END-TO-END ANSWER CHUNK EXTRACTION AND RANKING FOR READING COMPREHENSION

Yang Yu,* Wei Zhang,* Bowen Zhou, Kazi Hasan, Mo Yu, Bing Xiang
{yu, zhangwei, zhou, kshasan, yum, bingxia}@us.ibm.com
IBM Watson, Yorktown Heights, NY, USA

ABSTRACT

This paper proposes *dynamic chunk reader* (*DCR*), an end-to-end neural reading comprehension (RC) model that is able to extract and rank a set of answer candidates from a given document to answer questions. DCR is able to predict answers of variable lengths, whereas previous neural RC models primarily focused on predicting single tokens or entities. DCR encodes a document and an input question with recurrent neural networks, and then applies a word-by-word attention mechanism to acquire question-aware representations for the document, followed by the generation of chunk representations and a ranking module to propose the top-ranked chunk as the answer. Experimental results show that DCR could achieve a 66.3% Exact match and 74.7% F1 score on the Stanford Question Answering Dataset (Rajpurkar et al., 2016).

1 INTRODUCTION

Reading comprehension-based question answering (RCQA) is the task of answering a question with a chunk of text taken from related document(s). A variety of neural models have been proposed recently either for extracting a single entity or a single token as an answer from a given text (Hermann et al., 2015; Kadlec et al., 2016; Trischler et al., 2016b; Dhingra et al., 2016; Chen et al., 2016; Sordoni et al., 2016; Cui et al., 2016a); or for selecting the correct answer by ranking a small set of human-provided candidates (Yin et al., 2016; Trischler et al., 2016a). In both cases, an answer boundary is either easy to determine or already given.

Different from the above two assumptions for RCQA, in the real-world QA scenario, people may ask questions about both entities (factoid) and non-entities such as explanations and reasons (non-factoid) (see Table 1 for examples).

In this regard, RCQA has the potential to complement other QA approaches that leverage structured data (e.g., knowledge bases) for both the above question types. This is because RCQA can exploit the textual evidences to ensure increased answer coverage, which is particularly helpful for non-factoid answers. However, it is also challenging for RCQA to identify answer in arbitrary position in the passage with arbitrary length, especially for non-factoid answers which might be clauses or sentences.

As a result, apart from a few exceptions (Rajpurkar et al., 2016; Wang & Jiang, 2016), this research direction has not been fully explored yet.

Compared to the relatively easier RC task of predicting single tokens/entities[1], predicting answers of arbitrary lengths and positions significantly increase the search space complexity:

the number of possible candidates to consider is in the order of $O(n^2)$, where n is the number of passage words. In contrast, for previous works in which answers are single tokens/entities or from candidate lists, the complexity is in $O(n)$ or the size of candidate lists l (usually $l \leq 5$), respectively. To address the above complexity, Rajpurkar et al. (Rajpurkar et al., 2016) used a two-step chunk-and-rank approach that employs a rule-based algorithm to extract answer candidates from a passage,

*Both authors contribute equally

[1]State-of-the-art RC models have a decent accuracy of $\sim$70% on the widely used CNN/DailyMail dataset (Hermann et al., 2015).

Table 1: Example of questions (with answers) which can be potentially answered with RC on a Wikipedia passage. The first question is factoid, asking for an entity. The second and third are non-factoid.

The United Kingdom (UK) intends to withdraw from the European Union (EU), a process commonly known as Brexit, as a result of a June 2016 referendum in which 51.9% voted to leave the EU. The separation process is complex, causing political and economic changes for the UK and other countries. As of September 2016, neither the timetable nor the terms for withdrawal have been established: in the meantime, the UK remains a full member of the European Union. The term "Brexit" is a portmanteau of the words "British" and "exit".
Q1. Which country withdrew from EU in 2016? A1. United Kingdom
Q2. How did UK decide to leave the European Union? A2. as a result of a June 2016 referendum in which 51.9% voted to leave the EU
Q3. What has not been finalized for Brexit as of September 2016? A3. neither the timetable nor the terms for withdrawal

followed by a ranking approach with hand-crafted features to select the best answer. The rule-based chunking approach suffered from low coverage ($\approx$ 70% recall of answer chunks) that cannot be improved during training; and candidate ranking performance depends greatly on the quality of the hand-crafted features. More recently, Wang and Jiang (Wang & Jiang, 2016) proposed two end-to-end neural network models, one of which chunks a candidate answer by predicting the answer's two boundary indices and the other classifies each passage word into answer/not-answer. Both models improved significantly over the method proposed by Rajpurkar et al. (Rajpurkar et al., 2016).

