index in a similar manner:

$$\mathbf{p}^2 = \text{softmax}(\mathbf{w}_{(\mathbf{p}^2)}^\top [\mathbf{G}; \mathbf{M}^2]) \tag{4}$$

Training. We define the training loss (to be minimized) as the sum of the negative log probabilities of the true start and end indices by the predicted distributions, averaged over all examples:

$$L(\theta) = -\frac{1}{N} \sum_i^N \log(\mathbf{p}_{y_i^1}^1) + \log(\mathbf{p}_{y_i^2}^2) \tag{5}$$

where θ is the set of all trainable weights in the model (the weights and biases of CNN filters and LSTM cells, $\mathbf{w}_{(\mathbf{S})}$, $\mathbf{w}_{(\mathbf{p}^1)}$ and $\mathbf{w}_{(\mathbf{p}^2)}$), N is the number of examples in the dataset, y_i^1 and y_i^2 are the true start and end indices of the i-th example, respectively, and $\mathbf{p}_k$ indicates the k-th value of the vector $\mathbf{p}$.

Test. The answer span (k, l) where $k \leq l$ with the maximum value of $\mathbf{p}_k^1 \mathbf{p}_l^2$ is chosen, which can be computed in linear time with dynamic programming.

3 Related Work

Machine comprehension. A significant contributor to the advancement of MC models has been the availability of large datasets. Early datasets such as MCTest (Richardson et al., 2013) were too

small to train end-to-end neural models. Massive cloze test datasets (CNN/DailyMail by Hermann et al. (2015) and Childrens Book Test by Hill et al. (2016)), enabled the application of deep neural architectures to this task. More recently, Rajpurkar et al. (2016) released the Stanford Question Answering (SQuAD) dataset with over 100,000 questions. We evaluate the performance of our comprehension system on both SQuAD and CNN/DailyMail datasets.

Previous works in end-to-end machine comprehension use attention mechanisms in three distinct ways. The first group (largely inspired by Bahdanau et al. (2015)) uses a dynamic attention mechanism, in which the attention weights are updated dynamically given the query and the context as well as the previous attention. Hermann et al. (2015) argue that the dynamic attention model performs better than using a single fixed query vector to attend on context words on CNN & DailyMail datasets. Chen et al. (2016) show that simply using bilinear term for computing the attention weights in the same model drastically improves the accuracy. Wang & Jiang (2016) reverse the direction of the attention (attending on query words as the context RNN progresses) for SQuAD. In contrast to these models, BIDAF uses a memory-less attention mechanism.

The second group computes the attention weights once, which are then fed into an output layer for final prediction (e.g., Kadlec et al. (2016)). Attention-over-attention model (Cui et al., 2016) uses a 2D similarity matrix between the query and context words (similar to Equation 1) to compute the weighted average of query-to-context attention. In contrast to these models, BIDAF does not summarize the two modalities in the attention layer and instead lets the attention vectors flow into the modeling (RNN) layer.

The third group (considered as variants of Memory Network (Weston et al., 2015)) repeats computing an attention vector between the query and the context through multiple layers, typically referred to as *multi-hop* (Sordoni et al., 2016; Dhingra et al., 2016). Shen et al. (2016) combine Memory Networks with Reinforcement Learning in order to dynamically control the number of hops. One can also extend our BIDAF model to incorporate multiple hops.

Visual question answering. The task of question answering has also gained a lot of interest in the computer vision community. Early works on visual question answering (VQA) involved encoding the question using an RNN, encoding the image using a CNN and combining them to answer the question (Antol et al., 2015; Malinowski et al., 2015). Attention mechanisms have also been successfully employed for the VQA task and can be broadly clustered based on the granularity of their attention and the approach to construct the attention matrix. At the coarse level of granularity, the question attends to different patches in the image (Zhu et al., 2016; Xiong et al., 2016a). At a finer level, each question word attends to each image patch and the highest attention value for each spatial location (Xu & Saenko, 2016) is adopted. A hybrid approach is to combine questions representations at multiple levels of granularity (unigrams, bigrams, trigrams) (Yang et al., 2015). Several approaches to constructing the attention matrix have been used including element-wise product, element-wise sum, concatenation and Multimodal Compact Bilinear Pooling (Fukui et al., 2016).

