



## Article

# Modeling Extractive Question Answering Using Encoder-Decoder Models with Constrained Decoding and Evaluation-Based Reinforcement Learning

Shaobo Li , Chengjie Sun , Bingquan Liu, Yuanchao Liu and Zhenzhou Ji

School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China; shli@insun.hit.edu.cn (S.L.); liubq@hit.edu.cn (B.L.); lyc@insun.hit.edu.cn (Y.L.); jizhenzhou@hit.edu.cn (Z.J.)

\* Correspondence: sunchengjie@hit.edu.cn

**Abstract:** Extractive Question Answering, also known as machine reading comprehension, can be used to evaluate how well a computer comprehends human language. It is a valuable topic with many applications, such as in chatbots and personal assistants. End-to-end neural-network-based models have achieved remarkable performance on these tasks. The most frequently used approach to extract answers with neural networks is to predict the answer's start and end positions in the document, independently or jointly. In this paper, we propose another approach that considers all words in an answer jointly. We introduce an encoder-decoder model to learn from all words in the answer. This differs from previous works, which usually focused on the start and end and ignored the words in the middle. To help the encoder-decoder model to perform this task better, we employ evaluation-based reinforcement learning with different reward functions. The results of an experiment on the SQuAD dataset show that the proposed method can outperform the baseline in terms of F1 scores, offering another potential approach to solve the extractive QA task.

**Keywords:** natural language processing; question answering; encoder-decoder models; reinforcement learning; neural network; machine learning

**MSC:** 68T50



**Citation:** Li, S.; Sun, C.; Liu, B.; Liu, Y.; Ji, Z. Modeling Extractive Question Answering Using Encoder-Decoder Models with Constrained Decoding and Evaluation-Based Reinforcement Learning. *Mathematics* **2023**, *11*, 1624. <https://doi.org/10.3390/math11071624>

Academic Editor: Cheng-Chi Lee

Received: 8 March 2023

Revised: 20 March 2023

Accepted: 24 March 2023

Published: 27 March 2023



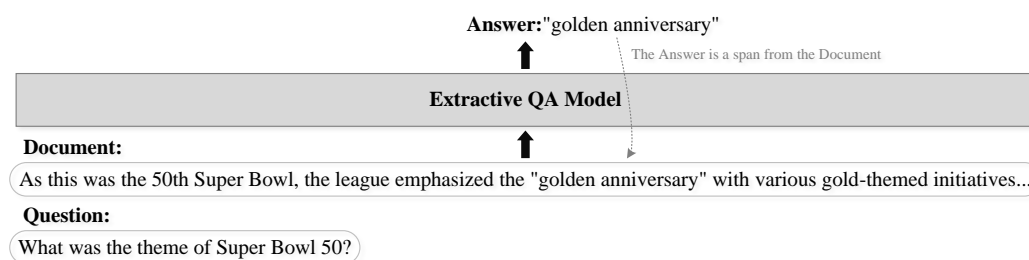
**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Text question-answering [1] systems can answer natural language questions automatically, providing a convenient way for people to obtain required knowledge. A successful system must be able to carry out several Natural Language Processing (NLP) tasks, such as natural language understanding [2], information retrieval [3], or natural language inference [4], making Question Answering (QA) one of the most challenging tasks that has attracted the interest of many NLP researchers.

Researchers have proposed and defined different types of text QA tasks, such as multichoice QA [5], generative QA [6], and extractive QA [7]. This paper focuses on extractive QA tasks, which take a question and a document as the input. The document contains a span that can serve as the correct answer for the question. Figure 1 shows an example. An extractive QA system needs to locate the start and end positions of the answer span in the input document and “extract” the span as the answer.

Extractive QA specifies the scope of the input question. The input question should be about a document, and the answer exists in the document as a substring. This setting enables us to confidently evaluate the predicted answers by comparing them to ground truth using overlap-based metrics, thus providing a relatively ideal and convenient test bed for QA models. Many works have used the extractive QA task to test models' ability to answer questions [2,8–10] or focus on achieving better results [11].



**Figure 1.** An extractive question-answering system takes a question and a document as the input and extracts a span from the document as the output answer.

The neural network models in [8,11–14] have achieved remarkable performances. They differ in their approaches for adapting end-to-end neural networks to model extractive QA tasks, e.g., the model proposed in [14] predicts the start and end positions of the span jointly, while [12] predicts them independently. This paper discusses several approaches to the modeling of extractive QA with neural networks. We propose the use of encoder-decoder models to solve the extractive QA task with evaluation-based reinforcement learning. The main ideas of this paper are as follows:

1. We solve the extractive QA task with an encoder-decoder model that generates all answer words jointly, enabling the model to use more information from the answers for training and to naturally output entire answers in the inference.
2. The proposed encoder-decoder extractive QA model uses evaluation-based reinforcement learning to enhance the model's performance. The experiment results show that the proposed model can achieve better results than the baseline.

The structure of this paper is as follows: Section 2 gives the background and related work, which includes a discussion about existing approaches to the training of neural-network-based extractive QA models, the introduction of the encoder-decoder model, and some general ideas behind reinforcement learning. Section 3 proposes our encoder-decoder model, the constrained decoding method, and evaluation-based reinforcement learning. Next, we present the experiment settings, results, and discussion in Section 4 and the conclusion in Section 5.

## 2. Background and Related Work

### 2.1. Extractive Question Answering

Extractive Question Answering is also known as span prediction or machine reading comprehension [15,16]. An extractive QA sample contains a question  $Q$ , a document  $D$ , and an answer  $A$  to question  $Q$ .  $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ ,  $D = \{d_1, d_2, \dots, d_{|D|}\}$ , and  $A = \{a_1, a_2, \dots, a_{|A|}\}$ . These are represented as word sequences where  $q_i$ ,  $d_i$ , and  $a_i$  denote the words, and  $|Q|$ ,  $|D|$ , and  $|A|$  are the number of words in each sequence. The word sequence  $A = \{a_1, a_2, \dots, a_{|A|}\}$  is a substring (subsequence occupies consecutive positions) in  $D$ , i.e.,  $s$  and  $e$  exist and satisfy

$$\{a_1, a_2, \dots, a_{|A|}\} = \{d_s, d_{s+1}, \dots, d_e\}, \text{ where } 0 < s \leq e < |D|. \quad (1)$$

We can solve this task by using an end-to-end model that takes  $Q$  and  $D$  as the input and  $A$  as the output. The model predicts the probability of a span  $S$  being the correct answer  $P(S|D, A)$  (We refer to  $P(S|D, A)$  as  $P(S)$  for brevity). The span  $S = \{d_s, d_{s+1}, \dots, d_e\}$  denotes a substring in  $D$ .

