BI-DIRECTIONAL ATTENTION FLOW FOR MACHINE COMPREHENSION

Guo Tianyi

2018.4.12

Outline

INTRODUCTION

- The tasks of machine comprehension (MC) and question answering (QA) have gained significant popularity.
- 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.

Bi-Directional Attention Flow

Bi-Directional Attention Flow (BIDAF)

BIDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation.

Improvements:

- The attention is computed for every time step and all the attended vectors are allowed to flow through to the subsequent modeling layer.
- Use a memory-less attention mechanism. The attention at each time step is a function of only the query and the context paragraph at the current time step.
- Use attention mechanisms in both directions, query-to-context and context-to-query, which provide complimentary information to each other.

Outline

Model Structure

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

1. Character Embedding Layer
2. Word Embedding Layer
3. Contextual Embedding Layer
4. Attention Flow Layer
5. Modeling Layer
6. Output Layer

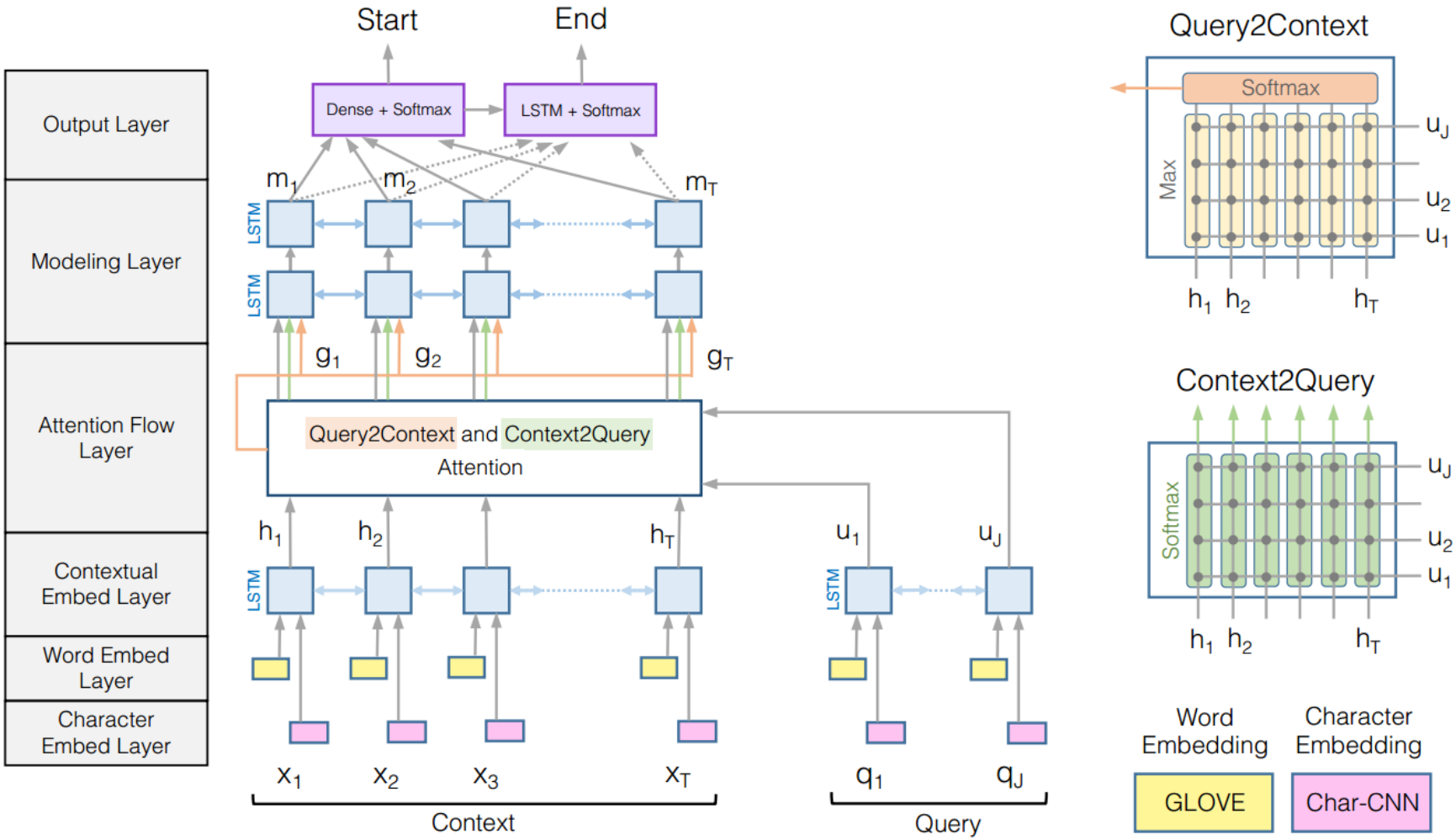

Figure: BiDirectional Attention Flow Model

Embedding layer

Character embedding layer is responsible for mapping each word to a high-dimensional vector space.
Following Kim (2014), we obtain the characterlevel embedding of each word using Convolutional Neural Networks (Char-CNN).

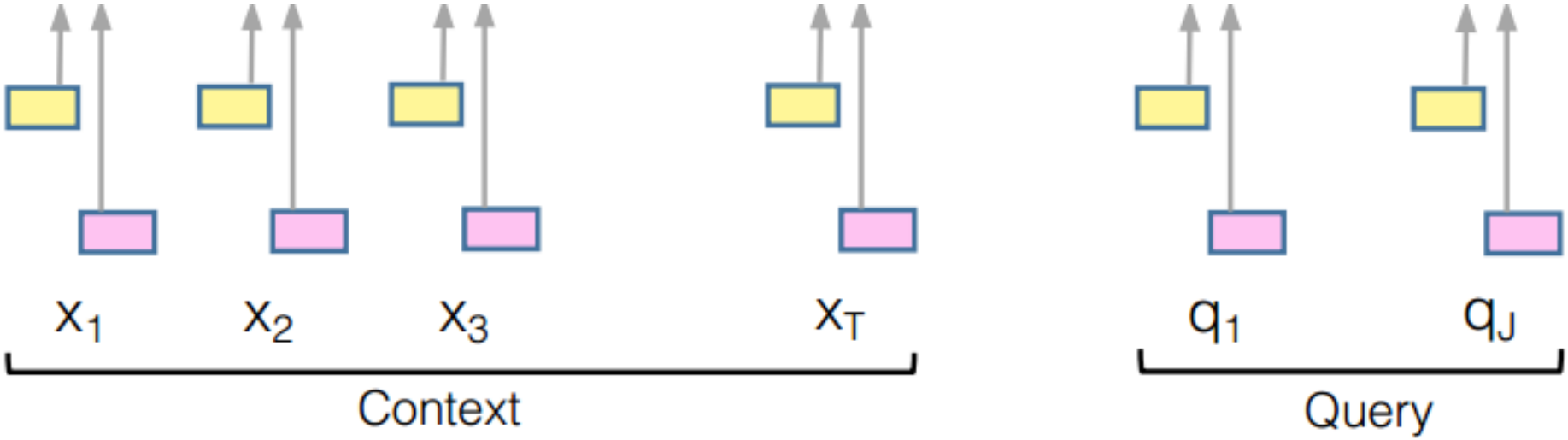

Figure: Character and word embedding layer

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 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.

Embedding Layer

Use a Long Short-Term Memory Network (LSTM) (Hochreiter & Schmidhuber, 1997) to model the temporal interactions between words. 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 $\mathbf{Q}$.