Lu et al. (2016) have recently shown that in addition to attending from the question to image patches, attending from the image back to the question words provides an improvement on the VQA task. This finding in the visual domain is consistent with our finding in the language domain, where our bi-directional attention between the query and context provides improved results. Their model, however, uses the attention weights directly in the output layer and does not take advantage of the attention flow to the modeling layer.

4 QUESTION ANSWERING EXPERIMENTS

In this section, we evaluate our model on the task of question answering using the recently released SQuAD (Rajpurkar et al., 2016), which has gained a huge attention over a few months. In the next section, we evaluate our model on the task of cloze-style reading comprehension.

Dataset. SQuAD is a machine comprehension dataset on a large set of Wikipedia articles, with more than 100,000 questions. The answer to each question is always a span in the context. The model is given a credit if its answer matches one of the human written answers. Two metrics are used to evaluate models: Exact Match (EM) and a softer metric, F1 score, which measures the weighted average of the precision and recall rate at character level. The dataset consists of 90k/10k

	Single Model		Ensemble	
	EM	F1	EM	F1
Logistic Regression Baseline[a]	40.4	51.0	-	-
Dynamic Chunk Reader[b]	62.5	71.0	-	-
Fine-Grained Gating[c]	62.5	73.3	-	-
Match-LSTM[d]	64.7	73.7	67.9	77.0
Multi-Perspective Matching[e]	65.5	75.1	68.2	77.2
Dynamic Coattention Networks[f]	66.2	75.9	71.6	80.4
R-Net[g]	**68.4**	**77.5**	72.1	79.7
BIDAF (Ours)	68.0	77.3	**73.3**	**81.1**

(a) Results on the SQuAD test set

	EM	F1
No char embedding	65.0	75.4
No word embedding	55.5	66.8
No C2Q attention	57.2	67.7
No Q2C attention	63.6	73.7
Dynamic attention	63.5	73.6
BIDAF (single)	67.7	77.3
BIDAF (ensemble)	72.6	80.7

(b) Ablations on the SQuAD dev set

Table 1: (1a) The performance of our model BIDAF and competing approaches by Rajpurkar et al. (2016)[a], Yu et al. (2016)[b], Yang et al. (2016)[c], Wang & Jiang (2016)[d], IBM Watson[e] (unpublished), Xiong et al. (2016b)[f], and Microsoft Research Asia[g] (unpublished) on the SQuAD test set. A concurrent work by Lee et al. (2016) does not report the test scores. All results shown here reflect the SQuAD leaderboard (`stanford-qa.com`) as of 6 Dec 2016, 12pm PST. (1b) The performance of our model and its ablations on the SQuAD dev set. Ablation results are presented only for single runs.

train/dev question-context tuples with a large hidden test set. It is one of the largest available MC datasets with human-written questions and serves as a great test bed for our model.

Model Details. The model architecture used for this task is depicted in Figure 1. Each paragraph and question are tokenized by a regular-expression-based word tokenizer (PTB Tokenizer) and fed into the model. We use 100 1D filters for CNN char embedding, each with a width of 5. The hidden state size (d) of the model is 100. The model has about 2.6 million parameters. We use the AdaDelta (Zeiler, 2012) optimizer, with a minibatch size of 60 and an initial learning rate of 0.5, for 12 epochs. A dropout (Srivastava et al., 2014) rate of 0.2 is used for the CNN, all LSTM layers, and the linear transformation before the softmax for the answers. During training, the moving averages of all weights of the model are maintained with the exponential decay rate of 0.999. At test time, the moving averages instead of the raw weights are used. The training process takes roughly 20 hours on a single Titan X GPU. We also train an ensemble model consisting of 12 training runs with the identical architecture and hyper-parameters. At test time, we choose the answer with the highest sum of confidence scores amongst the 12 runs for each question.