#### 2.1.1. Independent Assumption for the Start and End Positions

Since  $S$  could be an arbitrary span that holds  $s \leq e$ , the number of valid spans is quadratically related to  $|D|$  (e.g.,  $D$  has  $(|D|^2 + |D|)/2$  spans). Inferring the probabilities for all these spans is time-consuming and ineffective. Additionally, only one or a few

spans are the correct answers, so training models to learn from such imbalanced data is challenging. Therefore, precisely calculating the  $P(S)$  usually hinders such models from obtaining a state-of-the-art prediction accuracy [17,18]. Rajpurkar et al. [19] and Seo et al. [12] try to approximate  $P(S)$  by assuming that the start position  $s$  and the end position  $e$  are independent, which can also accelerate the computation:

$$P(S) = P(d_s, d_e) \approx P_{\text{start}}(d_s)P_{\text{end}}(d_e). \quad (2)$$

$d_s$  and  $d_e$  denote the start and end token of the span  $S$ .  $S$  can be uniquely identified by its boundary ( $s$  and  $e$ ), so  $P(d_s, d_e)$  equals  $P(S)$ , and words between  $d_s$  and  $d_e$  can be omitted from this formulation.  $P_{\text{start}}(d_s)$  denotes the probability of the  $s$ -th word in  $D$  being the start of the answer, and  $P_{\text{end}}(d_e)$  denotes that probability that the  $e$ -th word is the end.

With this approximation, a model  $\pi_\theta$  can be used to estimate the  $P_{\text{start}}(d_i)$  and  $P_{\text{end}}(d_e)$  for each word in  $D$  as  $\pi_{\theta, \text{start}}(d_i)$  and  $\pi_{\theta, \text{end}}(d_e)$  and then assemble them to get the predicted answer  $\tilde{A}$ :

$$\begin{aligned} \tilde{s}, \tilde{e} &= \underset{1 \leq i \leq |D|, i \leq j \leq |D|}{\operatorname{argmax}} (\pi_{\theta, \text{start}}(d_i) \pi_{\theta, \text{end}}(d_j)) \\ \tilde{A} &= \{d_{\tilde{s}}, \dots, d_{\tilde{e}}\}. \end{aligned} \quad (3)$$

The number of calculations is reduced to be linearly related to the document length  $|D|$ .

### 2.1.2. Greedy Search in the Multistep Decomposition

Although the independent assumption can reduce the computation load satisfactorily [8,14], it is somewhat counterintuitive, since these two positions cannot be independent. The start and end positions form the answer together, and if one varies, the other should also move to improvise a rational answer. Thus, Yang et al. [20] and Clark et al. [21] insisted on using a precise formula for determining the span's probability, but they decomposed it into multiple steps and used a greedy search to reduce the computational load. Usually, the end position is a conditional probability in the decomposition:

$$P(S) = P(d_s, d_e) = P_{\text{start}}(d_s)P_{\text{end}}(d_e | d_s). \quad (4)$$

Compared with Equation (2),  $P_{\text{end}}(d_e)$  becomes  $P_{\text{end}}(d_e | d_s)$  in Equation (4), indicating that the end position should depend on the start position. A model  $\pi_\theta$  can estimate the preceding probability  $P_{\text{start}}(d_i)$  over each word  $d_i \in D$  by picking the highest value as  $\tilde{s}$  and then finding the end position  $\tilde{e}$  based on  $\tilde{s}$ . The two steps are performed in a greedy manner:

$$\begin{aligned} \tilde{s} &= \underset{1 \leq i \leq |D|}{\operatorname{argmax}} (\pi_{\theta, \text{start}}(d_i)) \\ \tilde{e} &= \underset{\tilde{s} \leq j \leq |D|}{\operatorname{argmax}} (\pi_{\theta, \text{end}}(d_j | d_{\tilde{s}})). \end{aligned} \quad (5)$$

The independent (Equation (2)) and decomposition (Equation (4)) methods have similar levels of computational complexity and can achieve competitive performances. The decomposition method is more flexible, since its performance can be improved through the use of more complex search methods, such as the beam search [22]. It provides a more convenient way of balancing performance and efficiency. Moreover, Fajcik et al. [14] and Wang and Jiang [23] pointed out that independent methods may reduce the prediction accuracy, and therefore, the start and end positions should be considered jointly. Regarding approaches to predict the answer span, this paper explores another method to decompose the  $P(S)$ , which takes care of all tokens in the answer using an encoder-decoder model. Section 3 details the proposed approach.

### 2.1.3. Neural Network Models Used for Prediction

As introduced above, we need to predict  $P(d_s)$ ,  $P(d_e)$ , or  $P(d_e|d_s)$  to extract the answer span from  $D$ . Neural networks could be suitable choices with remarkable performance on this task [24]. Generally, neural-network-based extractive QA models consist of an encoder to transform each word into a feature vector and a classifier to compute the probability of each word (as start or end) based on its feature vector [8,12,13]:

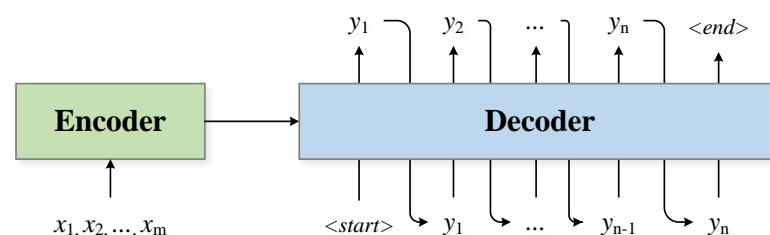
$$\begin{aligned} \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{|D|}\} &= \text{encoder}(d_1, d_2, \dots, d_{|D|}) \\ \pi_{\theta, \text{start}}(d_i) &= \text{classifier}_{\theta, \text{start}}(\mathbf{d}_i). \\ \pi_{\theta, \text{end}}(d_j) &= \begin{cases} \text{classifier}_{\theta, \text{end}}(\mathbf{d}_j) & \text{for } P(d_s) \text{ in Equation (2)} \\ \text{classifier}_{\theta, \text{end}}(\mathbf{d}_j, \mathbf{d}_{\tilde{s}}) & \text{for } P(d_j|d_{\tilde{s}}) \text{ Equation (4)} \end{cases} \end{aligned} \quad (6)$$

$\mathbf{d}_i$  and  $\mathbf{d}_j$  are the feature vectors for  $d_i$  and  $d_j$ , respectively. Commonly, the classifier is a multilayer perception with a softmax function that gives the final probability [25]. When calculating the end position with Equation (4), the classifier can take the features for both the predicted start word  $d_{\tilde{s}}$  and the candidate end word  $d_j$ . The encoder has more elaborate architectures like the Recurrent Neural Network (RNN) [26,27] or the Convolutional Neural Network (CNN) [28,29] to transform words into feature vectors effectively.  $\theta$  denotes the parameters of the model  $\pi_\theta$ . It can be optimized by minimizing a loss function on labeled data using the gradient descent.

### 2.2. Encoder-Decoder Models

This paper employs an encoder-decoder model to predict the answer span  $S$ . Encoder-decoder models are also known as sequence-to-sequence models, which can generate word sequences according to the input word sequences [30,31]. The input and output of the models, i.e., the sequence of words, are adaptable and can be used in many natural language generation tasks, such as dialogue systems [32,33], machine translation [34,35], text summarization [36,37], and knowledge graph completion [38,39].

Figure 2 illustrates an encoder-decoder model. The encoder transforms an input word sequence  $X = \{x_1, x_2, \dots, x_m\}$  into numerical features, similar to the encoder in Equation (6). These features are considered to capture the semantic information in  $X$ . The decoder generates the output sequence  $Y = \{y_1, y_2, \dots, y_n\}$  based on the features of  $X$  constructed by the encoder.



**Figure 2.** An encoder-decoder model autoregressively generates an output word sequence based on the input.  $\langle \text{start} \rangle$  and  $\langle \text{end} \rangle$  are the special tokens representing the generation's start and end.