Our proposed model, called *dynamic chunk reader* (*DCR*), not only significantly differs from both the above systems in the way that answer candidates are generated and ranked, but also shares merits with both works. First, our model uses deep networks to learn better representations for candidate answer chunks, instead of using fixed feature representations as in (Rajpurkar et al., 2016). Second, it represents answer candidates as chunks, as in (Rajpurkar et al., 2016), instead of word-level representations (Wang & Jiang, 2016), to make the model aware of the subtle differences among candidates (importantly, overlapping candidates).

The contributions of this paper are three-fold. (1) We propose a novel neural network model for joint candidate answer chunking and ranking, where the candidate answer chunks are dynamically constructed and ranked in an end-to-end manner. (2) we propose a new question-attention mechanism to enhance passage word representation, which is subsequently used to construct chunk representations. (3) We also propose several simple but effective features to strengthen the attention mechanism, which fundamentally improves candidate ranking, with the by-product of higher exact boundary match accuracy.

The experiments on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), which contains a variety of human-generated factoid and non-factoid questions, have shown the effectiveness of above three contributions.

Our paper is organized as follows. We formally define the RCQA problem first. Next, we describe our baseline with a neural network component. We present the end-to-end dynamic chunk reader model next. Finally, we analyze our experimental results and discuss the related work. In appendix, we show formal equations and details of the model.

2 Problem Definition

Table 1 shows an example of our RC setting where the goal is to answer a question Q_i, factoid (Q1) or non-factoid (Q2 and Q3), based on a supporting passage P_i, by selecting a continuous sequence of text $A_i \subseteq P_i$ as answer. Q_i, P_i, and A_i are all word sequences, where each word is drawn from a vocabulary, V. The i-th instance in the training set is a triple in the form of (P_i, Q_i, A_i), where $P_i = (p_{i1}, \ldots, p_{i|P_i|})$, $Q_i = (q_{i1}, \ldots, q_{i|Q_i|})$, and $A_i = (a_{i1}, \ldots, a_{i|A_i|})$ $(p_{i\cdot}, q_{i\cdot}, a_{i\cdot} \in V)$. Owing to the disagreement among annotators, there could be more than one correct answer for the same question; and the k-th answer to Q_i is denoted by $A_i^k = \{a_{i1}^k, \ldots, a_{i|A_i^k|}^k\}$. An answer candidate for the i-th training example is defined as $c_i^{m,n}$, a sub-sequence in P_i, that spans from position m to n $(1 \leq m \leq n \leq |P_i|)$. The ground truth answer A_i could be included in the set of all candidates

$C_i = \{c_i^{m,n} | \forall m, n \in N^+, subj(m, n, P_i)\ and\ 1 \leq m \leq n \leq |P_i|\}$, where $subj(m, n, P_i)$ is the constraint put on the candidate chunk for P_i, such as, "$c_i^{m,n}$ can have at most 10 tokens", or "$c_i^{m,n}$ must have a pre-defined POS pattern". To evaluate a system's performance, its top answer to a question is matched against the corresponding gold standard answer(s).

Remark: Categories of RC Tasks Other simpler variants of the aforementioned RC task were explored in the past. For example, ***quiz-style*** datasets (e.g., MCTest (Richardson et al., 2013), MovieQA (Tapaswi et al., 2015)) have multiple-choice questions with answer options. ***Cloze-style*** datesets(Hermann et al., 2015; Hill et al., 2015; Onishi et al., 2016), usually automatically generated, have factoid "question"s created by replacing the answer in a sentence from the text with blank. For the ***answer selection*** task this paper focuses on, several datasets exist, e.g. TREC-QA for factoid answer extraction from multiple given passages, bAbI (Weston et al., 2014) designed for inference purpose, and the SQuAD dataset (Rajpurkar et al., 2016) used in this paper. To the best of our knowledge, the SQuAD dataset is the only one for both factoid and non-factoid answer extraction with a question distribution more close to real-world applications.

3 BASELINE: CHUNK-AND-RANK PIPELINE WITH NEURAL RC

In this section we modified a state-of-the-art RC system for cloze-style tasks for our answer extraction purpose, to see how much gap we have for the two type of tasks, and to inspire our end-to-end system in the next section. In order to make the cloze-style RC system to make chunk-level decision, we use the RC model to generate features for chunks, which are further used in a feature-based ranker like in (Rajpurkar et al., 2016). As a result, this baseline can be viewed as a deep learning based counterpart of the system in (Rajpurkar et al., 2016). It has two main components: 1) a stand-alone answer chunker, which is trained to produce overlapping candidate chunks, and 2) a neural RC model, which is used to score each word in a given passage to be used thereafter for generating chunk scores.