Results. The results of our model and competing approaches on the hidden test are summarized in Table 1a. BIDAF (ensemble) achieves an EM score of 73.3 and an F1 score of 81.1, outperforming all previous approaches.

Ablations. Table 1b shows the performance of our model and its ablations on the SQuAD dev set. Both char-level and word-level embeddings contribute towards the model's performance. We conjecture that word-level embedding is better at representing the semantics of each word as a whole, while char-level embedding can better handle out-of-vocab (OOV) or rare words. To evaluate bi-directional attention, we remove C2Q and Q2C attentions. For ablating C2Q attention, we replace the attended question vector $\tilde{\mathbf{U}}$ with the average of the output vectors of the question's contextual embedding layer (LSTM). C2Q attention proves to be critical with a drop of more than 10 points on both metrics. For ablating Q2C attention, the output of the attention layer, $\mathbf{G}$, does not include terms that have the attended Q2C vectors, $\tilde{\mathbf{H}}$. To evaluate the attention flow, we study a dynamic attention model, where the attention is dynamically computed within the modeling layer's LSTM, following previous work (Bahdanau et al., 2015; Wang & Jiang, 2016). This is in contrast with our approach, where the attention is pre-computed before flowing to the modeling layer. Despite being a simpler attention mechanism, our proposed static attention outperforms the dynamically computed attention by more than 3 points. We conjecture that separating out the attention layer results in a richer set of features computed in the first 4 layers which are then incorporated by the modeling layer. We also show the performance of BIDAF with several different definitions of α and $\boldsymbol{\beta}$ functions (Equation 1 and 2) in Appendix B.

Layer	Query	Closest words in the Context using cosine similarity
Word	When	when, When, After, after, He, he, But, but, before, Before
Contextual	When	When, when, 1945, 1991, 1971, 1967, 1990, 1972, 1965, 1953
Word	Where	Where, where, It, IT, it, they, They, that, That, city
Contextual	Where	where, Where, Rotterdam, area, Nearby, location, outside, Area, across, locations
Word	Who	Who, who, He, he, had, have, she, She, They, they
Contextual	Who	who, whose, whom, Guiscard, person, John, Thomas, families, Elway, Louis
Word	city	City, city, town, Town, Capital, capital, district, cities, province, Downtown
Contextual	city	city, City, Angeles, Paris, Prague, Chicago, Port, Pittsburgh, London, Manhattan
Word	January	July, December, June, October, January, September, February, April, November, March
Contextual	January	January, March, December, August, December, July, July, July, March, December
Word	Seahawks	Seahawks, Broncos, 49ers, Ravens, Chargers, Steelers, quarterback, Vikings, Colts, NFL
Contextual	Seahawks	Seahawks, Broncos, Panthers, Vikings, Packers, Ravens, Patriots, Falcons, Steelers, Chargers
Word	date	date, dates, until, Until, June, July, Year, year, December, deadline
Contextual	date	date, dates, December, July, January, October, June, November, March, February

Table 2: Closest context words to a given query word, using a cosine similarity metric computed in the Word Embedding feature space and the Phrase Embedding feature space.