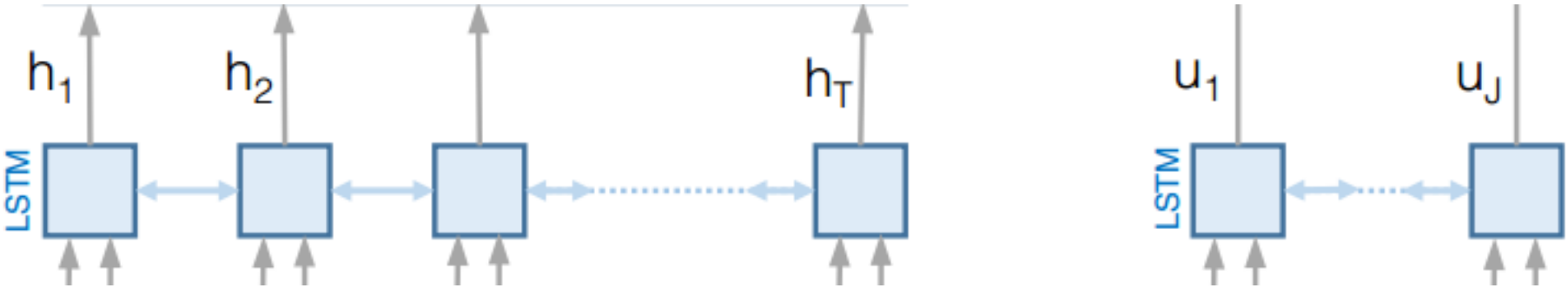

Figure: Contextual Embedding Layer

The first three layers of the model are computing features from the query and context at different levels of granularity.

Attention Flow Layer

Attention flow layer links and fuses information from the context and the query words.
The inputs to the layer are the context **H** and the query **U**. The outputs of the layer are the query-aware vector representations of the context words, **G**, along with **H**.

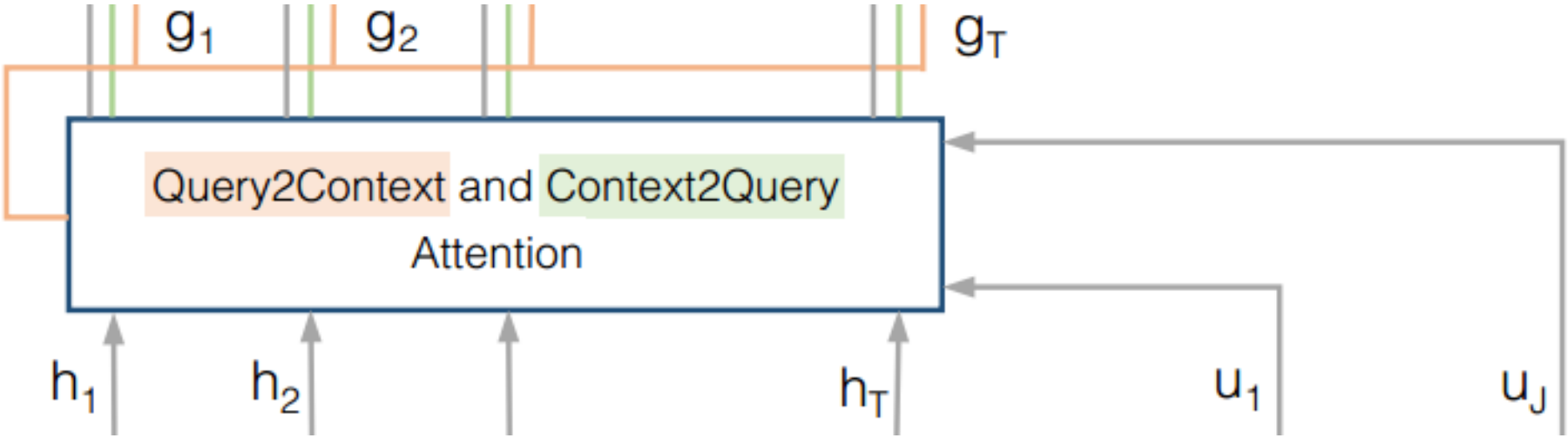

Figure: Attention Flow Layer

Attention Flow Layer

We compute attentions from context to query as well as from query to context based on 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.