Generally, an encoder-decoder model  $\pi_\theta$  estimates the conditional probability  $P(Y | X)$  with  $\pi_\theta(Y | X)$ . The number of output candidates is exponential, and it is impractical to enumerate them to find the one with the highest probability. Thus, the  $P(Y | X)$  is conditionally factorized for the decoder to handle it in an autoregressive manner:

$$P(Y | X) = P(y_1 | X)P(y_2 | X, y_1)P(y_3 | X, y_1, y_2) \cdots P(y_n | X, y_1, y_2, \dots, y_{n-1}). \quad (7)$$

The decoder can generate the output sequence in a word-by-word greedy manner. The generation of the  $i$ -th word depends upon the already generated words:

$$\tilde{y}_i = \underset{w \in \mathcal{V}}{\operatorname{argmax}}(\operatorname{decoder}(w \mid \tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_{i-1})). \quad (8)$$

$\mathcal{V}$  represents the vocabulary containing all of the candidate words. The process starts with a special “start of the sequence” token representing an empty generated sequence and stops if gets a special “end of the sequence” token. Additionally, advanced search algorithms in [22,40,41] can be used to improve the generation results.

### 2.3. Reinforcement Learning for Encoder-Decoder Models

Reinforcement Learning (RL) is learning to control an agent to accomplish an objective in the environment [42]. Usually, the agent does not learn, or barely learns, from examples that show which action is the best for the objective but requires the discovery of the superior actions by trial and error. RL uses reward functions  $r$  to quantify how well the agent accomplishes the objective. The agent’s goal is to maximize the rewards:

$$\mathbb{E}[r(\mathcal{A})\pi_{\theta}(\mathcal{A})], \quad (9)$$

where  $\mathcal{A}$  denotes actions, and  $\pi_{\theta}(\mathcal{A})$  determines the probability of choosing  $\mathcal{A}$ .  $r(\mathcal{A})$  is the reward obtained from the environment after taking  $\mathcal{A}$ .

A simple way to integrate RL with the encoder-decoder models is to consider the output sequence  $Y$  as actions  $\mathcal{A}$  and the input sequence  $X$  as the state [43]. Thus, Equation (9) becomes

$$\begin{aligned} & \mathbb{E}[r(Y)\pi_{\theta}(Y \mid X)] \\ &= [r(y_1, y_2, \dots, y_n)\pi_{\theta}(y_1, y_2, \dots, y_n \mid X)] \end{aligned} \quad (10)$$

$\pi_{\theta}$  denotes the encoder-decoder model here. We continue to use  $\pi_{\theta}$  to represent the model for convenience.  $\pi_{\theta}$  generates  $Y$  word-by-word and earns a reward when the generation is finished.  $\pi_{\theta}(Y \mid X)$  is the probability of generating the whole sequence  $Y$ . The selection of the reward function can be flexible. Task-specified evaluation methods, such as BLEU [44] for machine translation, ROUGE [45] for text summarization, or Diversity [46] for response generation, are suitable for the corresponding tasks.

The RL has been used in many NLP models. Ranzato et al. [47] proposed sequence-level training, which uses BLEU and ROUGE to calculate the reward for a generated sequence. Chen and Bansal [48] used RL to train a sentence selector in the proposed two-stage model, which selects salient sentences from the document and then summarizes the salient sentences to get the final summary. Li et al. [49] proposed the use of RL to help with the question generation task. Xiong et al. [50] defined an additional RL-based objective for the extractive QA model that independently predicts the answer span’s start and end positions. Hu et al. [51] improved the reward function used in [50], which involves overlap-based metrics and could neglect the acceptable answers. These works show the significant potential of using RL to help with the NLP. To our knowledge, the RL has not been evaluated in extractive QA models using the encoder-decoder framework. This paper fills this gap by providing empirical results. Other than extracting answers from textual evidence, locating answers over a knowledge graph [52] has been well-studied and applied in many real-world applications [53].

## 3. Methods

### 3.1. Modeling the Whole Answer Span Using the Encoder-Decoder Model

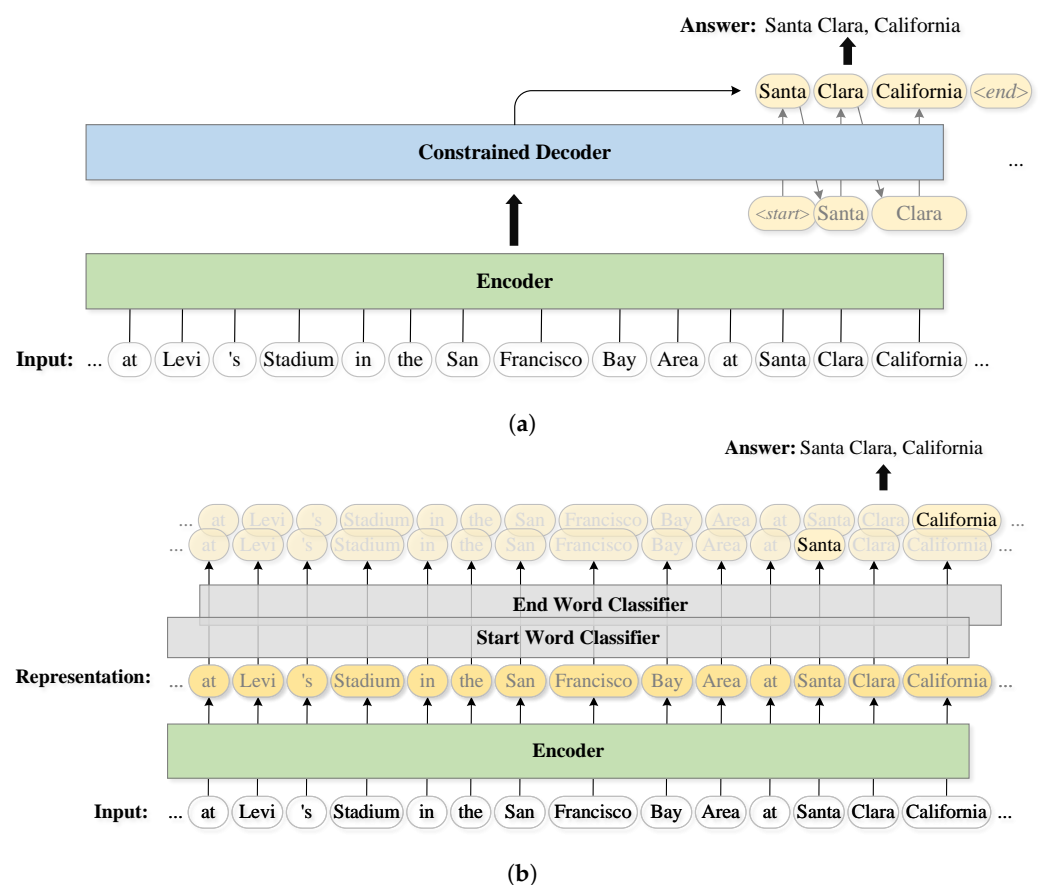
As introduced in Section 2, two kinds of approximation, independent assumption (Equation (2), described in Section 2.1.1), and greedy search (Equation (4) described in Section 2.1.2), have been developed for use in extractive QA. Each has advantages, e.g., independent assumption simplifies the training and inference process, and decompo-

sition requires no strong assumptions and can deal with the multistop problem with search algorithms.