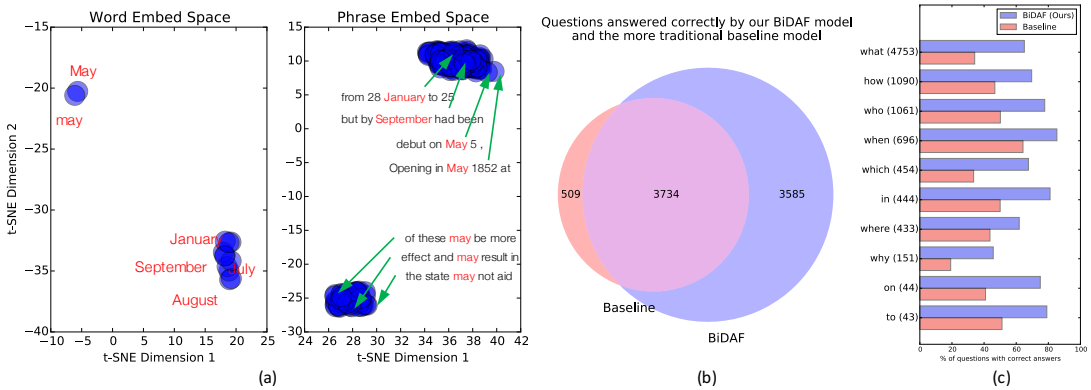

Figure 2: (a) t-SNE visualizations of the *months* names embedded in the two feature spaces. The contextual embedding layer is able to distinguish the two usages of the word *May* using context from the surrounding text. (b) Venn diagram of the questions answered correctly by our model and the *more traditional* baseline (Rajpurkar et al., 2016). (c) Correctly answered questions broken down by the 10 most frequent first words in the question.

Visualizations. We now provide a qualitative analysis of our model on the SQuAD dev set. First, we visualize the feature spaces after the word and contextual embedding layers. These two layers are responsible for aligning the embeddings between the query and context words which are the inputs to the subsequent attention layer. To visualize the embeddings, we choose a few frequent query words in the dev data and look at the context words that have the highest cosine similarity to the query words (Table 2). At the word embedding layer, query words such as *When*, *Where* and *Who* are not well aligned to possible answers in the context, but this dramatically changes in the contextual embedding layer which has access to context from surrounding words and is just 1 layer below the attention layer. *When* begins to match years, *Where* matches locations, and *Who* matches names.

We also visualize these two feature spaces using t-SNE in Figure 2. t-SNE is performed on a large fraction of dev data but we only plot data points corresponding to the months of the year. An interesting pattern emerges in the Word space, where *May* is separated from the rest of the months because *May* has multiple meanings in the English language. The contextual embedding layer uses contextual cues from surrounding words and is able to separate the usages of the word *May*. Finally we visualize the attention matrices for some question-context tuples in the dev data in Figure 3. In the first example, *Where* matches locations and in the second example, *many* matches quantities and numerical symbols. Also, entities in the question typically attend to the same entities in the context, thus providing a feature for the model to localize possible answers.

Discussions. We analyse the performance of our our model with a traditional language-feature-based baseline (Rajpurkar et al., 2016). Figure 2b shows a Venn diagram of the dev set questions correctly answered by the models. Our model is able to answer more than 86% of the questions

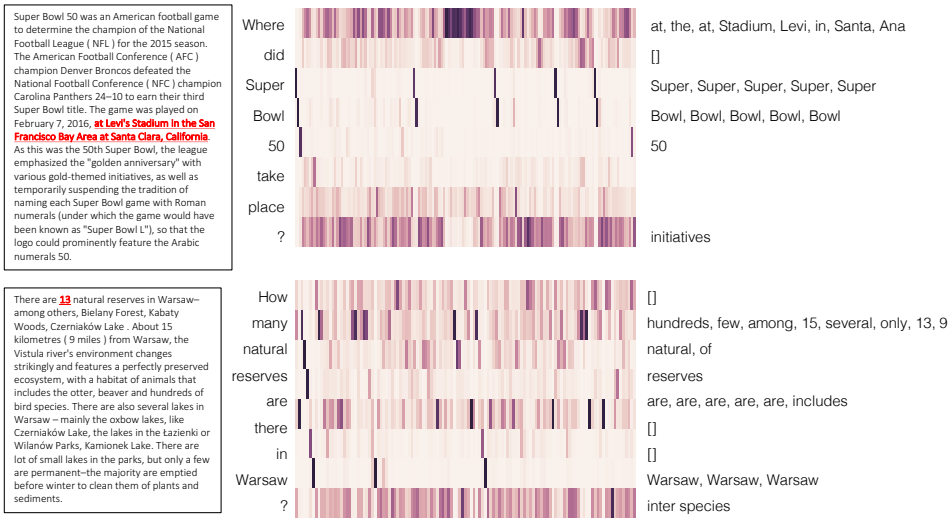

Figure 3: Attention matrices for question-context tuples. The left palette shows the context paragraph (correct answer in red and underlined), the middle palette shows the attention matrix (each row is a question word, each column is a context word), and the right palette shows the top attention points for each question word, above a threshold.

correctly answered by the baseline. The 14% that are incorrectly answered does not have a clear pattern. This suggests that neural architectures are able to exploit much of the information captured by the language features. We also break this comparison down by the first words in the questions (Figure 2c). Our model outperforms the traditional baseline comfortably in every category.