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

where $\alpha(\mathbf{h}; \mathbf{u}) = \mathbf{w}_{(\mathbf{S})}^{\top}[\mathbf{h}; \mathbf{u}; \mathbf{h} \circ \mathbf{u}]$, $w_{(\mathbf{S})} \in \mathbb{R}^{6d}$ is a trainable weight vector, $\circ$ is elementwise multiplication.

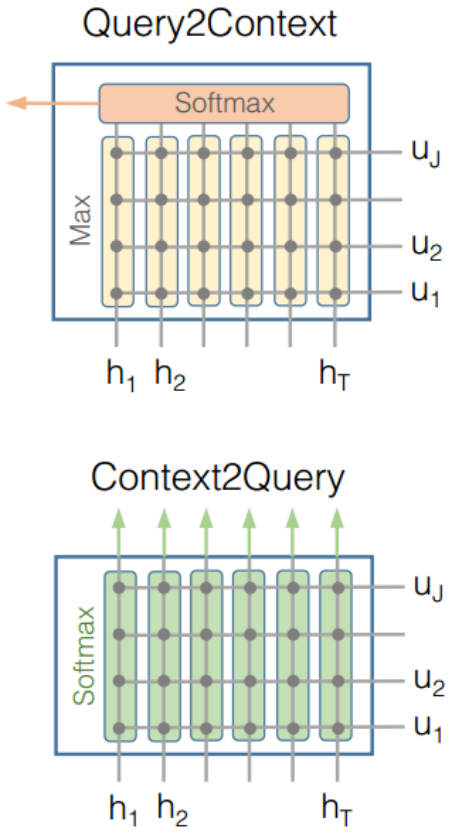

Figure: Attention

Attention Flow Layer

Context-to-query Attention.

Let $\mathbf{a}_t \in \mathbb{R}^J$ represent the attention weights on the query words by t-th context word. $\mathbf{a}_t = \text{softmax}(\mathbf{S}_{t:}) \in \mathbb{R}^J$, $\tilde{\mathbf{U}}_{:t} = \sum_j \mathbf{a}_{tj}\mathbf{U}_{:j}$.

Query-to-context Attention.

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 $\tilde{\mathbf{H}} \in \mathbb{R}^{2d \times T}$.

Attention Flow Layer

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

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

where $\beta(\mathbf{h}; \tilde{\mathbf{u}}; \tilde{\mathbf{h}}) = [\mathbf{h}; \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{u}}; \mathbf{h} \circ \tilde{\mathbf{h}}] \in \mathbb{R}^{8d \times T}$ (i.e., $d_G = 8d$)

Modeling Layer

We use two layers of bi-directional LSTM, with the output size of d for each direction. Hence we obtain a matrix $M \in \mathbb{R}^{2d \times T}$. Each column vector of M is expected to contain contextual information about the word with respect to the entire context paragraph and the query.