However, both approaches can neglect some words in answers, i.e., the words between  $a_1$  and  $a_{|A|}$  in the answer span  $A = \{a_1, a_2, \dots, a_{|A|}\}$ . This paper proposes that the words between boundaries,  $\{a_2, \dots, a_{|A|-1}\}$  (called middle words for convenience) can also provide useful semantic information for QA models to learn the semantics of the answer, which can aid in the understanding of the question and document. By considering the middle words, the probability of a span  $S$  becomes

$$P(S) = P(d_s)P(d_{s+1}|d_s)P(d_{s+2}|d_s, d_{s+1}) \cdots P(d_e|d_s, d_{s+1}, \dots, d_{e-1}), \quad (11)$$

which is a word sequence that an encoder-decoder model can generate.  $s$  and  $e$  are the start and end positions. Thus, we introduce an encoder-decoder model for extractive QA, as shown in Figure 3a.



**Figure 3.** The comparison between the proposed model and a baseline model. The input question is “Which NFL team represented the NFC at Super Bowl 50?”. The answer is “Santa Clara California”. **(a)** The encoder-decoder model is used to solve the extractive QA task, taking advantage of all words in the answer. **(b)** The baseline extractive QA models use the start and end words only.

The encoder-decoder model is trained to predict all the words in the answers, estimating the  $P(S)$  following Equation (11). As a comparison, Figure 3b shows a traditional extractive QA model which ignores the middle words.

The loss function for generating an answer is [43]:

$$\mathcal{L}_{\text{text}} = - \sum_{i=1}^{|A|} \log \pi_{\theta}(a_i | a_1, \dots, a_{i-1}, D, Q), \quad (12)$$

where  $\pi_{\theta}$  denotes the encoder-decoder model.



We use BART-*base* as the backbone of the proposed encoder-decoder model [54]. The encoder in the BART-*base* consists of six transformer encoder layers, and the decoder has the same number of transformer decoder layers. Each transformer layer contains a self-attention layer and a feed-forward network [55]. The input is the concatenation of the question and the document sequences, and the output is the word sequence in the answer.

### 3.2. Constrained Decoding

The decoder usually considers all words in the vocabulary as candidates when generating each word, as shown in Equation (8). However, the extractive QA ensures that the output answer is a substring in the document, so we need to limit the output space of the decoder, which is usually referred to as constrained decoding [56,57]. We implement constrained decoding based on a trie tree, as shown in Algorithm 1.

---

#### Algorithm 1 Constrained Decoding.

---

**Input:** Question  $Q$  and Document  $D$ , Vocabulary  $\mathcal{V}$

**Input:** The decoder, Trie tree  $T$  and its functions:  $\text{add}(T, \dots)$ ,  $\text{search}(T, \dots)$

**Output:** Answer  $\tilde{A}$

```

1:  $\tilde{A} \leftarrow \{\}, T \leftarrow \Phi, i \leftarrow 0, \tilde{y}_0 \leftarrow \langle \text{start} \rangle$ 
2: for  $k \leftarrow 1$  to  $|D|$  do ▷ Initialize the trie tree  $T$ 
3:    $\text{add}(T, \{d_k, d_{k+1}, \dots, d_{|D|}\})$  ▷ Add a substring that starts with  $d_k$  into trie tree  $T$ .
4: end for
5: while  $\tilde{y}_i \neq \langle \text{end} \rangle$  do
6:    $\mathcal{V}_c \leftarrow \{\langle \text{end} \rangle\}$  ▷ Initialize the constrained vocabulary
7:    $\mathcal{P} \leftarrow \text{search}(T, \tilde{A})$  ▷ Obtain the substring starting with  $\tilde{A}$ 
8:   foreach  $\{p_1, p_2, \dots\} \in \mathcal{P}$  do ▷ Loop over each substring in  $D$  starting with  $\tilde{A}$ 
9:      $P = \{p_1, p_2, \dots\} - \tilde{A}$  ▷ Remove the prefix  $\tilde{A}$  from substring  $\{p_1, p_2, \dots\}$ 
10:     $\mathcal{V}_c \leftarrow \mathcal{V}_c + P_{[1]}$  ▷ Add the first token  $P_{[1]}$  in  $P$  into  $\mathcal{V}_c$ 
11:   end for
12:    $\tilde{y}_i = \underset{w \in \mathcal{V}_c}{\text{argmax}}(\text{decoder}(w \mid \tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_{i-1}))$ 
13:    $\tilde{A} \leftarrow \tilde{A} + \tilde{y}_i$  ▷ Save the predicted words
14: end while
15: return  $\tilde{A}$ 

```

---

$T$  is a trie tree, also known as a prefix tree. It is a tree data structure that can store and locate a set of sequences [58,59]. It groups the sequences based on their prefixes, and we can obtain the sequences that have a specified prefix from all the stored sequences. The function  $\text{search}(T, \dots)$  is shown in Algorithm 1. The  $\text{add}(T, \dots)$  function denotes the addition of a new sequence to  $T$ . We add all possible starts, as shown in Lines 2–3. The  $\text{search}(T, \dots)$  obtains the sequences that start with a prefix (We use the trie tree implemented in <https://github.com/pytries/marisa-trie>, accessed on 7 August 2021).  $\mathcal{V}_c$  is the constrained vocabulary that stores all of the candidate words and can ensure that the generation becomes invalid. It is updated at each generation step.

The constrained decoding required here is not trivial, since we need to take care of the tokenization [60], which can split a natural word into pieces with different indices to the origin word. The constrained decoder should not reject a generated word (piece) if its index differs from the expected index, but it should be checked carefully. Thus, in practice, we append every word  $w$  in the vocabulary  $\mathcal{V}$  after the current sequence, check whether the updated sequence satisfies the constraints, and save the valid word into the current constrained vocabulary  $\mathcal{V}_c$ .

### 3.3. Evaluation-Based Reinforcement Learning

Predicting a whole answer span is more challenging than predicting only its start and end words, since there are more labels to be predicted. Therefore, we introduce

evaluation-based reinforcement learning to help the encoder-decoder model. Wvaluation-based reinforcement learning is implemented as an auxiliary loss:

$$\mathcal{L}_{\text{RL}} = -\log \pi_{\theta}(\tilde{A}|D, Q)r(\tilde{A}, A). \quad (13)$$

Both the  $\mathcal{L}_{\text{text}}$  and  $\mathcal{L}_{\text{RL}}$  are optimized in training. The final loss function of the proposed model is  $\mathcal{L}_{\text{text}} + \mathcal{L}_{\text{RL}}$ . Minimizing this loss function encourages the model to maximize the reward. To make the optimization process more stable, we generate  $k$  candidate answers for each sample ( $Q$ ,  $D$ , and  $A$ ) and standardize the rewards within the same sample:

$$\hat{r}(\tilde{A}, A) = \frac{1}{k} \sum i^k \frac{r(\tilde{A}_i, A) - \text{mean}(r(\tilde{A}_1, A), \dots, r(\tilde{A}_k, A))}{\text{std}(r(\tilde{A}_1, A), \dots, r(\tilde{A}_k, A))}. \quad (14)$$

$\tilde{A}_i$  denotes the  $i$ -th answer sampled from  $\pi_{\theta}$ . We standardize the  $k$  raw rewards by subtracting their mean values from them and then dividing them by their standard variance. This standardization can help to ensure that the final rewards have both positive and negative values.

