Machine Comprehension Using Match-LSTM and Answer Pointer

Shuohang Wang
School of Information Systems
Singapore Management University
shwang.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

Machine comprehension of text is an important problem in natural language processing. A recently released dataset, the Stanford Question Answering Dataset (SQuAD), offers a large number of real questions and their answers created by humans through crowdsourcing. SQuAD provides a challenging testbed for evaluating machine comprehension algorithms, partly because compared with previous datasets, in SQuAD the answers do not come from a small set of candidate answers and they have variable lengths. We propose an end-to-end neural architecture for the task. The architecture is based on match-LSTM, a model we proposed previously for textual entailment, and Pointer Net, a sequence-to-sequence model proposed by Vinyals et al. (2015) to constrain the output tokens to be from the input sequences. We propose two ways of using Pointer Net for our task. Our experiments show that both of our two models substantially outperform the best results obtained by Rajpurkar et al. (2016) using logistic regression and manually crafted features. Besides, our boundary model outperforms the baseline of "Golden Passage" on the MSMARCO data set (Nguyen et al., 2016).

1 Introduction

Machine comprehension of text is one of the ultimate goals of natural language processing. While the ability of a machine to understand text can be assessed in many different ways, in recent years, several benchmark datasets have been created to focus on answering questions as a way to evaluate machine comprehension (Richardson et al., 2013; Hermann et al., 2015; Hill et al., 2016; Weston et al., 2016; Rajpurkar et al., 2016; Nguyen et al., 2016). In this setup, typically the machine is first presented with a piece of text such as a news article or a story. The machine is then expected to answer one or multiple questions related to the text.

In most of the benchmark datasets, a question can be treated as a multiple choice question, whose correct answer is to be chosen from a set of provided candidate answers (Richardson et al., 2013; Hill et al., 2016). Presumably, questions with more given candidate answers are more challenging. The Stanford Question Answering Dataset (SQuAD) introduced recently by Rajpurkar et al. (2016) contains such more challenging questions whose correct answers can be any sequence of tokens from the given text. Moreover, unlike some other datasets whose questions and answers were created automatically in Cloze style (Hermann et al., 2015; Hill et al., 2016), the questions and answers in SQuAD were created by humans through crowdsourcing, which makes the dataset more realistic. Another real dataset, the Human-Generated MAchine Reading COmprehension dataset (MSMARCO) (Nguyen et al., 2016), provided a query together with several related documents collected from Bing Index. The answer to the query is generated by human and the answer words can not only come from the paragraphs.

Given these advantages of the SQuAD and MSMARCO dataset, in this paper, we focus on these new datasets to study machine comprehension of text. Traditional solutions to this kind of question answering tasks rely on NLP pipelines that involve multiple steps of linguistic analyses and feature engineering, including syntactic parsing, named entity recognition, question classification, semantic parsing, etc. Recently, with the advances of applying neural network models in NLP, there has been much interest in building end-to-end neural architectures for various NLP tasks, including several

pieces of work on machine comprehension (Hermann et al., 2015; Hill et al., 2016; Yin et al., 2016; Kadlec et al., 2016; Cui et al., 2016). However, given the properties of previous machine comprehension datasets, existing end-to-end neural architectures for the task either rely on the candidate answers (Hill et al., 2016; Yin et al., 2016) or assume that the answer is a single token (Hermann et al., 2015; Kadlec et al., 2016; Cui et al., 2016), which make these methods unsuitable for the SQuAD/MSMARCO dataset. In this paper, we propose a new end-to-end neural architecture to address the machine comprehension problem as defined in the SQuAD/MSMARCO dataset. And for the MSMARCO dataset, we will only make use of the words in the paragraphs to generate the answer.

Specifically, observing that in the SQuAD/MSMARCO dataset many questions could be entailed from some sentences in the original text, we adopt a match-LSTM model that we developed earlier for textual entailment (Wang & Jiang, 2016) as one layer of our model. We build a bi-directional match-LSTM on the given paragraph with attentions on the question for each word so that each position in the paragraph will have a hidden representation reflecting its relation to the question. Then we further adopt the Pointer Net (Ptr-Net) model developed by Vinyals et al. (2015) to select the words in these positions based on the hidden representations built by match-LSTM as an answer. We propose two ways to apply the Ptr-Net model for our task: a **sequence model** which selects the answer word by word, and a **boundary model** which only selects the start and end points of the answer span. Experiments on the SQuAD dataset show that our two models both outperform the best performance reported by Rajpurkar et al. (2016). Moreover, using an ensemble of several of our models, we can achieve very competitive performance on SQuAD. For the MSMARCO dataset, a real query based problem, our boundary model outperforms our sequence model with a big margin. It also outperforms the golden passage baseline.

Our contributions can be summarized as follows: (1) We propose two new end-to-end neural network models for machine comprehension, which combine match-LSTM and Ptr-Net to handle the special properties of the SQuAD dataset. In the models related to machine comprehension, to the best of our knowledge, we are the first to concatenate the representations of each position in the paragraph with the weighted sum of question representations. And we are the first to propose the boundary model which is more suitable to the SQuAD/MSMARCO tasks. (2) We have achieved the performance of an exact match score of 71.3% and an F1 score of 80.8% on the unseen test dataset, which is much better than the feature-engineered solution (Rajpurkar et al., 2016). Our performance is also close to the state of the art on SQuAD, which is 74.8% in terms of exact match and 82.2% in terms of F1 collected from the SQuAD Leaderboard[1]. Our boundary model achieves the state-of-art performance on the MSMARCO dataset with BLUE-1/2/3/4 40.7/33.9/30.6/28.7 and Rouge-l 37.3. (3) Our further visualization of the models reveals some useful insights of the attention mechanism for reasoning the questions. And we also show that the boundary model can overcome the early stop prediction problem in the sequence model. Besides, we also made our code available online [2].