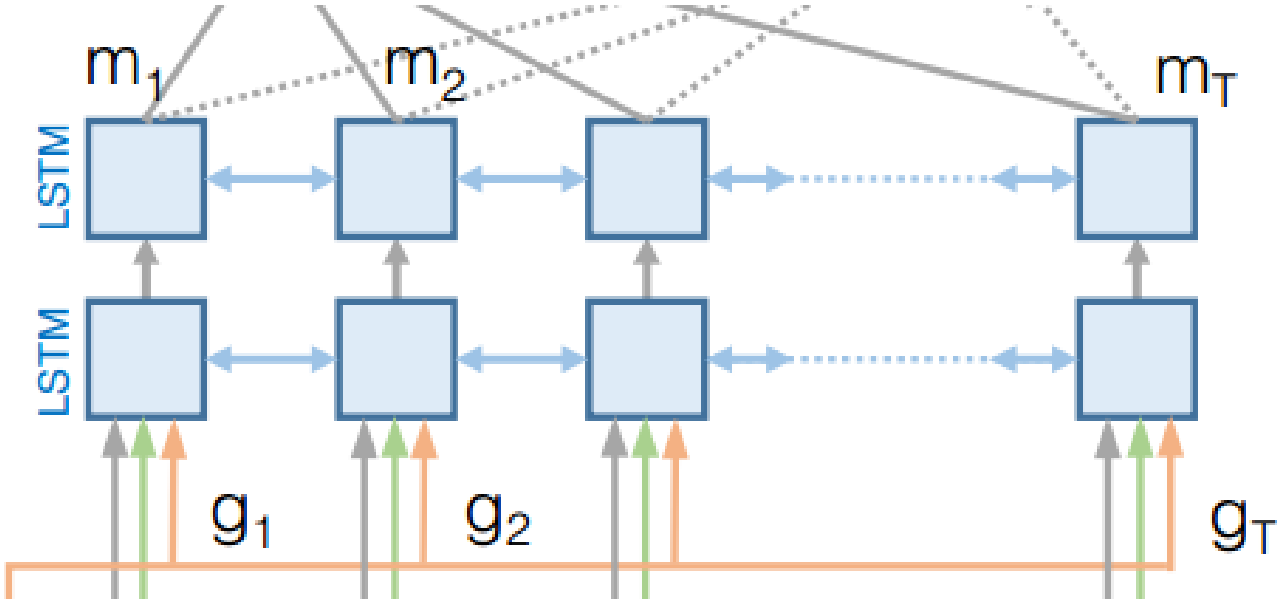

Figure: Modeling Layer

Output Layer

The output layer is application-specific.
The phrase is derived by predicting the start and the end indices of the phrase in the paragraph.

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

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

where $\mathbf{M}^2 = \text{BiLSTM}(\mathbf{M})$

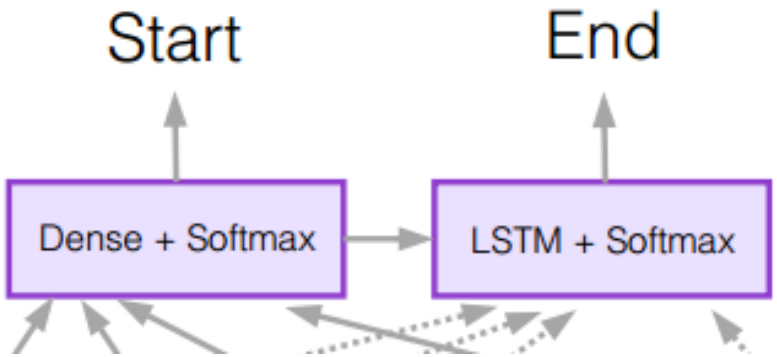

Figure: Output Layer

Training & Test

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:

$$L = -\frac{1}{N}\sum_{i}^{N} \log(\mathbf{p}^1_{y^1_i}) + \log(\mathbf{p}^2_{y^2_i}) \tag{5}$$

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

Outline

Machine comprehension

Massive cloze test datasets (CNN/DailyMail by Hermann et al. (2015) and Childrens Book Test by Hill et al. (2016)) . Rajpurkar et al. (2016) released the Stanford Question Answering (SQuAD) dataset with over 100,000 questions.
Previous works in end-to-end machine comprehension use attention mechanisms in three distinct ways:

1. The first group (largely inspired by Bahdanau et al. (2015)) uses a dynamic attention mechanism.
2. 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)).
3. 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.

Visual question answering

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).

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).

Outline

Result

SQuAD is a machine comprehension dataset on a large set of Wikipedia articles, with more than 100,000 questions.
The results of our model and competing approaches on the hidden test are summarized in Table below. BIDAF (ensemble) achieves an EM score of 73.3 and an F1 score of 81.1, outperforming all previous approaches.

	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**

Table: Results on the SQuAD test set

Ablations

	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

Table: Ablations on the SQuAD dev set

Visualizations

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: 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.