Error Analysis. We randomly select 50 incorrect questions (based on EM) and categorize them into 6 classes. 50% of errors are due to the imprecise boundaries of the answers, 28% involve syntactic complications and ambiguities, 14% are paraphrase problems, 4% require external knowledge, 2% need multiple sentences to answer, and 2% are due to mistakes during tokenization. See Appendix A for the examples of the error modes.

5 Cloze Test Experiments

We also evaluate our model on the task of cloze-style reading comprehension using the CNN and Daily Mail datasets (Hermann et al., 2015).

Dataset. In a cloze test, the reader is asked to fill in words that have been removed from a passage, for measuring one's ability to comprehend text. Hermann et al. (2015) have recently compiled a massive Cloze-style comprehension dataset, consisting of 300k/4k/3k and 879k/65k/53k (train/dev/test) examples from CNN and DailyMail news articles, respectively. Each example has a news article and an incomplete sentence extracted from the human-written summary of the article. To distinguish this task from language modeling and force one to refer to the article to predict the correct missing word, the missing word is always a named entity, anonymized with a random ID. Also, the IDs must be shuffled constantly during test, which is also critical for full anonymization.

Model Details. The model architecture used for this task is very similar to that for SQuAD (Section 4) with only a few small changes to adapt it to the cloze test. Since each answer in the CNN/DailyMail datasets is always a single word (entity), we only need to predict the start index ($\mathbf{p}^1$); the prediction for the end index ($\mathbf{p}^2$) is omitted from the loss function. Also, we mask out all non-entity words in the final classification layer so that they are forced to be excluded from possible answers. Another important difference from SQuAD is that the answer entity might appear more than once in the context paragraph. To address this, we follow a similar strategy from Kadlec et al. (2016). During training, after we obtain $\mathbf{p}^1$, we sum all probability values of the entity instances

in the context that correspond to the correct answer. Then the loss function is computed from the summed probability. We use a minibatch size of 48 and train for 8 epochs, with early stop when the accuracy on validation data starts to drop. Inspired by the window-based method (Hill et al., 2016), we split each article into short sentences where each sentence is a 19-word window around each entity (hence the same word might appear in multiple sentences). The RNNs in BIDAF are not feed-forwarded or back-propagated across sentences, which speed up the training process by parallelization. The entire training process takes roughly 60 hours on eight Titan X GPUs. The other hyper-parameters are identical to the model described in Section 4.

Results. The results of our single-run models and competing approaches on the CNN/DailyMail datasets are summarized in Table 3. * indicates ensemble methods. BIDAF outperforms previous single-run models on both datasets for both val and test data. On the DailyMail test, our single-run model even outperforms the best ensemble method.

	CNN		DailyMail	
	val	test	val	test
Attentive Reader (Hermann et al., 2015)	61.6	63.0	70.5	69.0
MemNN (Hill et al., 2016)	63.4	6.8	-	-
AS Reader (Kadlec et al., 2016)	68.6	69.5	75.0	73.9
DER Network (Kobayashi et al., 2016)	71.3	72.9	-	-
Iterative Attention (Sordoni et al., 2016)	72.6	73.3	-	-
EpiReader (Trischler et al., 2016)	73.4	74.0	-	-
Stanford AR (Chen et al., 2016)	73.8	73.6	77.6	76.6
GAReader (Dhingra et al., 2016)	73.0	73.8	76.7	75.7
AoA Reader (Cui et al., 2016)	73.1	74.4	-	-
ReasoNet (Shen et al., 2016)	72.9	74.7	77.6	76.6
BIDAF (Ours)	**76.3**	**76.9**	**80.3**	**79.6**
MemNN* (Hill et al., 2016)	66.2	69.4	-	-
ASReader* (Kadlec et al., 2016)	73.9	75.4	78.7	77.7
Iterative Attention* (Sordoni et al., 2016)	74.5	75.7	-	-
GA Reader* (Dhingra et al., 2016)	76.4	77.4	79.1	78.1
Stanford AR* (Chen et al., 2016)	77.2	77.6	80.2	79.2

Table 3: Results on CNN/DailyMail datasets. We also include the results of previous ensemble methods (marked with *) for completeness.