2 METHOD

In this section, we first briefly review match-LSTM and Pointer Net. These two pieces of existing work lay the foundation of our method. We then present our end-to-end neural architecture for machine comprehension.

2.1 MATCH-LSTM

In a recent work on learning natural language inference, we proposed a match-LSTM model for predicting textual entailment (Wang & Jiang, 2016). In textual entailment, two sentences are given where one is a premise and the other is a hypothesis. To predict whether the premise entails the hypothesis, the match-LSTM model goes through the tokens of the hypothesis sequentially. At each position of the hypothesis, attention mechanism is used to obtain a weighted vector representation of the premise. This weighted premise is then to be combined with a vector representation of the current token of the hypothesis and fed into an LSTM, which we call the match-LSTM. The match-

---

[1] https://rajpurkar.github.io/SQuAD-explorer/
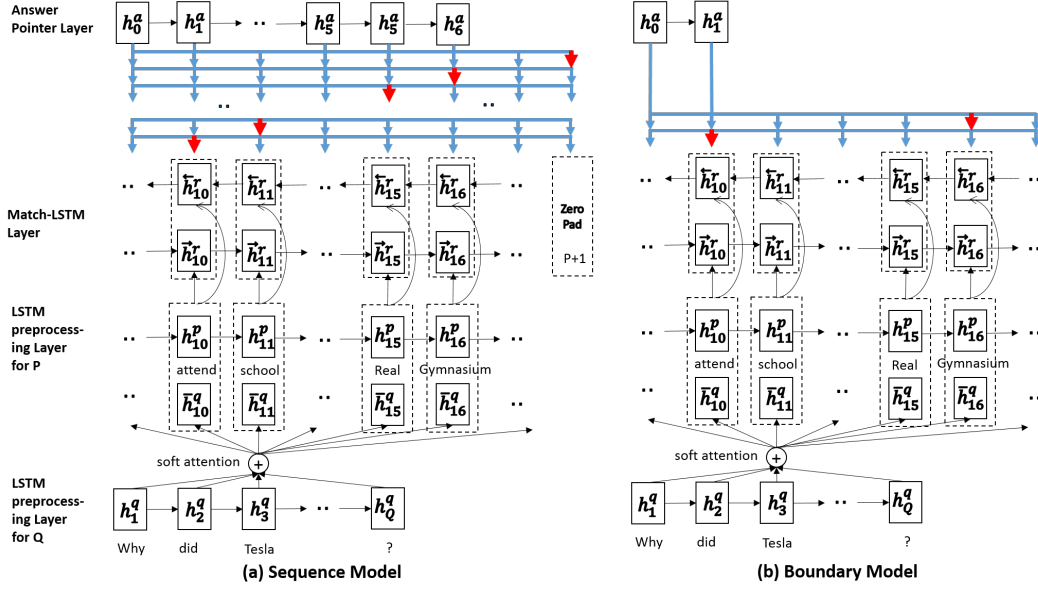[2] https://github.com/shuohangwang/SeqMatchSeq

Figure 1: An overview of our two models. Both models consist of an LSTM preprocessing layer, a match-LSTM layer and an Answer Pointer layer. For each match-LSTM in a particular direction, $\bar{h}_i^q$, which is defined as $\mathbf{H}^q \alpha_i^\mathsf{T}$, is computed using the α in the corresponding direction, as described in Eqn. (2)

LSTM essentially sequentially aggregates the matching of the attention-weighted premise to each token of the hypothesis and uses the aggregated matching result to make a final prediction.

2.2 Pointer Net

Vinyals et al. (2015) proposed a Pointer Network (Ptr-Net) model to solve a special kind of problems where we want to generate an output sequence whose tokens must come from the input sequence. Instead of picking an output token from a fixed vocabulary, Ptr-Net uses attention mechanism as a pointer to select a position from the input sequence as an output symbol. The pointer mechanism has inspired some recent work on language processing (Gu et al., 2016; Kadlec et al., 2016). Here we adopt Ptr-Net in order to construct answers using tokens from the input text.

2.3 Our Method

Formally, the problem we are trying to solve can be formulated as follows. We are given a piece of text, which we refer to as a passage, and a question related to the passage. The passage is represented by matrix $\mathbf{P} \in \mathbb{R}^{d \times P}$, where P is the length (number of tokens) of the passage and d is the dimensionality of word embeddings. Similarly, the question is represented by matrix $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ where Q is the length of the question. Our goal is to identify a subsequence from the passage as the answer to the question.

As pointed out earlier, since the output tokens are from the input, we would like to adopt the Pointer Net for this problem. A straightforward way of applying Ptr-Net here is to treat an answer as a sequence of tokens from the input passage but ignore the fact that these tokens are consecutive in the original passage, because Ptr-Net does not make the consecutivity assumption. Specifically, we represent the answer as a sequence of integers $\mathbf{a} = (a_1, a_2, \ldots)$, where each a_i is an integer between 1 and P, indicating a certain position in the passage.

Alternatively, if we want to ensure consecutivity, that is, if we want to ensure that we indeed select a subsequence from the passage as an answer, we can use the Ptr-Net to predict only the start and the end of an answer. In this case, the Ptr-Net only needs to select two tokens from the input passage, and all the tokens between these two tokens in the passage are treated as the answer. Specifically,

we can represent the answer to be predicted as two integers $\mathbf{a} = (a_s, a_e)$, where a_s an a_e are integers between 1 and P.

We refer to the first setting above as a *sequence* model and the second setting above as a *boundary* model. For either model, we assume that a set of training examples in the form of triplets $\{(\mathbf{P}_n, \mathbf{Q}_n, \mathbf{a}_n)\}_{n=1}^N$ are given.

An overview of the two neural network models are shown in Figure 1. Both models consist of three layers: (1) An LSTM preprocessing layer that preprocesses the passage and the question using LSTMs. (2) A match-LSTM layer that tries to match the passage against the question. (3) An Answer Pointer (Ans-Ptr) layer that uses Ptr-Net to select a set of tokens from the passage as the answer. The difference between the two models only lies in the third layer.