Answer Chunking To reduce the errors generated by the rule-based chunker in (Rajpurkar et al., 2016), first, we capture the part-of-speech (POS) pattern of all answer sub-sequences in the training dataset to form a *POS pattern trie tree*, and then apply the answer POS patterns to passage P_i to acquire a collection of all subsequences (chunk candidates) C_i whose POS patterns can be matched to the *POS pattern trie*. This is equivalent to putting an constraint $subj(m, n, P_i)$ to candidate answer chunk generation process that only choose the chunk with a POS pattern seen for answers in the training data. Then the sub-sequences C_i are used as answer candidates for P_i. Note that overlapping chunks could be generated for a passage, and we rely on the ranker to choose the best candidate based on features from the cloze-style RC system. Experiments showed that for $> 90\%$ of the questions on the development set, the ground truth answer is included in the candidate set constructed in such manner.

Feature Extraction and Ranking For chunk ranking, we (1) use neural RCQA model to annotate each p_{ij} in passage P_i to get score s_{ij}, then (2) for every chunk $c_i^{m,n}$ in passage i, collect scores $(s_{im}, \ldots, s_{in})$ for all the $(p_{im}, ..., p_{in})$ contained within $c_i^{m,n}$, and (3) extract features on the sequence of scores $(s_{im}, \ldots, s_{in})$ to characterize its scale and distribution information, which serves as the feature representation of $c_i^{m,n}$. In step (1) to acquire s_{ij} we train and apply a word-level single-layer Gated Attention Reader [2] (Dhingra et al., 2016), which has state-of-the-art performance on CNN/DailyMail cloze-style RC task. In step (3) for chunk $c_i^{m,n}$, we designed 5 features, including 4 statistics on $(s_{im}, \ldots, s_{in})$: *maximum, minimum, average and sum*; as well as the count of matched POS pattern within the chunk, which serves as an answer prior. We use these 5 features in a state-of-the-art ranker (Ganjisaffar et al., 2011).

4 DYNAMIC CHUNK READER

The dynamic chunk reader (DCR) model is presented in Figure 1. Inspired by the baseline we built, DCR is deemed to be superior to the baseline for 3 reasons. First, each chunk has a representation constructed dynamically, instead of having a set of pre-defined feature values. Second, each passage

[2] We tried using more than one layers in Gated Attention Reader, but no improvement was observed.

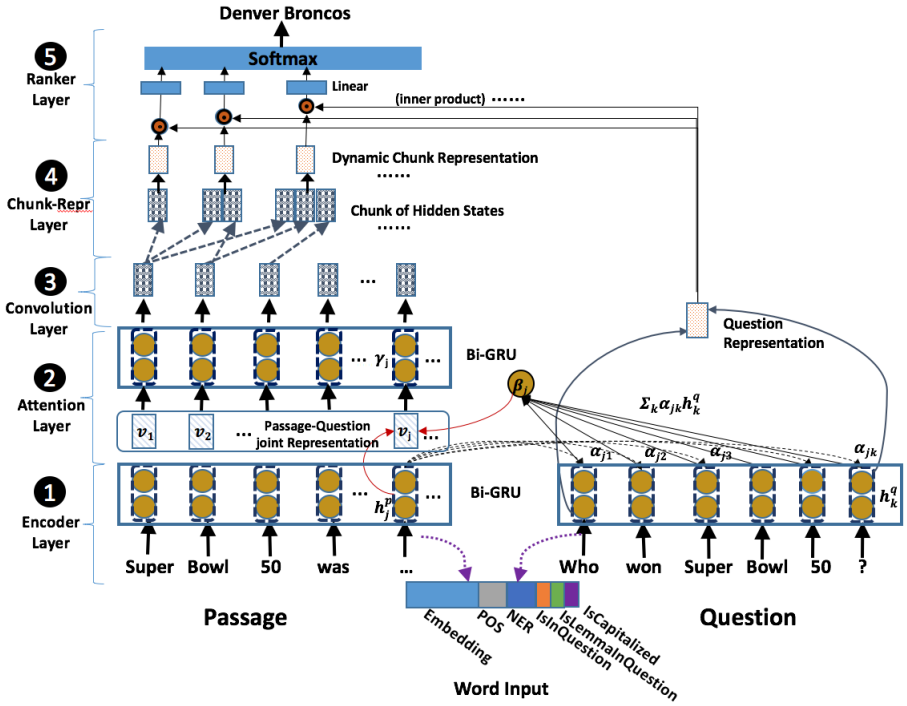


Figure 1: The main components in dynamic chunk reader model (from bottom to top) are bi-GRU encoders for passage and question, a word-by-word attention bi-GRU for passage, dynamic chunk representations that are transformed from pooled dynamic chunks of hidden states, the question attention on every chunk representation and final answer chunk prediction.

word's representation is enhanced by word-by-word attention that evaluates the relevance of the passage word to the question. Third, these components are all within a single, end-to-end model that can be trained in a joint manner.