$r(\tilde{A}, A)$  is the reward obtained by comparing the predicted answer  $\tilde{A}$  with the ground truth answer  $A$ . We compute the reward using conventional metrics for the extractive QA, F1 and Exact Match (EM) [19]. We also test out the use of the ROUGE-L [45] for reward calculation in our experiment.

Specifically, EM is used to check whether the prediction is the same as the ground truth:

$$\text{EM}(\tilde{A}, A) = \begin{cases} 1 & \text{if the predicted sequence } \tilde{A} \text{ equals the ground truth } A, \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

Additionally, F1 is used to measure the word-level overlaps between the prediction and the ground truth:

$$\begin{aligned} \text{overlap}(\tilde{A}, A) &= \sum_{w \in \{A\} \cap \{\tilde{A}\}} \min(\text{count}(w, A), \text{count}(w, \tilde{A})) \\ \text{precision}(\tilde{A}, A) &= \frac{\text{overlap}(\tilde{A}, A)}{|\tilde{A}|} \\ \text{recall}(\tilde{A}, A) &= \frac{\text{overlap}(\tilde{A}, A)}{|A|} \\ \text{F1}(\tilde{A}, A) &= \frac{2 \times \text{precision}(\tilde{A}, A) \times \text{recall}(\tilde{A}, A)}{\text{precision}(\tilde{A}, A) + \text{recall}(\tilde{A}, A)}. \end{aligned} \quad (16)$$

In the above equation,  $w \in \{A\} \cap \{\tilde{A}\}$  denotes enumerating the words that exist in both  $A$  and  $\tilde{A}$ ; each word only counts once.  $\text{count}(w, A)$  is the number of times that word  $w$  appears in the sequence.  $|A|$  and  $|\tilde{A}|$  denote the lengths of sequences  $A$  and  $\tilde{A}$ , respectively.

The official evaluation script (<https://worksheets.codalab.org/rest/bundles/0x6b567e1cf2e041ec80d7098f031c5c9e/contents/blob/>, accessed on 21 February 2022) normalizes the answers before computing EM and F1, for example, by converting the answers into lower case and removing some stop words, punctuation, and extra spaces. ROUGE-L indicates the similarity between the prediction  $\tilde{A}$  and ground truth  $A$  based on their Longest Common Subsequence (LCS [61,62]):

$$\begin{aligned} \text{precision}_{\text{LCS}}(\tilde{A}, A) &= \frac{\text{LCS}(\tilde{A}, A)}{|\tilde{A}|} \\ \text{recall}_{\text{LCS}}(\tilde{A}, A) &= \frac{\text{LCS}(\tilde{A}, A)}{|A|} \\ \text{ROUGE}_L(\tilde{A}, A) &= \frac{(1 + \beta^2) \times \text{recall}_{\text{LCS}}(\tilde{A}, A) \times \text{precision}_{\text{LCS}}(\tilde{A}, A)}{\beta^2 \times \text{precision}_{\text{LCS}}(\tilde{A}, A) + \text{recall}_{\text{LCS}}(\tilde{A}, A)}, \end{aligned} \quad (17)$$



where  $LCS$  is the function that gives the LCS between the two input sequences,  $A$  and  $\tilde{A}$ .

## 4. Results and Discussion

### 4.1. Experiment Settings

We evaluated the proposed model on the SQuAD dataset [19], where each sample consists of a question, a document, and an annotated answer. The documents were obtained from Wikipedia (<https://www.wikipedia.org/>, accessed on 16 June 2016) paragraphs. Table 1 shows two examples from the SQuAD dataset. The ground-truth answers are marked in bold and the corresponding clues are presented in blue.

**Table 1.** Samples from the SQuAD dataset. An extractive QA model needs to understand the natural language question and the evidence in the document to find the answer span from the document \*.

NO.	Question	Document	Answer
1	Which NFL team represented the AFC at Super Bowl 50?	Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The <b>American Football Conference (AFC)</b> champion <b>Denver Broncos</b> defeated the National Football Conference (NFC) champion Carolina Panthers 24-10 to earn their third Super Bowl title...	Denver Broncos
2	Who was in charge of the papal army in the War of Barbastro?	The legendary religious zeal of the Normans was exercised in religious wars long before the First Crusade carved out a Norman principality in Antioch. They were major foreign participants in the Reconquista in Iberia. In 1018, Roger de Tosny traveled to the Iberian Peninsula to carve out a state for himself from Moorish lands, but he failed. In 1064, <b>during the War of Barbastro, William of Montreuil led the papal army...</b>	William of Montreuil

\* The ground-truth answers are in bold, and the corresponding clues are in blue.

There were 87,599 samples in the training set and 10,570 samples in the validation set. We used the validation set for both validation and testing. We trained each model for 100,000 steps in total, saved checkpoints every 10,000 steps, and chose the best checkpoint using the results of the validation set. Table 2 shows the other hyperparameters used.

**Table 2.** The Hyperparameters.

Hyperparameter	Value	Description
Batch size	32	Number of Samples in each Batch
Learning Rate (LR)	$5 \times 10^{-5}$	Coefficient for updating the parameters
LR scheduler	Linear warmup	Tune the LR as the training step increases <sup>1</sup>
LR warmup steps	500	The parameter for LR scheduler
Optimizer	AdamW	Adamw optimizer provided by Pytorch <sup>2</sup>
Weight Decay	0.01	Coefficient for scaling the parameters down
Betas	0.9, 0.999	Coefficients used for computing running averages of gradient and its square <sup>3</sup>
$k$	4	Sampled sequences in Equation (14)
beam size	4	The number of beams for the beam search

<sup>1</sup> [https://huggingface.co/docs/transformers/v4.26.1/en/main\\_classes/optimizer\\_schedules#transformers.get\\_linear\\_schedule\\_with\\_warmup](https://huggingface.co/docs/transformers/v4.26.1/en/main_classes/optimizer_schedules#transformers.get_linear_schedule_with_warmup), accessed on 20 February 2023. <sup>2</sup> <https://pytorch.org/>, accessed on 20 February 2023. <sup>3</sup> <https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html>, accessed on 20 February 2023.

### 4.2. Main Results

Table 3 shows the experiment results. The baseline models include the following:

1. **BiDAF [12]**: a classical extractive QA model that uses bidirectional attention flow (question-to-document and document-to-question attention) to enrich the representation of words. BiDAF predicts the answers' start and end positions independently according to the representations.
2. **BiDAF\w compound (best) [14]**: uses the approaches proposed in [14] that jointly predict the start and end positions to enhance the BiDAF model. The best result is reported.
3. **DCN [63]**: locates the answer spans by iteratively predicting the start and end positions to overcome the initial local maxima, which may lead to the wrong answers.
4. **DCN+ [50]**: introduces reinforcement learning techniques to optimize the F1 metric for extractive QA directly.
5. **R.M-Reader [51]**: a memory-based model that uses reinforcement learning with a reward function refined for better coverage.
6. **BERT-base [8]**: an extractive QA model based on a powerful pretrained language model. We downloaded the model from <https://huggingface.co/csarron/bert-base-uncased-squad-v1/tree/main> (accessed on 20 February 2023) and evaluated it locally.
7. **BERT-base\w compound (best) [14]**: jointly predicts the start and end positions. It is similar to Model 2: **BiDAF\w compound (best)**.
8. **BART-base**: directly trains a BART-base model to generate the whole answer based on the question and answer.