LSTM Preprocessing Layer

The purpose for the LSTM preprocessing layer is to incorporate contextual information into the representation of each token in the passage and the question. We use a standard one-directional LSTM (Hochreiter & Schmidhuber, 1997) to process the passage and the question separately, as shown below:

$$\mathbf{H}^p = \overrightarrow{LSTM}(\mathbf{P}), \quad \mathbf{H}^q = \overrightarrow{LSTM}(\mathbf{Q}). \tag{1}$$

The resulting matrices $\mathbf{H}^p \in \mathbb{R}^{l \times P}$ and $\mathbf{H}^q \in \mathbb{R}^{l \times Q}$ are hidden representations of the passage and the question, where l is the dimensionality of the hidden vectors. In other words, the ith column vector $\mathbf{h}_i^p$ (or $\mathbf{h}_i^q$) in $\mathbf{H}^p$ (or $\mathbf{H}^q$) represents the ith token in the passage (or the question) together with some contextual information from the left.

Match-LSTM Layer

We apply the match-LSTM model (Wang & Jiang, 2016) proposed for textual entailment to our machine comprehension problem by treating the question as a premise and the passage as a hypothesis. The match-LSTM sequentially goes through the passage. At position i of the passage, it first uses the standard word-by-word attention mechanism to obtain attention weight vector $\overrightarrow{\alpha}_i \in \mathbb{R}^{1 \times Q}$ as follows:

$$\begin{aligned} \overrightarrow{\mathbf{G}}_i &= \tanh(\mathbf{W}^q \mathbf{H}^q + (\mathbf{W}^p \mathbf{h}_i^p + \mathbf{W}^r \overrightarrow{\mathbf{h}}_{i-1}^r + \mathbf{b}^p) \otimes \mathbf{e}_Q), \\ \overrightarrow{\alpha}_i &= \text{softmax}(\mathbf{w}^\mathsf{T} \overrightarrow{\mathbf{G}}_i + b \otimes \mathbf{e}_Q), \end{aligned} \tag{2}$$

where $\mathbf{W}^q, \mathbf{W}^p, \mathbf{W}^r \in \mathbb{R}^{l \times l}$, $\mathbf{b}^p, \mathbf{w} \in \mathbb{R}^{l \times 1}$ and $b \in \mathbb{R}$ are parameters to be learned, $\overrightarrow{\mathbf{G}}_i \in \mathbb{R}^{l \times Q}$ is the intermediate result, $\overrightarrow{\mathbf{h}}_{i-1}^r \in \mathbb{R}^{l \times 1}$ is the hidden vector of the one-directional match-LSTM (to be explained below) at position $i-1$, and the outer product $(\cdot \otimes \mathbf{e}_Q)$ produces a matrix or row vector by repeating the vector or scalar on the left for Q times.

Essentially, the resulting attention weight $\overrightarrow{\alpha}_{i,j}$ above indicates the degree of matching between the ith token in the passage with the jth token in the question. Next, we use the attention weight vector $\overrightarrow{\alpha}_i$ to obtain a weighted version of the question and combine it with the current token of the passage to form a vector $\overrightarrow{\mathbf{z}}_i$:

$$\overrightarrow{\mathbf{z}}_i = \begin{bmatrix} \mathbf{h}_i^p \\ \mathbf{H}^q \overrightarrow{\alpha}_i^\mathsf{T} \end{bmatrix}, \tag{3}$$

where $\mathbf{H}^q \in \mathbb{R}^{l \times Q}$, $\overrightarrow{\alpha}_i \in \mathbb{R}^{1 \times Q}$ and $\mathbf{h}_i^p \in \mathbb{R}^{l \times 1}$. This vector $\overrightarrow{\mathbf{z}}_i$ is fed into a standard one-directional LSTM to form our so-called match-LSTM:

$$\overrightarrow{\mathbf{h}}_i^r = \overrightarrow{LSTM}(\overrightarrow{\mathbf{z}}_i, \overrightarrow{\mathbf{h}}_{i-1}^r), \tag{4}$$

where $\overrightarrow{\mathbf{h}}_i^r \in \mathbb{R}^{l \times 1}$.

We further build a similar match-LSTM in the reverse direction. The purpose is to obtain a representation that encodes the contexts from both directions for each token in the passage.

Let $\overrightarrow{\mathbf{H}}^r \in \mathbb{R}^{l \times P}$ represent the hidden states $[\overrightarrow{\mathbf{h}}_1^r, \overrightarrow{\mathbf{h}}_2^r, \ldots, \overrightarrow{\mathbf{h}}_P^r]$ and $\overleftarrow{\mathbf{H}}^r \in \mathbb{R}^{l \times P}$ represent $[\overleftarrow{\mathbf{h}}_1^r, \overleftarrow{\mathbf{h}}_2^r, \ldots, \overleftarrow{\mathbf{h}}_P^r]$, the hidden states of match-LSTM in the reverse direction. We define $\mathbf{H}^r \in$

$\mathbb{R}^{2l \times P}$ as the concatenation of the two:

$$\mathbf{H}^{\mathrm{r}} = \begin{bmatrix} \overrightarrow{\mathbf{H}}^{\mathrm{r}} \\ \overleftarrow{\mathbf{H}}^{\mathrm{r}} \end{bmatrix}. \tag{5}$$

Answer Pointer Layer

The top layer, the Answer Pointer (Ans-Ptr) layer, is motivated by the Pointer Net introduced by Vinyals et al. (2015). This layer uses the sequence $\mathbf{H}^{\mathrm{r}}$ as input. Recall that we have two different models: The *sequence* model produces a sequence of answer tokens but these tokens may not be consecutive in the original passage. The *boundary* model produces only the start token and the end token of the answer, and then all the tokens between these two in the original passage are considered to be the answer. We now explain the two models separately.