DCR works in four steps. First, the ***encoder layer*** encodes passage and question separately, by using bidirectional recurrent neural networks (RNN).

Second, the ***attention layer*** calculates the relevance of each passage word to the question.

Third, the ***convolution layer*** generates unigram, bigram and trigram representation for each word. bigram and trigram of a word ends with the same word, and proper padding is applied on the first word to make sure the output is the same length as input to CNN layer.

Fourth, the ***chunk representation layer*** dynamically extracts the candidate chunks from the given passage, and create chunk representation that encodes the contextual information of each chunk.

Fifth, the ***ranker layer*** scores the relevance between the representations of a chunk and the given question, and ranks all candidate chunks using a softmax layer.

We describe each step below.

Encoder Layer We use bi-directional RNN encoder to encode P_i and Q_i of example i, and get hidden state for each word position p_{ij} and q_{ik}.[3] As RNN input, a word is represented by a row vector $x \in \mathbb{R}^n$. x can be the concatenation of word embedding and word features (see Fig. 1). The word vector for the t-th word is x_t. A word sequence is processed using an RNN encoder with gated recurrent units (GRU) (Cho et al., 2014), which was proved to be effective in RC and neural machine translation tasks (Bahdanau et al., 2015; Kadlec et al., 2016; Dhingra et al., 2016). For each position t, GRU computes h_t with input x_t and previous state h_{t-1}, as:

[3]We can have separated parameters for question and passage encoders but a single shared encoder for both works better in the experiments.

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \quad (1)$$
$$u_t = \sigma(W_u x_t + U_u h_{t-1}) \quad (2)$$
$$\bar{h_t} = tanh(W x_t + U(r_t \odot h_{t-1})) \quad (3)$$
$$h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \bar{h_t} \quad (4)$$

where h_t, r_t, and $u_t \in \mathbb{R}^d$ are d-dimensional hidden state, reset gate, and update gate, respectively; $W_{\{r,u\}}$, $W \in \mathbb{R}^{n \times d}$ and $U_{\{r,u\}}$, $U \in \mathbb{R}^{d \times d}$ are the parameters of the GRU; σ is the sigmoid function, and $\odot$ denotes element-wise production. For a word at t, we use the hidden state $\overrightarrow{h}_t$ from the forward RNN as a representation of the preceding context, and the $\overleftarrow{h}_t$ from a backward RNN that encodes text reversely, to incorporate the context after t. Next, $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$, the bi-directional contextual encoding of x_t, is formed. $[\cdot;\cdot]$ is the concatenation operator. To distinguish hidden states from different sources, we denote the h_j of j-th word in P and the h_k of k-th word in Q as h_j^p and h_k^q respectively.

Attention Layer Attention mechanism in previous RC tasks (Kadlec et al., 2016; Hermann et al., 2015; Sordoni et al., 2016; Dhingra et al., 2016; Cui et al., 2016a;b) enables question-aware passage representations. We propose a novel attention mechanism inspired by word-by-word style attention methods (Rocktäschel et al., 2015; Wang & Jiang, 2015; Santos et al., 2016). For each p_j, a question-attended representation v_j is computed as follows (example index i is omitted for simplicity):

$$\alpha_{jk} = h_j^p \cdot h_k^q, \quad (5)$$
$$\beta_j = \sum_{k=1}^{|Q|} \alpha_{jk} h_k^q \quad (6)$$
$$v_j = [h_j^p; \beta_j] \quad (7)$$

where h_j^p and h_k^q are hidden states from the bi-directional RNN encoders (see Figure 1). An inner product, α_{jk}, is calculated between h_j^p and every question word h_k^q. It indicates how well the passage word p_j matches with every question word q_k. β_j is a weighted pooling of $|Q|$ question hidden states, which serves as a p_j-aware question representation. The concatenation of h_j^p and β_j leads to a passage-question joint representation, $v_j \in \mathbb{R}^{4d}$.[4] Next, we apply a second bi-GRU layer taking the v_js as inputs, and obtain forward and backward representations $\overrightarrow{\gamma_j}$ and $\overleftarrow{\gamma_j} \in \mathbb{R}^d$, and in turn their concatenation, $\gamma_j = [\overrightarrow{\gamma_j}; \overleftarrow{\gamma_j}]$.