6 CONCLUSION

In this paper, we introduce BIDAF, a multi-stage hierarchical process that represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization. The experimental evaluations show that our model achieves the state-of-the-art results in Stanford Question Answering Dataset (SQuAD) and CNN/DailyMail cloze test. The ablation analyses demonstrate the importance of each component in our model. The visualizations and discussions show that our model is learning a suitable representation for MC and is capable of answering complex questions by attending to correct locations in the given paragraph. Future work involves extending our approach to incorporate multiple hops of the attention layer.

ACKNOWLEDGMENTS

This research was supported by the NSF (IIS 1616112), NSF (III 1703166), Allen Institute for AI (66-9175), Allen Distinguished Investigator Award, Google Research Faculty Award, and Samsung GRO Award. We thank the anonymous reviewers for their helpful comments.

REFERENCES

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*, 2016.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, 2016.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*, 2016.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *ACL*, 2016.

Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. Dynamic entity representation with max-pooling improves machine reading. In *NAACL-HLT*, 2016.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013.

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

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*, 2016.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. In *EMNLP*, 2016.

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

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *ICML*, 2016a.

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

Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*, 2016.

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

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*, 2015.

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

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *CVPR*, 2016.

A ERROR ANALYSIS

Table 4 summarizes the modes of errors by BIDAF and shows examples for each category of error in SQuAD.

Error type	Ratio (%)	Example
Imprecise answer boundaries	50	**Context**: "The Free Movement of Workers Regulation articles 1 to 7 set out the main provisions on equal treatment of workers." **Question**: "Which articles of the Free Movement of Workers Regulation set out the primary provisions on equal treatment of workers?" **Prediction**: "1 to 7", **Answer**: "articles 1 to 7"
Syntactic complications and ambiguities	28	**Context**: "A piece of paper was later found on which Luther had written his last statement. " **Question**: "What was later discovered written by Luther?" **Prediction**: "A piece of paper", **Answer**: "his last statement"
Paraphrase problems	14	**Context**: "Generally, education in Australia follows the three-tier model which includes primary education (primary schools), followed by secondary education (secondary schools/high schools) and tertiary education (universities and/or TAFE colleges)." **Question**: "What is the first model of education, in the Australian system?" **Prediction**: "three-tier", **Answer**: "primary education"
External knowledge	4	**Context**: "On June 4, 2014, the NFL announced that the practice of branding Super Bowl games with Roman numerals, a practice established at Super Bowl V, would be temporarily suspended, and that the game would be named using Arabic numerals as Super Bowl 50 as opposed to Super Bowl L." **Question**: "If Roman numerals were used in the naming of the 50th Super Bowl, which one would have been used?' **Prediction**: "Super Bowl 50", **Answer**: "L"
Multi-sentence	2	**Context**: "Over the next several years in addition to host to host interactive connections the network was enhanced to support terminal to host connections, host to host batch connections (remote job submission, remote printing, batch file transfer), interactive file transfer, gateways to the Tymnet and Telenet public data networks, X.25 host attachments, gateways to X.25 data networks, Ethernet attached hosts, and eventually TCP/IP and additional public universities in Michigan join the network. All of this set the stage for Merit's role in the NSFNET project starting in the mid-1980s." **Question**: "What set the stage for Merits role in NSFNET" **Prediction**: "All of this set the stage for Merit 's role in the NSFNET project starting in the mid-1980s", **Answer**: "Ethernet attached hosts, and eventually TCP/IP and additional public universities in Michigan join the network"
Incorrect preprocessing	2	**Context**: "English chemist John Mayow (1641-1679) refined this work by showing that fire requires only a part of air that he called spiritus nitroaereus or just nitroaereus." **Question**: "John Mayow died in what year?" **Prediction**: "1641-1679", **Answer**: "1679"

Table 4: Error analysis on SQuAD. We randomly selected EM-incorrect answers and classified them into 6 different categories. Only relevant sentence(s) from the context shown for brevity.