The Sequence Model: Recall that in the sequence model, the answer is represented by a sequence of integers $\mathbf{a} = (a_1, a_2, \ldots)$ indicating the positions of the selected tokens in the original passage. The Ans-Ptr layer models the generation of these integers in a sequential manner. Because the length of an answer is not fixed, in order to stop generating answer tokens at certain point, we allow each a_k to take up an integer value between 1 and $P + 1$, where $P + 1$ is a special value indicating the end of the answer. Once a_k is set to be $P + 1$, the generation of the answer stops.

In order to generate the kth answer token indicated by a_k, first, the attention mechanism is used again to obtain an attention weight vector $\beta_k \in \mathbb{R}^{1 \times (P+1)}$, where $\beta_{k,j}$ ($1 \leq j \leq P + 1$) is the probability of selecting the jth token from the passage as the kth token in the answer, and $\beta_{k,(P+1)}$ is the probability of stopping the answer generation at position k. β_k is modeled as follows:

$$\mathbf{F}_k = \tanh(\mathbf{V}\widetilde{\mathbf{H}}^{\mathrm{r}} + (\mathbf{W}^{\mathrm{a}}\mathbf{h}^{\mathrm{a}}_{k-1} + \mathbf{b}^a) \otimes \mathbf{e}_{(P+1)}), \tag{6}$$

$$\beta_k = \mathrm{softmax}(\mathbf{v}^{\mathsf{T}}\mathbf{F}_k + c \otimes \mathbf{e}_{(P+1)}), \tag{7}$$

where $\widetilde{\mathbf{H}}^{\mathrm{r}} \in \mathbb{R}^{2l \times (P+1)}$ is the concatenation of $\mathbf{H}^{\mathrm{r}}$ with a zero vector, defined as $\widetilde{\mathbf{H}}^{\mathrm{r}} = [\mathbf{H}^{\mathrm{r}}; \mathbf{0}]$, $\mathbf{V} \in \mathbb{R}^{l \times 2l}, \mathbf{W}^{\mathrm{a}} \in \mathbb{R}^{l \times l}, \mathbf{b}^a, \mathbf{v} \in \mathbb{R}^{l \times 1}$ and $c \in \mathbb{R}$ are parameters to be learned, $\mathbf{F}_k \in \mathbb{R}^{l \times (P+1)}$ is the intermediate result, $(\cdot \otimes \mathbf{e}_{(P+1)})$ follows the same definition as before, and $\mathbf{h}^{\mathrm{a}}_{k-1} \in \mathbb{R}^{l \times 1}$ is the hidden vector at position $k - 1$ of an answer LSTM as defined below:

$$\mathbf{h}^{\mathrm{a}}_k = \overrightarrow{LSTM}(\widetilde{\mathbf{H}}^{\mathrm{r}}\beta_k^{\mathsf{T}}, \mathbf{h}^{\mathrm{a}}_{k-1}). \tag{8}$$

We can then model the probability of generating the answer sequence as

$$p(\mathbf{a}|\mathbf{H}^{\mathrm{r}}) = \prod_k p(a_k|a_1, a_2, \ldots, a_{k-1}, \mathbf{H}^{\mathrm{r}}), \tag{9}$$

and

$$p(a_k = j|a_1, a_2, \ldots, a_{k-1}, \mathbf{H}^{\mathrm{r}}) = \beta_{k,j}. \tag{10}$$

To train the model, we minimize the following loss function based on the training examples:

$$-\sum_{n=1}^{N} \log p(\mathbf{a}_n|\mathbf{P}_n, \mathbf{Q}_n). \tag{11}$$

The Boundary Model: The boundary model works in a way very similar to the sequence model above, except that instead of predicting a sequence of indices $a_1, a_2, \ldots$, we only need to predict two indices a_{s} and a_{e}. So the main difference from the sequence model above is that in the boundary model we do not need to add the zero padding to $\mathbf{H}^{\mathrm{r}}$, and the probability of generating an answer is simply modeled as

$$p(\mathbf{a}|\mathbf{H}^{\mathrm{r}}) = p(a_{\mathrm{s}}|\mathbf{H}^{\mathrm{r}})p(a_{\mathrm{e}}|a_{\mathrm{s}}, \mathbf{H}^{\mathrm{r}}). \tag{12}$$

As this boundary model could point to a span covering too many tokens without any restriction, we try to manually limit the length of the predicted span and then search the span with the highest probability computed by $p(\mathbf{a}_s) \times p(\mathbf{a}_e|\mathbf{a}_s)$ as the answer. Besides, as the boundary has a sequence of fixed number of values, **bi-directional Ans-Ptr** can be simply combined to fine-tune the correct span.

3 EXPERIMENTS

In this section, we present our experiment results and perform some analyses to better understand how our models works.

3.1 DATA

We use the Stanford Question Answering Dataset (SQuAD) v1.1 and MSMARCO: Human-Generated MAchine Reading COmprehension dataset v1.1 to conduct our experiments.

Passages in SQuAD come from 536 articles from Wikipedia covering a wide range of topics. Each passage is a single paragraph from a Wikipedia article, and each passage has around 5 questions associated with it. In total, there are 23,215 passages and 107,785 questions. The data has been split into a training set (with 87,599 question-answer pairs), a development set (with 10,570 question-answer pairs) and a hidden test set.

For the MSMARCO dataset, the questions are user queries issued to the Bing search engine, the context passages are real Web documents and the answers are human generated. We select the span that has the highest F1 score with the golden answer for training and only predict the span in the passages during the evaluation. The data has been split into a training set (82326 pairs), a development set (10047 pairs) and a test set (9650 pairs).

3.2 EXPERIMENT SETTINGS

We first tokenize all the passages, questions and answers. We use word embeddings from GloVe (Pennington et al., 2014) to initialize the model. Words not found in GloVe are initialized as zero vectors. The word embeddings are not updated during the training of the model.