Visualizations

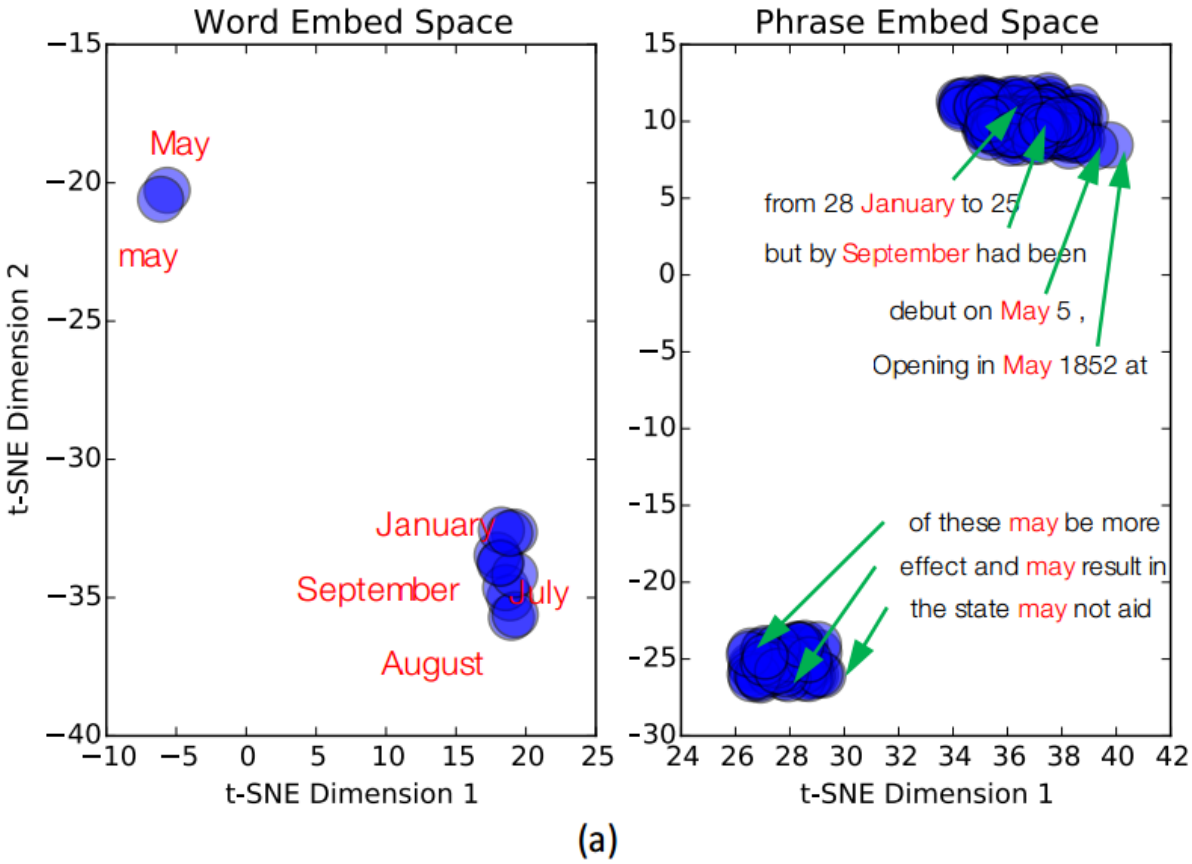

Figure: t-SNE visualizations of the *months* names embedded in the two feature spaces.