B VARIATIONS OF SIMILARITY AND FUSION FUNCTIONS

	EM	F1
Eqn. 1: dot product	65.5	75.5
Eqn. 1: linear	59.5	69.7
Eqn. 1: bilinear	61.6	71.8
Eqn. 1: linear after MLP	66.2	76.4
Eqn. 2: MLP after concat	67.1	77.0
BIDAF (single)	68.0	77.3

Table 5: Variations of similarity function α (Equation 1) and fusion function $\boldsymbol{\beta}$ (Equation 2) and their performance on the dev data of SQuAD. See Appendix B for the details of each variation.

In this appendix section, we experimentally demonstrate how different choices of the similarity function α (Equation 1) and the fusion function $\boldsymbol{\beta}$ (Equation 2) impact the performance of our model. Each variation is defined as following:

Eqn. 1: dot product. Dot product α is defined as

$$\alpha(\mathbf{h}, \mathbf{u}) = \mathbf{h}^\top \mathbf{u} \tag{6}$$

where $\top$ indicates matrix transpose. Dot product has been used for the measurement of similarity between two vectors by Hill et al. (2016).

Eqn. 1: linear. Linear α is defined as

$$\alpha(\mathbf{h}, \mathbf{u}) = \mathbf{w}_{\text{lin}}^\top [\mathbf{h}; \mathbf{u}] \tag{7}$$

where $\mathbf{w}_{\text{lin}}^\top \in \mathbb{R}^{4d}$ is a trainable weight matrix. This can be considered as the simplification of Equation 1 by dropping the term $\mathbf{h} \circ \mathbf{u}$ in the concatenation.

Eqn. 1: bilinear. Bilinear α is defined as

$$\alpha(\mathbf{h}, \mathbf{u}) = \mathbf{h}^\top \mathbf{W}_{\text{bi}} \mathbf{u} \tag{8}$$

where $\mathbf{W}_{\text{bi}} \in \mathbb{R}^{2d \times 2d}$ is a trainable weight matrix. Bilinear term has been used by Chen et al. (2016).

Eqn. 1: linear after MLP. We can also perform linear mapping after single layer of perceptron:

$$\alpha(\mathbf{h}, \mathbf{u}) = \mathbf{w}_{\text{lin}}^\top \tanh(\mathbf{W}_{\text{mlp}} [\mathbf{h}; \mathbf{u}] + \mathbf{b}_{\text{mlp}}) \tag{9}$$

where $\mathbf{W}_{\text{mlp}}$ and $\mathbf{b}_{\text{mlp}}$ are trainable weight matrix and bias, respectively. Linear mapping after perceptron layer has been used by Hermann et al. (2015).

Eqn. 2: MLP after concatenation. We can define $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta}(\mathbf{h}, \tilde{\mathbf{u}}, \tilde{\mathbf{h}}) = \max(0, \mathbf{W}_{\text{mlp}} [\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{h}}] + \mathbf{b}_{\text{mlp}}) \tag{10}$$

where $\mathbf{W}_{\text{mlp}} \in \mathbb{R}^{2d \times 8d}$ and $\mathbf{b}_{\text{mlp}} \in \mathbb{R}^{2d}$ are trainable weight matrix and bias. This is equivalent to adding ReLU after linearly transforming the original definition of $\boldsymbol{\beta}$. Since the output dimension of $\boldsymbol{\beta}$ changes, the input dimension of the first LSTM of the modeling layer will change as well.

The results of these variations on the dev data of SQuAD are shown in Table 5. It is important to note that there are non-trivial gaps between our definition of α and other definitions employed by previous work. Adding MLP in $\boldsymbol{\beta}$ does not seem to help, yielding slightly worse result than $\boldsymbol{\beta}$ without MLP.