The dimensionality l of the hidden layers is set to be 150 or 300. We use ADAMAX (Kingma & Ba, 2015) with the coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimize the model. Each update is computed through a minibatch of 30 instances. We do not use L2-regularization.

For the SQuAD data set, the performance is measured by two metrics: percentage of exact match with the ground truth answers, and word-level F1 score when comparing the tokens in the predicted answers with the tokens in the ground truth answers. Note that in the development set and the test set each question has around three ground truth answers. F1 scores with the best matching answers are used to compute the average F1 score. For the MSMARCO dataset, the metrics in the official tool of MSMARCO evaluation are BLEU-1/2/3/4 and Rouge-l, which is widely used in many domains.

3.3 RESULTS

The SQuAD results of our models as well as the results of the baselines given by Rajpurkar et al. (2016) are shown in Table 1. We also show the DCR model (Yu et al., 2016) here, while our models were released two months earlier than theirs. For the "LSTM with Ans-Ptr" models, they are for the ablation experiment of attention mechanism in match-LSTM. Specifically we use the final representation of the question to take place of the weighted sum of the question representations. The MSMARCO results of our models are shown in Table 2. As there are around 10 passages as context for each question, the "Golden Passage" baseline is the human labeled passage which could answer the question.

From the results on SQuAD data set, we can see that both of our two models have clearly outperformed the logistic regression model by Rajpurkar et al. (2016), which relies on carefully designed features. Furthermore, our boundary model has outperformed the sequence model, achieving an exact match score of 63.0% and an F1 score of 72.7%. In particular, in terms of the exact match score, the boundary model has a clear advantage over the sequence model. The improvement of our models over the logistic regression model shows that our end-to-end neural network models without much feature engineering are very effective on this task and this dataset. Our boundary model also outperformed the DCR model (Yu et al., 2016), which maximizes the probability of the golden span from all the candidate spans through a neural network structure. Considering the effectiveness of boundary model, we further explore this model. We tried to add bi-directional pre-processing LSTM, add

	$|\theta|$	Exact Match		F1	
		Dev	Test	Dev	Test
Random Guess	0	1.1	1.3	4.1	4.3
Logistic Regression (Rajpurkar et al., 2016)	-	40.0	40.4	51.0	51.0
DCR (Yu et al., 2016)	-	62.5	62.5	71.2	71.0
Human	-	80.3	77.0	90.5	86.8
LSTM with Ans-Ptr (Sequence+b)	994K	37.7	-	48.5	-
LSTM with Ans-Ptr (Boundary+b)	994K	45.2	-	55.3	-
Match-LSTM with Ans-Ptr (Sequence)	882K	54.4	-	68.2	-
Match-LSTM with Ans-Ptr (Boundary)	882K	63.0	-	72.7	-
Match-LSTM with Ans-Ptr (Boundary+b)	1.1M	63.4	-	73.0	-
Match-LSTM with Bi-Ans-Ptr (Boundary+b)	1.4M	64.3	-	74.2	-
Match-LSTM with Bi-Ans-Ptr (Boundary+b+ns)	2.3M	67.0	-	75.9	-
Match-LSTM with Bi-Ans-Ptr (Boundary+b+ns+deep)	3.3M	**67.5**	-	76.6	-
Match-LSTM with Bi-Ans-Ptr (Boundary+b+ns+deep+elem)	3.7M	67.0	**66.9**	**77.2**	**77.1**
Match-LSTM with Ans-Ptr (Boundary+en)	882K	67.6	67.9	76.8	77.0
Match-LSTM with Bi-Ans-Ptr (Boundary+b+ns+deep+elem+en)	3.7M	**71.3**	**72.6**	**80.0**	**80.8**

Table 1: Experiment Results on SQuAD data set. Here "LSTM with Ans-Ptr" ablaze the attention mechanism in Match-LSTM. "b" refers to using bi-directional pre-processing LSTM , "ns" refers to not sharing the parameters between two directions of Match-LSTMs & LSTMs, "deep" refers to adding another two-layer bi-directional LSTM on the representations of Match-LSTMs, "elem" refers to concatenating element-wise comparison $(\mathbf{h}_i^p - \mathbf{H}^q \alpha_i^\intercal)$ and $(\mathbf{h}_i^p \odot \mathbf{H}^q \alpha_i^\intercal)$ into Eqn 3 and "en" refers to ensemble method.

			Dev/Test		
	bleu_1	bleu_2	bleu_3	bleu_4	rouge_l
Golden Passage	19.6/-	18.8/-	18.1/-	17.5/-	32.3/-
Match-LSTM with Ans-Ptr (Sequence)	12.5/-	9.2/-	7.5/-	6.5/-	22.5/-
Match-LSTM with Ans-Ptr (Boundary)	29.1/-	22.8/-	19.9/-	18.2/-	31.4/-
Match-LSTM with Bi-Ans-Ptr (Boundary)	**40.1/40.7**	**33.3/33.9**	**30.1/30.6**	**28.2/28.7**	**37.2/37.3**

Table 2: Experiment Results on MSMARCO data set. "Match-LSTM with Bi-Ans-Ptr (Boundary)" refers to "Match-LSTM with Bi-Ans-Ptr (Boundary+b+ns+deep+elem)" here.

bi-directional Ans-Ptr, add more LSTM layers on the top of match-LSTM layer, and add element-wise comparisons between passage and question states, $(\mathbf{h}_i^p - \mathbf{H}^q \alpha_i^\intercal)$ and $(\mathbf{h}_i^p \odot \mathbf{H}^q \alpha_i^\intercal)$, into Eqn 3. Finally, we explore the ensemble method by simply computing the product of the boundary probabilities collected from independently trained boundary models and then select the most likely span with no more than 15 tokens. This ensemble method achieved the best performance as shown in the table.