Visualizations

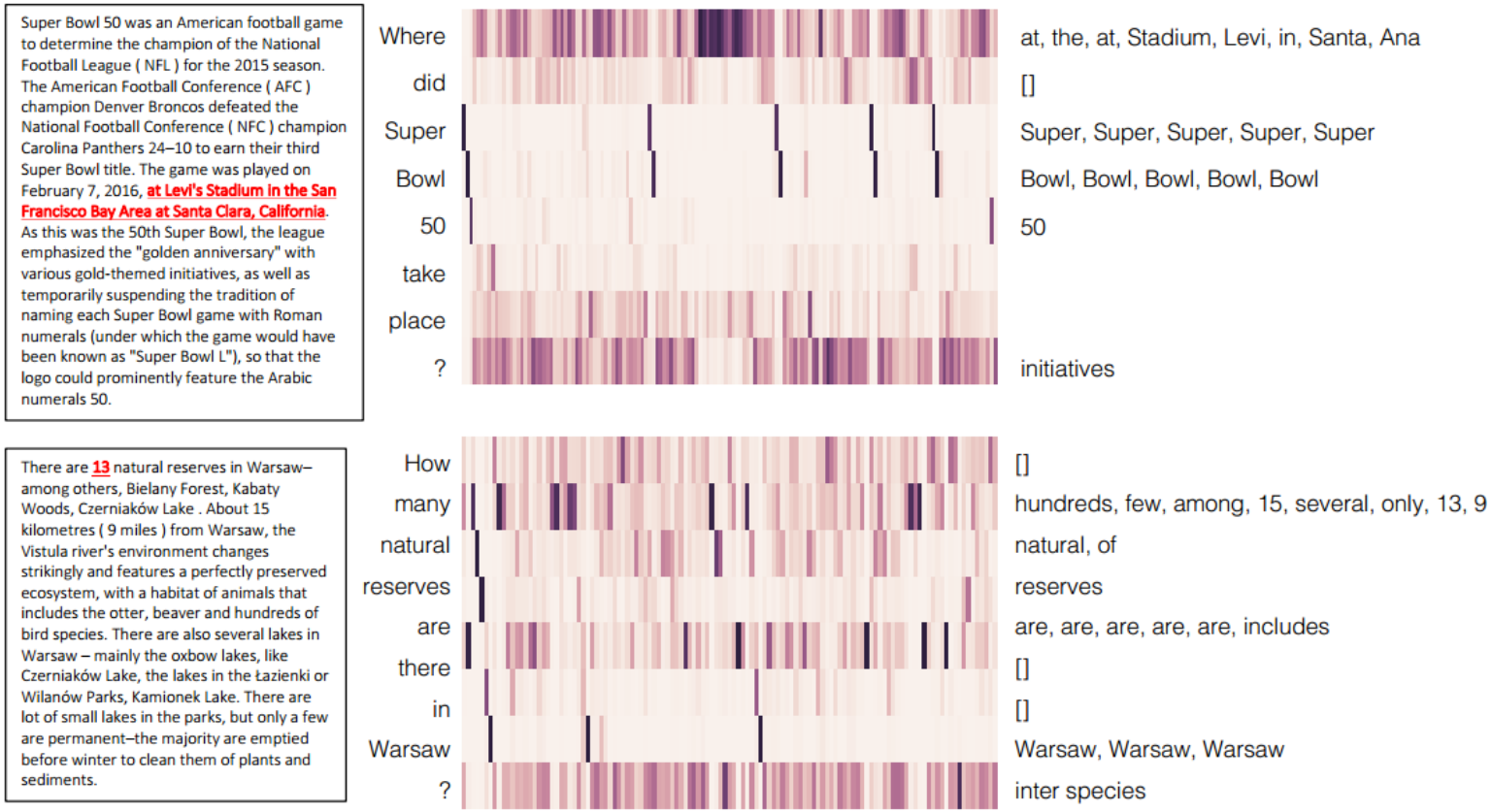

Figure: Attention matrices for question-context tuples.