Convolution Layer Every word is encoded with complete passage context through attention layer RNN. We would like to model more complex representation of the words, by introducing unigram, bigram and trigram representations. There are two benefits for this enhanced representation: 1) each word could be enhanced with local context information to help identify the boundary of the answer chunk. Using previous words has been a common feature used in POS tagging and Named entity recognition; and 2) The information brought in by the ngram into the word representation could enhance the semantic match between the answer chunk internal and the question. Imagine scenario of a three word candidate, where the last word representation includes the two previous words through the convolution layer. Matching to the last word could also lead to the match to the semantics of the internal of the chunk. Specifically, we create for every word position j three representations, by using ngrams ending with the hidden state j:

$$\tilde{\gamma}_{j1} = \gamma_j \cdot W_{c1} \quad (8)$$
$$\tilde{\gamma}_{j2} = [\gamma_{j-1}; \gamma_j] \cdot W_{c2} \quad (9)$$
$$\tilde{\gamma}_{j3} = [\gamma_{j-2}; \gamma_{j-1}; \gamma_j] \cdot W_{c3} \quad (10)$$

[4] We tried another word-by-word attention methods as in (Santos et al., 2016), which has similar passage representation input to question side. However, this does not lead to improvement due to the confusion caused by long passages in RC. Consequently, we used the proposed simplified version of word-by-word attention on passage side only.

The details shown in equations above. We used three different convolution kernels for different n-grams.

Chunk Representation Layer A candidate answer chunk representation is dynamically created given convolution layer output. We first decide the text boundary for the candidate chunk, and then form a chunk representation using all or part of those γ_j outputs inside the chunk. To decide a candidate chunk (boundary): we tried two ways: (1) adopt the *POS trie*-based approach used in our baseline, and (2) enumerate all possible chunks up to a maximum number of tokens. For (2), we create up to N (max chunk length) chunks starting from any position j in P_j. Approach (1) can generate candidates with arbitrary lengths, but fails to recall candidates whose POS pattern is unseen in training set; whereas approach (2) considers all possible candidates within a window and is more flexible, but over-generates invalid candidates.

For a candidate answer chunk $c^{m,n}$ spanning from position m to n inclusively, we construct chunk representation $\overline{\gamma}^l_{m,n} \in \mathbb{R}^{2d}$ using every $\tilde{\gamma}_{jl}$ within range $[m, n]$, with a function $g(\cdot)$, and $l \in \{1, 2, 3\}$. Formally,

$$\overline{\gamma}^l_{m,n} = g(\tilde{\gamma}_{ml}, \ldots, \tilde{\gamma}_{nl})$$

Each $\tilde{\gamma}_{jl}$ is a convolution output over concatenated forward and backward RNN hidden states from attention layer. So the first half in $\tilde{\gamma}_{jl}$ encodes information in forward RNN hidden states and the second half encodes information in backward RNN hidden states. We experimented with several pooling functions (e.g., max, average) for $g(\cdot)$, and found out that, instead of pooling, the best $g(\cdot)$ function is to concatenate the first half of convolution output of the chunk's first word and the second half of convolution output of the chunk's last word. Formally,

$$\overline{\gamma}^l_{m,n} = g(\tilde{\gamma}_{ml}, \ldots, \tilde{\gamma}_{nl}) = [\overrightarrow{\tilde{\gamma}_{ml}}; \overleftarrow{\tilde{\gamma}_{nl}}] \tag{11}$$

where $\overrightarrow{\tilde{\gamma}_{ml}}$ is half of the hidden state for l-gram word representation corresponding to forward attention RNN output. We hypothesize that the hidden states at that two ends can better represent the chunk's contexts, which is critical for this task, than the states within the chunk. This observation also agrees with (Kobayashi et al., 2016).

Ranker Layer A score $s^l_{m,n}$ for each l-gram chunk representation $\overline{\gamma}^l_{m,n}$ denoting the probability of that chunk to be the true answer is calculated by dot product with question representation. The question representation is the concatenation of the last hidden state in forward RNN and the first hidden state in backward RNN. Formally for the chunk $c^{m,n}_i$ we have

$$s^l(c^{m,n}_i | P_i, Q_i) = \overline{\gamma}^l_{m,n} \cdot [\overrightarrow{h^{Q_i}_{|Q_i|}}; \overleftarrow{h^{Q_i}_1}] \tag{12}$$

where s^l denotes the score generated from l-gram representation. $\overrightarrow{h^{Q_i}_k}$ or $\overleftarrow{h^{Q_i}_k}$ is the k-th hidden state output from question Q_i's forward and backward RNN encoder, respectively.

After that, the final score for $c^{m,n}_i$ is evaluated as the linear combination of three scores, followed by a softmax:

$$s(c^{m,n}_i | P_i, Q_i) = softmax(W \cdot [s^1; s^2; s^3]) \tag{13}$$

where s^l is the shorthand notation for $s^l(c^{m,n}_i | P_i, Q_i)$; $W \in \mathbb{R}^3$. In runtime, the chunk with the highest probability is taken as the answer. In training, the following negative log likelihood is minimized:

$$\mathbb{L} = -\sum_{i=1}^{N} \log \mathbb{P}(A_i | P_i, Q_i) \tag{14}$$

Note that the i-th training instance is only used when A_i is included in the corresponding candidate chunk set C_i, i.e. $\exists_{m,n} A_i = c^{m,n}_i$. The softmax in the final layer serves as the list-wise ranking module similar in spirit to (Cao et al., 2007).