From the results on MSMARCO data set, we can see that the boundary model significantly outperforms the sequence model in both BLEU and ROUGE-l scores. The main reason is the length of the answers in this data set is much longer, with 16 words in average. And the sequence model is more likely to stop prediction earlier. The average length of the predicted answers by sequence model is 7, while the boundary model is 21 [3]. Moreover, our boundary model outperforms the Gold Passage baseline in all the metrics. As not all the words in the Golden passage are the answer, that gives the space for us to improve.

3.4   FURTHER ANALYSES

To better understand the strengths and weaknesses of our models, we perform some further analyses of the results below.

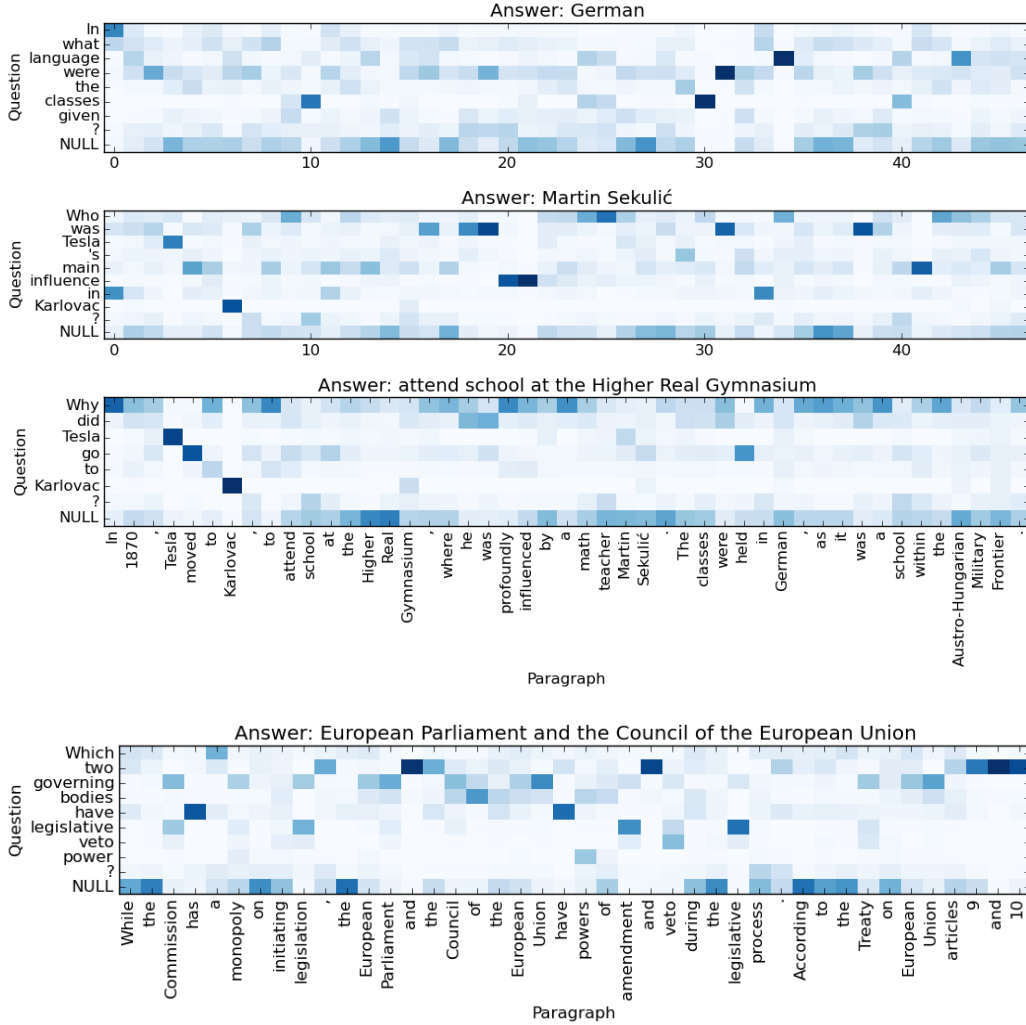[3]Two cases of the comparison between sequence and boundary models are shown in Appendix B

Figure 2: Visualization of the attention weights α for four questions. The first three questions share the same paragraph. The title is the answer predicted by our model.

First, we suspect that longer answers are harder to predict. To verify this hypothesis, we analysed the performance in terms of both exact match and F1 score with respect to the answer length on the development set. For example, for questions whose answers contain more than 9 tokens, the F1 score of the boundary model drops to around 55% and the exact match score drops to only around 30%, compared to the F1 score and exact match score of close to 72% and 67%, respectively, for questions with single-token answers. And that supports our hypothesis.

Next, we analyze the performance of our models on different groups of questions. We use a crude way to split the questions into different groups based on a set of question words we have defined, including "what," "how," "who," "when," "which," "where," and "why." These different question words roughly refer to questions with different types of answers. For example, "when" questions look for temporal expressions as answers, whereas "where" questions look for locations as answers. According to the performance on the development data set, our models work the best for "when" questions. This may be because in this dataset temporal expressions are relatively easier to recognize. Other groups of questions whose answers are noun phrases, such as "what" questions, "which" questions and "where" questions, also get relatively better results. On the other hand, "why" questions are the hardest to answer. This is not surprising because the answers to "why" questions can be very diverse, and they are not restricted to any certain type of phrases.

Finally, we would like to check whether the attention mechanism used in the match-LSTM layer is effective in helping the model locate the answer. We show the attention weights α in Figure 2. In the figure the darker the color is the higher the weight is. We can see that some words have been well aligned based on the attention weights. For example, the word "German" in the passage is aligned well to the word "language" in the first question, and the model successfully predicts "German" as the answer to the question. For the question word "who" in the second question, the word "teacher" actually receives relatively higher attention weight, and the model has predicted the phrase "Martin Sekulic" after that as the answer, which is correct. For the third question that starts with "why", the attention weights are more evenly distributed and it is not clear which words have been aligned to "why". For the last question, we can see that the word knowledge needed for generating the answer can also be detected by match-LSTM. For example, the words "European", "Parliament", "Council", "European" and "Union" have higher attention weights on "governing" in the question.