The following models are trained with the proposed approaches. “RL\w F1”, “RL\w ROUGE-L”, and “RL\w EM&F1” represent the BART-base models that are trained using the reinforcement learning loss with the F1 rewards (Equation (16)), ROUGE-L (Equation (17)), and the sum of the F1 and EM values (Equations (16) and (15)), respectively. “Constrained” denotes the use of constrained decoding for answer generation.

**Table 3.** The experimental results on the SQuAD dataset.

No.	Model	EM	F1	#Out of Document
1	BiDAF [12]	66.16	76.19	0
2	\w compound (best) [14]	66.96	75.90	0
3	DCN [63]	65.4	75.6	0
4	DCN+ [50]	74.5	83.1	0
5	R.M-Reader [51]	78.9	86.3	0
6	BERT-base[8]	80.92	88.24	0
7	\w compound (best) [14]	81.83	88.52	0
8	BART-base	78.10	87.17	410
9	BART-base\w Constrained	79.80	88.05	0
10	RL\w EM&F1	78.37	87.87	329
11	RL\w EM&F1 Constrained	79.84	88.39	0
12	RL\w F1	78.83	88.04	310
13	RL\w F1 Constrained	80.02	88.54	0
14	RL\w ROUGE-L	78.27	87.44	304
15	RL\w ROUGE-L Constrained	79.39	87.97	0

As shown in Table 3, we can see that the proposed method, Model 13, can achieve better F1 results than the baseline models. Model 13 uses the F1 score as the reward function for reinforcement learning and constrained decoding. Model 13 outperforms the models that also use reinforcement learning for extractive QA and works slightly better than the model that also jointly models the start and end positions, showing the potential for considering the whole answer span in extractive QA.

The use of metrics as rewards enables the models to be directly optimized based on the metrics. However, Model 10 uses all of the metrics, EM and F1, as rewards but does not achieve better results, indicating that the discrete EM value (0 or 1) may not be

suitable for a reward function. Again, we emphasize that the reward functions should be chosen carefully.

The “#Out of Document” shows the number of the predicted answer spans that are not a substring of the input document, violating the settings of extractive QA. We can see that there are always some invalid predictions without constrained decoding. This means that the encoder-decoder model used, BART-*base*, still cannot thoroughly learn the input–output paradigm of extractive QA. Generally, we find that the constrained decoding strategy always brings a notable improvement to the EM score but does not improve F1 as much. Meanwhile, the proposed evaluation-based reinforcement learning method is always helpful for improving the performance, even when the ROUGE-L is used as the reward. ROUGE-L does not directly correspond with the evaluation metrics and is slightly different from EM and F1 in terms of the calculation formula. The results again demonstrate the effectiveness of the combination of the RL and text generation.

We set different beam sizes for the proposed encoder-decoder model and obtained similar results, as shown in Table 4. One possible reason for this is that the generated sequence is short, and the search space for generation is small, so the search algorithm is limited. We set the beam size to 4 to obtain the results presented in Table 3.

**Table 4.** The experiment results with different beam sizes.

Model	EM	F1	Beam Size
BART- <i>base</i>	77.83	87.15	1
	78.10	87.17	4
	77.98	87.18	16
BART- <i>base</i> RL\w EM&F1	77.92	87.51	1
	78.37	87.87	4
	78.09	87.81	16
BART- <i>base</i> RL\w F1	78.43	87.77	1
	78.83	88.04	4
	78.61	88.01	16
BART- <i>base</i> RL\w ROUGE-L	77.77	87.08	1
	78.27	87.44	4
	78.14	87.41	16

#### 4.3. Case Study and Discussion

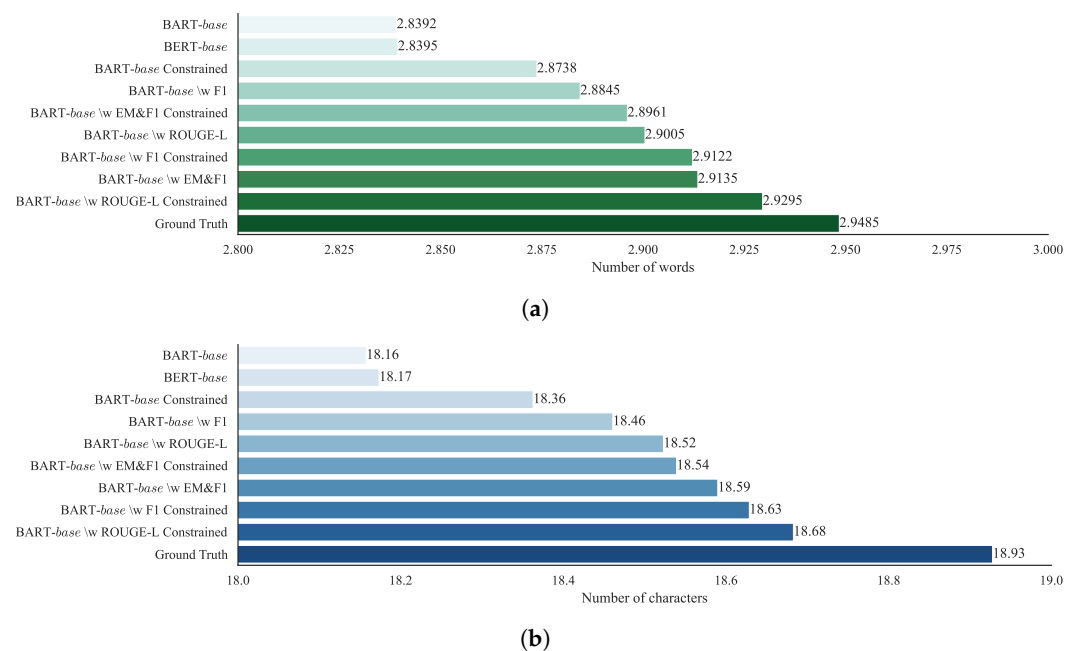
Table 5 presents a case study to show the improvement qualitatively. The proposed model is compared with a strong baseline model BERT on the SQuAD datasets. Question 1 asks about the allies of Normans in the war. The baseline model gives their opponent, whereas the proposed model yields the correct answer, demonstrating that it understands the document correctly. Question 2 explicitly asks for the complete date, including the year, month, and day, but the baseline provides the month only and omits the others.

Based on these cases, the proposed model seems to be better at producing longer answers. To verify this assumption quantitatively, we investigated the lengths of the answers predicted by different models. Figure 4 shows the average number of words/characters in answers, respectively. The horizontal axis shows the answer’s length, and the vertical axis displays the models.