5 EXPERIMENTS

Dataset We used the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) for the experiment. SQuAD came into our sight because it is a mix of factoid and non-factoid

Table 2: Results on the SQuAD dataset.

	Dev		Test	
Models	EM	F1	EM	F1
Rajpurkar 2016	39.8%	51.0%	40.4%	51.0%
Wang 2016	59.1%	70.0%	59.5%	70.3%
DCR w/o Conv.	62.5%	71.2%	62.5%	71.0%
DCR	63.4%	72.3%	-	-
DCR Ensemble	66.3%	74.7%	-	-

questions, a real-world data (crowd-sourced), and of large scale (over 100K question-answer pairs collected from 536 Wikipedia articles). Answers range from single words to long, variable-length phrase/clauses. It is a relaxation of assumptions by the cloze-style and quiz-style RC datasets in the Problem Definition section.

Features The input vector representation of each word w to encoder RNNs has six parts including a pre-trained 300-dimensional GloVe embedding (Pennington et al., 2014) and five features (see Figure 1): (1) a one-hot encoding (46 dimensions) for the part-of-speech (POS) tag of w; (2) a one-hot encoding (14 dimensions) for named entity (NE) tag of w; (3) a binary value indicating whether w's surface form is the same to any word in the quesiton; (4) if the lemma form of w is the same to any word in the question; and (5) if w is caplitalized. Feature (3) and (4) are designed to help the model align the passage text with question. Note that some types of questions (e.g., "who", "when" questions) have answers that have a specific POS/NE tag pattern. For instance, "who" questions mostly have proper nouns/persons as answers and "when" questions may frequently have numbers/dates (e.g., a year) as answers. Thus, we believe that the model could exploit the co-relation between question types and answer POS/NE patterns easier with POS and NE tag features. **Implementation Details** We pre-processed the SQuAD dataset using Stanford CoreNLP tool[5] (Manning et al., 2014) with its default setting to tokenize the text and obtain the POS and NE annotations. To train our model, we used stochastic gradient descent with the ADAM optimizer (Kingma & Ba, 2014), with an initial learning rate of 0.001. All GRU weights were initialized from a uniform distribution between (-0.01, 0.01). The hidden state size, d, was set to 300 for all GRUs. The question bi-GRU shared parameters with the passage bi-GRU, while the attention-based passage bi-GRU had its own parameters. We shuffled all training examples at the beginning of each epoch and adopted a curriculum learning approach (Bengio et al., 2009), by sorting training instances by length in every 10 batches, to enable the model start learning from relatively easier instances and to harder ones. We also applied dropout of rate 0.2 to the embedding layer of input bi-GRU encoder, and gradient clipping when the norm of gradients exceeded 10. We trained in mini-batch style (mini-batch size is 180) and applied zero-padding to the passage and question inputs in each batch. We also set the maximum passage length to be 300 tokens, and pruned all the tokens after the 300-th token in the training set to save memory and speed up the training process. This step reduced the training set size by about 1.6%. During test, we test on the full length of passage, so that we don't prune out the potential candidates. We trained the model for at most 30 epochs, and in case the accuracy did not improve for 10 epochs, we stopped training.

For the feature ranking-based system, we used jforest ranker (Ganjisaffar et al., 2011) with LambdaMART-RegressionTree algorithm and the ranking metric was NDCG@10. For the Gated Attention Reader in baseline system, we replicated the method and use the same configurations as in (Dhingra et al., 2016).

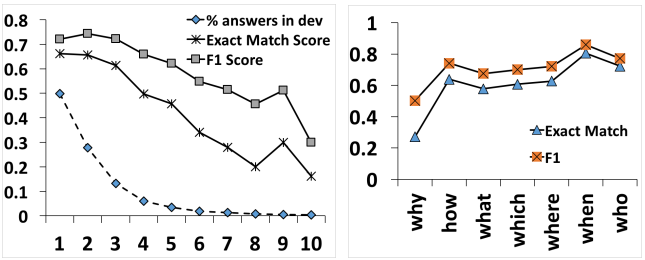
Results

Table 2 shows our main results on the SQuAD dataset. Compared to the scores reported in (Wang & Jiang, 2016), our exact match (EM) and F1 on the development set and EM score on the test set are better, and F1 on the test set is comparable. We also studied how each component in our model contributes to the overall performance. Table 3 shows the details as well as the results of the baseline ranker. As the first row of Table 3 shows, our baseline system improves 10% (EM) over Rajpurkar et al. (Rajpurkar et al., 2016) (Table 2, row 1), the feature-based ranking system. However when compared to our DCR model (Table 3, row 2), the baseline (row 1) is more than 12% (EM) behind

[5] stanfordnlp.github.io/CoreNLP/

Table 3: Detailed system experiments on the SQuAD development set.