4 RELATED WORK

Machine comprehension of text has gained much attention in recent years, and increasingly researchers are building data-drive, end-to-end neural network models for the task. We will first review the recently released datasets and then some end-to-end models on this task.

4.1 DATASETS

A number of datasets for studying machine comprehension were created in Cloze style by removing a single token from a sentence in the original corpus, and the task is to predict the missing word. For example, Hermann et al. (2015) created questions in Cloze style from CNN and Daily Mail highlights. Hill et al. (2016) created the Children's Book Test dataset, which is based on children's stories. Cui et al. (2016) released two similar datasets in Chinese, the People Daily dataset and the Children's Fairy Tale dataset.

Instead of creating questions in Cloze style, a number of other datasets rely on human annotators to create real questions. Richardson et al. (2013) created the well-known MCTest dataset and Tapaswi et al. (2016) created the MovieQA dataset. In these datasets, candidate answers are provided for each question. Similar to these two datasets, the SQuAD dataset (Rajpurkar et al., 2016) was also created by human annotators. Different from the previous two, however, the SQuAD dataset does not provide candidate answers, and thus all possible subsequences from the given passage have to be considered as candidate answers.

Besides the datasets above, there are also a few other datasets created for machine comprehension, such as WikiReading dataset (Hewlett et al., 2016) and bAbI dataset (Weston et al., 2016), but they are quite different from the datasets above in nature.

4.2 END-TO-END NEURAL NETWORK MODELS FOR MACHINE COMPREHENSION

There have been a number of studies proposing end-to-end neural network models for machine comprehension. A common approach is to use recurrent neural networks (RNNs) to process the given text and the question in order to predict or generate the answers (Hermann et al., 2015). Attention mechanism is also widely used on top of RNNs in order to match the question with the given passage (Hermann et al., 2015; Chen et al., 2016). Given that answers often come from the given passage, Pointer Network has been adopted in a few studies in order to copy tokens from the given passage as answers (Kadlec et al., 2016; Trischler et al., 2016). Compared with existing work, we use match-LSTM to match a question and a given passage, and we use Pointer Network in a different way such that we can generate answers that contain multiple tokens from the given passage.

Memory Networks (Weston et al., 2015) have also been applied to machine comprehension (Sukhbaatar et al., 2015; Kumar et al., 2016; Hill et al., 2016), but its scalability when applied to a large dataset is still an issue. In this work, we did not consider memory networks for the SQuAD/MSMARCO datasets.

The setting of visual question answering (Antol et al., 2015) is quite similar to machine comprehension, while their answers are usually very short. So the sequence order of the word-level attention

representation used to align the figure and the question(Xu & Saenko, 2016; Fukui et al., 2016; Lu et al., 2016), are not used in VQA. While our model focus on the word-by-word attention and use LSTM to concatenate the aligned pairs and that would be helpful to generate a longer sequence as answer.

5 CONCLUSIONS

In this paper, We developed two models for the machine comprehension problem defined in the Stanford Question Answering (SQuAD) and A Human-Generated MAchine Reading COmprehension (MSMARCO) datasets, both making use of match-LSTM and Pointer Network. Experiments on the SQuAD and MSMARCO datasets showed that our second model, the boundary model, could achieve a performance close to the state-of-the-art performance on the SQuAD dataset and achieved the state-of-the-art on the MSMARCO dataset. We also show the boundary model could overcome the early stop prediction problem of the sequence model.

In the future, we plan to look further into the different types of questions and focus on those questions which currently have low performance, such as the "why' questions and multi-sentences related questions. We also plan to test how our models could be applied to other machine comprehension datasets.

6 ACKNOWLEDGMENTS

REFERENCES

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433, 2015.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. Consensus attention-based neural networks for chinese reading comprehension. In *arXiv preprint arXiv:1607.02250*, 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 *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. WIKIREADING: A novel large-scale language understanding task over wikipedia. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *Proceedings of the Conference on Association for Computational Linguistics*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*, 2016.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2016.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: a human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Proceedings of the Conference on Advances in neural information processing systems*, 2015.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. MovieQA: Understanding stories in movies through question-answering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the EpiReader. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, 2015.

Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In *Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics*, 2016.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *Proceedings of the International Conference on Learning Representations*, 2015.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the International Conference on Learning Representations*, 2016.

Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 2016.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*, 2016.

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

A APPENDIX

We show the performance breakdown by answer lengths and question types for our sequence model, boundary model and the ensemble model in Figure 3.
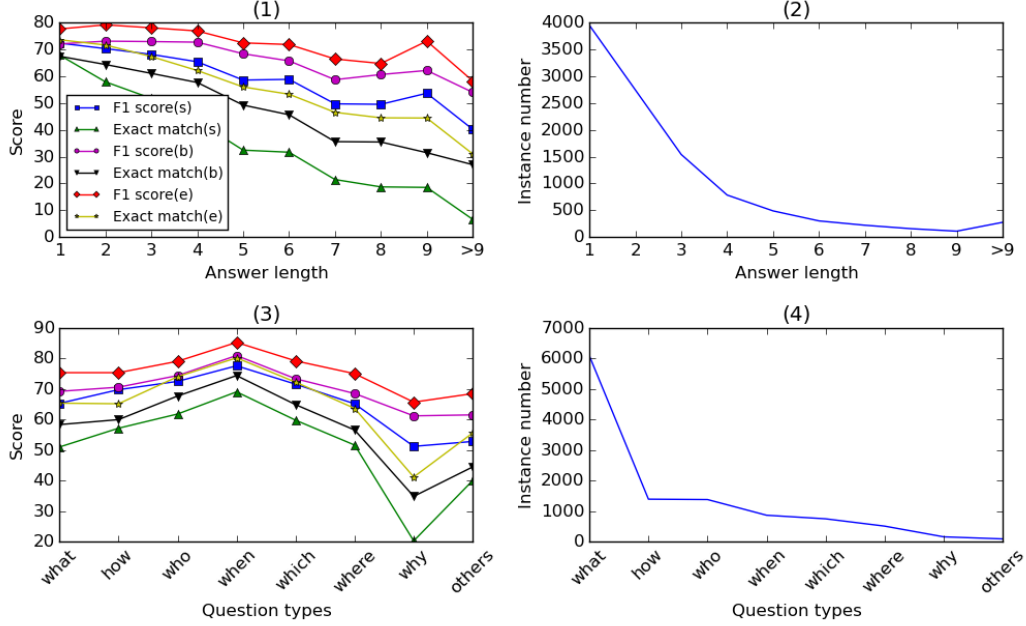