**Table 5.** A case study for BERT-base (Baseline) and BART-base RL\w F1 (Our Model) on SQuAD dataset \*.

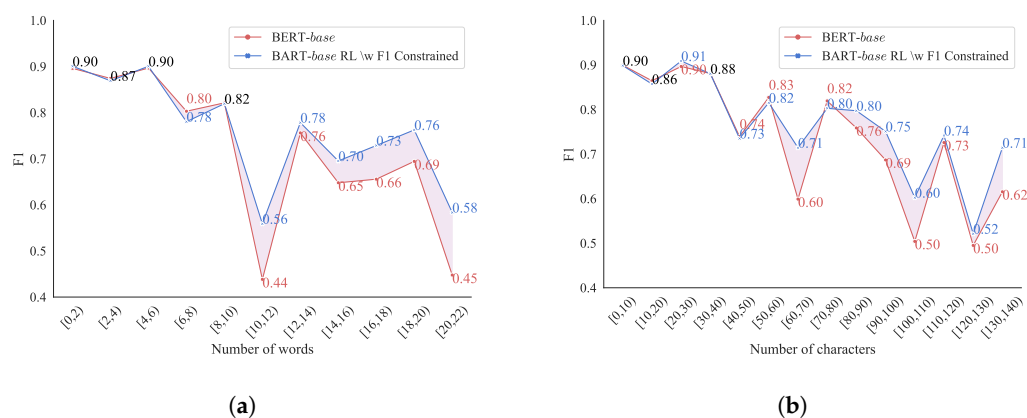
NO.	Question	Document	Predictions
1	Who did the Normans team up with in Anatolia?	Some <b>Normans</b> joined <b>Turkish</b> forces to aid in the destruction of the <b>Armenians</b> vassal-states of Sassoun and Taron in far eastern Anatolia. Later, many took up service with...	Baseline: Armenians Our Model: Turkish forces
2	What month, day, and year did the Super Bowl 50 take place?	Super Bowl 50 was an American football game used to determine the champion of the National Football League... The game <b>was played on 7 February 2016</b> at Levi's Stadium in the...	Baseline: February Our Model: 7 February 2016

\* The ground-truth answers are in bold, and the corresponding clues are in blue.

**Figure 4.** The average length of the answers predicted by the models. (a) The numbers of words in the answers. (b) The numbers of characters in the answers.

We sorted the models by the answers' lengths in ascending order. We can see that the ground-truth answers were the longest. The plain encoder-decoder model BART-base and the baseline model BERT-base had the shortest average lengths, demonstrating that they cannot perform ideally with long answers. The models equipped with the proposed methods, constrained decoding and evaluation-based RL (denoted by Constrained and RL\w, respectively), prefer to give longer answers that are closer to the ground-truth's length.

However, generating longer answers does not mean generating better answers, and we need to evaluate their quality with the corresponding metrics. To analyze how the answer length affects the models' prediction results, we correlated the length of the ground-truth answer with the F1 score of the answer predictions, as shown in Figure 5.



**Figure 5.** The F1 scores of the predictions as the answer gets longer. (a) Grouped by numbers of words. (b) Grouped by numbers of characters.

We grouped questions with similar answer lengths in the same range together and averaged the F1 scores of the predictions for the questions in the same group. The horizontal axis shows the length ranges, and the vertical axis denotes the F1 scores. Overall, the performance degraded as the target answer became longer. Notably, the proposed model (BART-base RL \w F1 Constrained) and the baseline model (BERT-base) performed similarly when the answers were shorter (e.g., less than ten words). In comparison, the performance of the proposed model was much better than the baseline model for longer answers. This demonstrates that the proposed model generates longer answers and performs better when the ground truth is longer, which also reveals that the improvement brought by the proposed model is mainly for longer answers.

## 5. Conclusions

This paper proposes an encoder-decoder model for extractive QA. The proposed encoder-decoder model can predict all the words in the answer span, which differs from the previous extractive QA models, which only predict the start and end positions. Thus, the extractive QA task runs in a more natural way, and the model should give a precise answer, rather than answer pieces. We additionally introduced reinforcement learning to create an auxiliary objective based on evaluation metrics to assist the models in training. We evaluated the proposed method on the SQuAD dataset. The experiment results show that the proposed encoder-decoder model can achieve competitive results to state-of-the-art baseline models, showing the potential of the encoder-decoder models that use reinforcement learning and unifying the NLP tasks with the encoder-decoder model.

**Author Contributions:** Conceptualization, S.L., C.S., B.L., Y.L. and Z.J.; methodology, S.L., C.S., B.L., Y.L. and Z.J.; software, S.L.; validation, C.S., B.L., Y.L. and Z.J.; formal analysis, S.L.; investigation, S.L.; resources, S.L.; data curation, C.S., B.L., Y.L. and Z.J.; writing—original draft preparation, S.L.; writing—review and editing, C.S., B.L., Y.L. and Z.J.; visualization, S.L.; supervision, C.S., B.L. and Z.J.; project administration, C.S., B.L., Y.L. and Z.J.; funding acquisition, C.S., B.L., Y.L. and Z.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** Supported by the National Key Research and Development Project (2021YFF0901600), National Natural Science Foundation of China (62176074), Interdisciplinary Development Program of Harbin Institute of Technology (No. SYL-JC-202203), and the Fundamental Research Funds for the Central Universities (project number: 2022FRFK0600XX).

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Gupta, P.; Gupta, V. A Survey of Text Question Answering Techniques. *Int. J. Comput. Appl.* **2012**, *53*, 1–8. [\[CrossRef\]](#)
- Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Brussels, Belgium, 1 November 2018; pp. 353–355. [\[CrossRef\]](#)
- Mitra, B.; Craswell, N. An Introduction to Neural Information Retrieval. *Found. Trends Inf. Retr.* **2018**, *13*, 1–126. [\[CrossRef\]](#)
- Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 632–642. [\[CrossRef\]](#)
- Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring Massive Multitask Language Understanding. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021.
- Fan, A.; Jernite, Y.; Perez, E.; Grangier, D.; Weston, J.; Auli, M. ELI5: Long Form Question Answering. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019; Volume 1: Long Papers, pp. 3558–3567. [\[CrossRef\]](#)
- Wang, L.; Zheng, K.; Qian, L.; Li, S. A Survey of Extractive Question Answering. In Proceedings of the 2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS), Tianjin, China, 10–11 December 2022; pp. 147–153. [\[CrossRef\]](#)
- Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; Volume 1 (Long and Short Papers), pp. 4171–4186. [\[CrossRef\]](#)
- Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
- Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
- Yamada, I.; Asai, A.; Shindo, H.; Takeda, H.; Matsumoto, Y. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020; pp. 6442–6454. [\[CrossRef\]](#)
- Seo, M.J.; Kembhavi, A.; Farhadi, A.; Hajishirzi, H. Bidirectional Attention Flow for Machine Comprehension. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
- Yu, A.W.; Dohan, D.; Luong, M.; Zhao, R.; Chen, K.; Norouzi, M.; Le, Q.V. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
- Fajcik, M.; Jon, J.; Smrz, P. Rethinking the Objectives of Extractive Question Answering. In Proceedings of the 3rd Workshop on Machine Reading for Question Answering, Online, 10 November 2021; pp. 14–27. [\[CrossRef\]](#)
- Chen, D. Neural Reading Comprehension and Beyond. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2018.
- Liu, S.; Zhang, X.; Zhang, S.; Wang, H.; Zhang, W. Neural Machine Reading Comprehension: Methods and Trends. *Appl. Sci.* **2019**, *9*, 3698. [\[CrossRef\]](#)
- Lee, K.; Kwiatkowski, T.; Parikh, A.P.; Das, D. Learning Recurrent Span Representations for Extractive Question Answering. *arXiv* **2016**, arXiv:1611.01436.
- Lee, J.; Sung, M.; Kang, J.; Chen, D. Learning Dense Representations of Phrases at Scale. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 5–6 August 2021; Volume 1: Long Papers. [\[CrossRef\]](#)
- Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–4 November 2016. . [\[CrossRef\]](#)
- Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.G.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 5754–5764.
- Clark, K.; Luong, M.; Le, Q.V.; Manning, C.D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
- Dahlmeier, D.; Ng, H.T. A Beam-Search Decoder for Grammatical Error Correction. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, Jeju Island, Republic of Korea, 12–14 July 2012; pp. 568–578.
- Wang, S.; Jiang, J. Machine Comprehension Using Match-LSTM and Answer Pointer. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.