Discussion

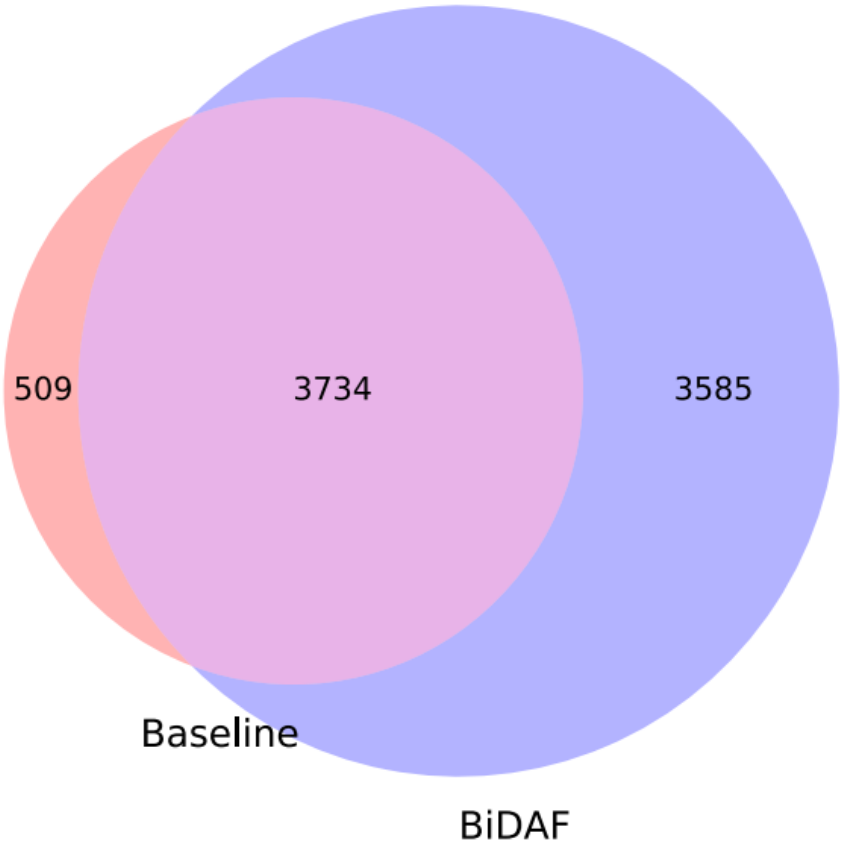

Figure: Questions answered correctly by our BiDAF model and the more traditional baseline model

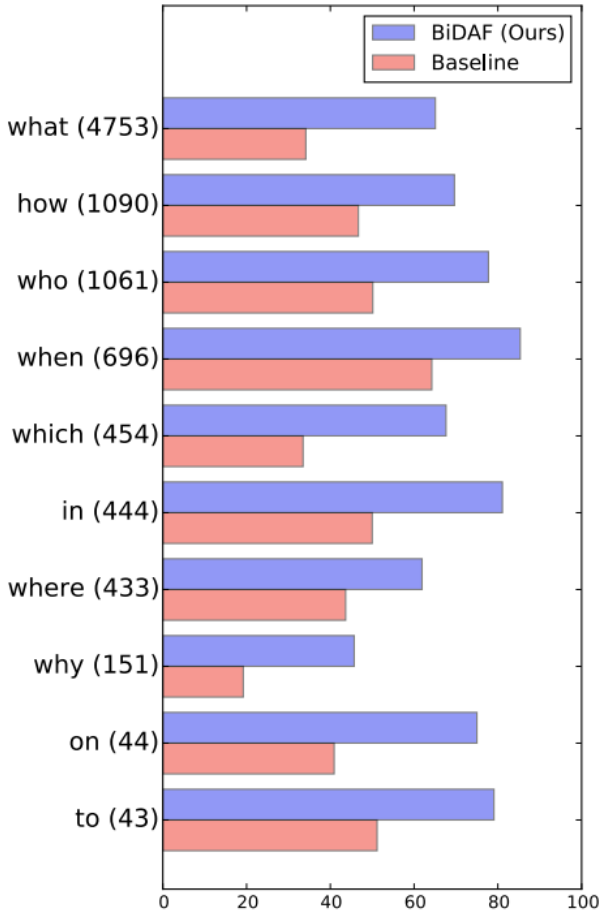

Figure: % of questions with correct answers

Outline

Dataset

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.

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. 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.
Another important difference from SQuAD is that the answer entity might appear more than once in the context paragraph. 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.

Results

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

Outline