Figure 3: Performance breakdown by answer lengths and question types. Top: Plot (1) shows the performance of our two models (where *s* refers to the sequence model , *b* refers to the boundary model, and *e* refers to the ensemble boundary model) over answers with different lengths. Plot (2) shows the numbers of answers with different lengths. Bottom: Plot (3) shows the performance our the two models on different types of questions. Plot (4) shows the numbers of different types of questions.

B APPENDIX

We show the predictions our boundary and sequence models on two cases from two data sets in Table 3. It can be seen that the sequence model is more likely to predict a shorter sequence which is the problem of early stop prediction.

(1) Context	Asopposed to broadcasts of primetime series, CBS broadcast special episodes of its late night talk shows as its lead-out programs for Super Bowl 50, beginning with a special episode of **The Late Show with Stephen Colbert** following the game.
Question (Syntactic)	What CBS show followed the Super Bowl?
Golden Anwser	The Late Show with Stephen Colbert
Match-LSTM (Sequence)	The Late Show
Match-LSTM (Boundary)	The Late Show with Stephen Colbert
(2) Context	Urinalysis is a test that evaluates a sample of your urine. Urinalysis is used to **detect and assess a wide range of disorders, such as urinary tract infection, kidney disease and diabetes.** Urinalysis involves examining the appearance, concentration and content of urine. Abnormal urinalysis results may point to a disease or illness. For example, a urinary tract infection can make urine look cloudy instead of clear. Increased levels of protein in urine can be a sign of kidney disease.
Query	what can urinalysis detect?
Golden Anwser	Detect and assess a wide range of disorders, such as urinary tract infection, kidney disease and diabetes.
Match-LSTM (Sequence)	Urinalysis
Match-LSTM (Boundary)	Urinalysis is used to detect and assess a wide range of disorders, such as urinary tract infection, kidney disease and diabetes

Table 3: Prediction samples for sequence and boundary models. The first case is sampled from SQuAD data set and the second is sampled from MSMARCO data set.

C   APPENDIX

(1) Context	The Rankine cycle is sometimes referred to as a **practical Carnot cycle** because, when an efficient turbine is used, the TS diagram begins to resemble the Carnot cycle.
Question (Synonymy)	What is the Rankine cycle sometimes called?
Golden Anwser	practical Carnot cycle
LSTM (Sequence)	Carnot cycle
Match-LSTM (Sequence)	Carnot cycle
LSTM (Boundary)	practical Carnot cycle
Match-LSTM (Boundary)	Carnot cycle
(2) Context	While the Commission has a monopoly on initiating legislation, **the European Parliament and the Council of the European Union** have powers of amendment and veto during the legislative process.
Question (Knowledge)	Which two governing bodies have legislative veto power?
Golden Anwser	the European Parliament and the Council of the European Union
LSTM (Sequence)	European Parliament and the Council of the European Union
Match-LSTM (Sequence)	European Parliament and the Council of the European Union
LSTM (Boundary)	European Parliament and the Council of the European Union
Match-LSTM (Boundary)	European Parliament and the Council of the European Union
(3) Context	Current faculty include the anthropologist Marshall Sahlins, historian Dipesh Chakrabarty, ... Shakespeare scholar **David Bevington**, and renowned political scientists John Mearsheimer and Robert Pape.
Question (Syntactic)	What Shakespeare scholar is currently on the university's faculty?
Golden Anwser	David Bevington
LSTM (Sequence)	Marshall Sahlins
Match-LSTM (Sequence)	David Bevington
LSTM (Boundary)	Marshall Sahlins
Match-LSTM (Boundary)	David Bevington
(4) Context	The V&A Theatre & Performance galleries, formerly the Theatre Museum, opened in March 2009. The collections are stored by the V&A, and are available for research, exhibitions and other shows. They hold the UK's biggest national collection of **material about live performance** in the UK since Shakespeare's day, covering drama, dance, musical theatre, circus, music hall, rock and pop, and most other forms of live entertainment.
Question (Reasoning)	What collection does the V&A Theatre & Performance galleries hold?
Golden Anwser	material about live performance
LSTM (Sequence)	Theatre
Match-LSTM (Sequence)	the Theatre Museum
LSTM (Boundary)	research, exhibitions and other shows
Match-LSTM (Boundary)	Theatre Museum
(5) Context	Along with giving the offender his "just deserts", **achieving crime control via incapacitation** and deterrence is a major goal of criminal punishment.
Question (Ambiguous)	What is the main goal of criminal punishment of civil disobedients?
Golden Anwser	achieving crime control via incapacitation and deterrence
LSTM (Sequence)	deterrence
Match-LSTM (Sequence)	just deserts
LSTM (Boundary)	incapacitation and deterrence
Match-LSTM (Boundary)	incapacitation and deterrence

Table 4: Different types of reasoning samples in SQuAD data set. "Match-LSTM" refers to the "Match-LSTM with Ans-Ptr" and "LSTM" refers to the "LSTM with Ans-Ptr" which is the ablation of attention mechanism in Match-LSTM.

We show how four different models work on different type questions in SQuAD data set through Table 4. After the analysis of a hundred cases, we see that our models are not able to solve the questions that need multi-sentences reasoning. And the model without attention mechanism has less power to identify the important key word like the third case shown in Table 4.