24. Richard, M.D.; Lippmann, R.P. Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities. *Neural Comput.* **1991**, *3*, 461–483. [CrossRef] [PubMed]
25. Rosenblatt, F. Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. Technical Report. 1961. Available online: <https://apps.dtic.mil/sti/pdfs/AD0256582.pdf> (accessed on 20 February 2023). [CrossRef]
26. Salehinejad, H.; Baarbe, J.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent Advances in Recurrent Neural Networks. *arXiv* **2018**, arXiv:1801.01078.
27. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
28. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [CrossRef] [PubMed]
29. Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; Dauphin, Y.N. Convolutional Sequence to Sequence Learning. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1243–1252.
30. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
31. Cho, K.; van Merriënboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
32. Wu, Y.; Wu, W.; Yang, D.; Xu, C.; Li, Z. Neural Response Generation With Dynamic Vocabularies. *Proc. Aaai Conf. Artif. Intell.* **2018**, *32*, 5594–5601. [CrossRef]
33. Li, S.; Sun, C.; Xu, Z.; Tiwari, P.; Liu, B.; Gupta, D.; Shankar, K.; Ji, Z.; Wang, M. Toward Explainable Dialogue System Using Two-stage Response Generation. *ACM Trans. Asian-Low-Resour. Lang. Inf. Process.* **2023**, *22*, 1–18. [CrossRef]
34. He, X.; Haffari, G.; Norouzi, M. Sequence to Sequence Mixture Model for Diverse Machine Translation. In Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, 31 October–1 November 2018; pp. 583–592.
35. Neubig, G. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. *arXiv* **2017**, arXiv:1703.01619.
36. Nallapati, R.; Zhou, B.; dos Santos, C.N.; Gülçehre, Ç.; Xiang, B. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, 11–12 August 2016; pp. 280–290. [CrossRef]
37. See, A.; Liu, P.J.; Manning, C.D. Get To The Point: Summarization with Pointer-Generator Networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1: Long Papers, pp. 1073–1083.
38. Xie, X.; Zhang, N.; Li, Z.; Deng, S.; Chen, H.; Xiong, F.; Chen, M.; Chen, H. From Discrimination to Generation: Knowledge Graph Completion with Generative Transformer. In Proceedings of the Companion of The Web Conference 2022, Virtual Event/Lyon, France, 25–29 April 2022; pp. 162–165. [CrossRef]
39. Ye, H.; Zhang, N.; Chen, H.; Chen, H. Generative Knowledge Graph Construction: A Review. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 1–17.
40. Pascual, D.; Egressy, B.; Bolli, F.; Wattenhofer, R. Directed Beam Search: Plug-and-Play Lexically Constrained Language Generation. *arXiv* **2020**, arXiv:2012.15416.
41. Vijayakumar, A.K.; Cogswell, M.; Selvaraju, R.R.; Sun, Q.; Lee, S.; Crandall, D.J.; Batra, D. Diverse Beam Search for Improved Description of Complex Scenes. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 7371–7379.
42. Sutton, R.; Barto, A. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Netw.* **1998**, *9*, 1054. [CrossRef]
43. Keneshloo, Y.; Shi, T.; Ramakrishnan, N.; Reddy, C.K. Deep Reinforcement Learning for Sequence-to-Sequence Models. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 2469–2489. [CrossRef] [PubMed]
44. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. Bleu: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 6–12 July 2002; pp. 311–318. [CrossRef]
45. Lin, C.Y.; Hovy, E. Manual and automatic evaluation of summaries. In Proceedings of the ACL-02 Workshop on Automatic Summarization, Philadelphia, USA, 11–12 July 2002.
46. Li, J.; Galley, M.; Brockett, C.; Gao, J.; Dolan, B. A Diversity-Promoting Objective Function for Neural Conversation Models. In Proceedings of the NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 110–119. [CrossRef]
47. Ranzato, M.; Chopra, S.; Auli, M.; Zaremba, W. Sequence Level Training with Recurrent Neural Networks. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016.

48. Chen, Y.; Bansal, M. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, 15–20 July 2018; Volume 1: Long Papers, pp. 675–686.
49. Li, X.; Huang, Z.; Liu, F.; Wang, C.; Hu, M.; Xu, S.; Peng, Y. RAD: Reinforced Attention Decoder Model On Question Generation. In Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, UK, 19–24 July 2020; pp. 1–8. [\[CrossRef\]](#)
50. Xiong, C.; Zhong, V.; Socher, R. DCN+: Mixed Objective And Deep Residual Coattention for Question Answering. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
51. Hu, M.; Peng, Y.; Huang, Z.; Qiu, X.; Wei, F.; Zhou, M. Reinforced Mnemonic Reader for Machine Reading Comprehension. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; pp. 4099–4106. [\[CrossRef\]](#)
52. Saxena, A.; Kochsiek, A.; Gemulla, R. Sequence-to-Sequence Knowledge Graph Completion and Question Answering. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022, Dublin, Ireland, 22–27 May 2022; Volume 1: Long Papers, pp. 2814–2828. [\[CrossRef\]](#)
53. Ngai, E.W.; Lee, M.C.; Luo, M.; Chan, P.S.; Liang, T. An intelligent knowledge-based chatbot for customer service. *Electron. Commer. Res. Appl.* **2021**, *50*, 101098. [\[CrossRef\]](#)
54. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020; pp. 7871–7880.
55. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
56. Hokamp, C.; Liu, Q. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1: Long Papers, pp. 1535–1546.
57. Post, M.; Vilar, D. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, LA, USA, 1–6 June 2018; Volume 1 (Long Papers), pp. 1314–1324.
58. McClellan, M.T.; Minker, J.; Knuth, D.E. The Art of Computer Programming, Vol. 3: Sorting and Searching. *Math. Comput.* **1974**, *28*, 1175. [\[CrossRef\]](#)
59. Jankowski, R. Advanced data structures by Peter Brass Cambridge University Press 2008. *ACM SIGACT News* **2010**, *41*, 19–20. [\[CrossRef\]](#)
60. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners. 2019. Available online: <https://paperswithcode.com/paper/language-models-are-unsupervised-multitask> (accessed on 20 February 2023 ).
61. Maier, D. The Complexity of Some Problems on Subsequences and Supersequences. *J. ACM* **1978**, *25*, 322–336. [\[CrossRef\]](#)
62. Bergroth, L.; Hakonen, H.; Raita, T. A Survey of Longest Common Subsequence Algorithms. In Proceedings of the Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000, A Coruña, Spain, 27–29 September 2000; pp. 39–48. [\[CrossRef\]](#)
63. Xiong, C.; Zhong, V.; Socher, R. Dynamic Coattention Networks For Question Answering. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.