Models	EM	F1
Chunk-and-Rank Pipeline Baseline	49.7%	64.9%
DCR w/o Convolution	62.5%	71.2%
DCR w/o Word-by-Word Attention	57.6%	68.7%
DCR w/o POS feature (1)	59.2%	68.8%
DCR w/o NE feature (2)	60.4%	70.2%
DCR w/o Question-word feature (3)	59.5%	69.0%
DCR w/o Question-lemma feature (4)	61.2%	69.9%
DCR w/o Capitalized feature (5)	61.5%	70.6%
DCR w/o Conv. w POS-trie	62.1%	70.8%



(a) (b)

Figure 2: (a) Variations of DCR performance on ground truth answer length (up to 10) in the development set. The curve with diamond knots also shows the percentage of answers for each length in the development set. (b) Performance comparisons for different question head word.

even though it is based on the state-of-the-art model for cloze-style RC tasks. This can be attributed to the advanced model structure and end-to-end manner of DCR.

We also did ablation tests on our DCR model. First, replacing the word-by-word attention with Attentive Reader style attention (Hermann et al., 2015) decreases the EM score by about 4.5%, showing the strength of our proposed attention mechanism.

Second, we remove the features in input to see the contribution of each feature. The result shows that POS feature (1) and question-word feature (3) are the two most important features.

Finally, combining the DCR model with the proposed POS-trie constraints yields a score similar to the one obtained using the DCR model with all possible n-gram chunks. The result shows that (1) our chunk representations are powerful enough to differentiate even a huge amount of chunks when no constraints are applied; and (2) the proposed POS-trie reduces the search space at the cost of a small drop in performance.

Analysis To better understand our system, we calculated the accuracy of the attention mechanism of the gated attention reader used in our deep learning-based baseline. We found that it is 72% accurate i.e., 72% of the times a word with the highest attention score is inside the correct answer span. This means that, if we could accurately detect the boundary around the word with the highest attention score to form the answer span, we could achieve an accuracy close to 72%. In addition, we checked the answer recall of our candidate chunking approach. When we use a window size of 10, 92% of the time, the ground truth answer will be included in the extracted Candidate chunk set. Thus the upper bound of the exact match score of our baseline system is around 66% (92% (the answer recall) $\times$ 72%). From the results, we see our DCR system's exact match score is at 62%. This shows that DCR is proficient at differentiating answer spans dynamically.

To further analyze the system's performance while predicting answers of different lengths, we show the exact match (EM) and F1 scores for answers with lengths up to 10 tokens in Figure 2(a). From the graph, we can see that, with the increase of answer length, both EM and F1 drops, but in different speed. The gap between F1 and exact match also widens as answer length increases. However, the model still yields a decent accuracy when the answer is longer than a single word. Additionally, Figure 2(b) shows that the system is better at "when" and "who" questions, but performs poorly

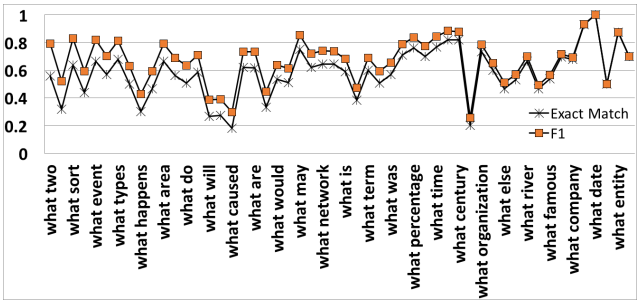


Figure 3: Development set performance comparisons for different types of "what" questions (considering the types with more than 20 examples in the development set).

on "why" questions. The large gap between exact match and F1 on "why" questions means that perfectly identifying the span is harder than locating the core of the answer span.

Since "what", "which", and "how" questions contain a broad range of question types, we split them further based on the bigram a question starts with, and Figure 3 shows the breakdown for "what" questions. We can see that "what" questions asking for explanations such as "what happens" and "what happened" have lower EM and F1 scores. In contrast, "what" questions asking for year and numbers have much higher scores and, for these questions, exact match scores are close to F1 scores, which means chunking for these questions are easier for DCR.

6 Related Work

Attentive Reader was the first neural model for factoid RCQA (Hermann et al., 2015). It uses Bidirectional RNN (Cho et al., 2014; Chung et al.,2014) to encode document and query respectively, and use query representation to match with every token from the document. Attention Sum Reader (Kadlec et al., 2016) simplifies the model to just predicting positions of correct answer in the document and the training speed and test accuracy are both greatly improved on the CNN/Daily Mail dataset. (Chen et al., 2016) also simplified Attentive Reader and reported higher accuracy. Window-based Memory Networks (MemN2N) is introduced along with the CBT dataset (Hill et al., 2015), which does not use RNN encoders, but embeds contexts as memory and matches questions with embedded contexts. Those models' mechanism is to learn the match between answer context with question/query representation. In contrast, memory enhanced neural networks like Neural Turing Machines (Graves et al., 2014) and its variants (Zhang et al., 2015; Gulcehre et al., 2016; Zaremba & Sutskever, 2015; Chandar et al., 2016; Grefenstette et al., 2015) were also potential candidates for the task, and Gulcehre et al. (Gulcehre et al., 2016) reported results on the bAbI task, which is worse than memory networks. Similarly, sequence-to-sequence models were also used (Yu et al., 2015; Hermann et al., 2015), but they did not yield better results either.

Recently, several models have been proposed to enable more complex inference for RC task. For instance, gated attention model (Dhingra et al., 2016) employs a multi-layer architecture, where each layer encodes the same document, but the attention is updated from layer to layer. EpiReader (Trischler et al., 2016b) adopted a joint training model for answer extractor and reasoner, where the extractor proposes top candidates, and the reasoner weighs each candidate by examining entailment relationship between question-answer representation and the document. An iterative alternating attention mechanism and gating strategies were proposed in (Sordoni et al., 2016) to optimize the attention through several hops. In contrast, Cui et al. (Cui et al., 2016a;b) introduced fine-grained document attention from each question word and then aggregated those attentions from each question token by summation with or without weights. This system achieved the state-of-the-art score on the CNN dataset. Those different variations all result in roughly 3-5% improvement over attention sum reader, but none of those could achieve higher than that. Other methods include using dynamic entity representation with max-pooling (Kobayashi et al., 2016) that aims to change entity representation with context, and Weissenborn's (Weissenborn, 2016) system, which tries to separate entity from the context and then matches the question to context, scoring an accuracy around 70% on the CNN dataset.

However, all of those models assume that the answers are single tokens. This limits the type of questions the models can answer. Wang and Jiang (Wang & Jiang, 2016) proposed a match-lstm and achieved good results on SQuAD. However, this approach predicts a chunk boundary or whether a word is part of a chunk or not. In contrast, our approach explicitly constructs the chunk representations and similar chunks are compared directly to determine correct answer boundaries.

7 CONCLUSION

In this paper we proposed a novel neural reading comprehension model for question answering. Different from the previously proposed models for factoid RCQA, the proposed model, dynamic chunk reader, is not restricted to predicting a single named entity as an answer or selecting an answer from a small, pre-defined candidate list. Instead, it is capable of answering both factoid and non-factoid questions as it learns to select answer chunks that are suitable for an input question. DCR achieves this goal with a joint deep learning model enhanced with a novel attention mechanism and five simple yet effective features. Error analysis shows that the DCR model achieves good performance, but still needs to improve on predicting longer answers, which are usually non-factoid in nature.

REFERENCES

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136. ACM, 2007.

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*, 2016.

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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*, 2016a.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. Consensus attention-based neural networks for chinese reading comprehension. *arXiv preprint arXiv:1607.02250*, 2016b.

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

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. pp. 85–94, 2011. doi: http://doi.acm.org/10.1145/2009916.2009932.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pp. 1828–1836, 2015.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic neural turing machine with soft and hard addressing schemes. *arXiv preprint arXiv:1607.00036*, 2016.

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

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

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

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. Who did What: A large-scale person-centered cloze dataset. In *Proc. of EMNLP*, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pp. 1532–43, 2014.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, pp. 4, 2013.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*, 2016.

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

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. *arXiv preprint arXiv:1512.02902*, 2015.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*, 2016a.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*, 2016b.

Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

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

Dirk Weissenborn. Separating answers from queries for neural reading comprehension. *arXiv preprint arXiv:1607.03316*, 2016.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. URL http://arxiv.org/abs/1410.3916.

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, Chung-Wei Hang, and Bowen Zhou. Empirical study on deep learning models for question answering. *arXiv preprint arXiv:1510.07526*, 2015.

Wojciech Zaremba and Ilya Sutskever. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521*, 362, 2015.

Wei Zhang, Yang Yu, and Bowen Zhou. Structured memory for neural turing machines. *arXiv preprint arXiv:1510.03931*, 2015.