Open-Domain Why-Question Answering with Adversarial Learning to Encode Answer Texts

Jong-Hoon Oh[§] **Kazuma Kadowaki**[§‡] **Julien Kloetzer**[§] **Ryu Iida**[§¶] **Kentaro Torisawa**[§¶]

Data-driven Intelligent System Research Center (DIRECT),
National Institute of Information and Communications Technology (NICT)[§]
Advanced Technology Laboratory, The Japan Research Institute, Limited (JRI)[‡]
Graduate School of Science and Technology, NAIST[¶]
{rovellia, kadowaki, julien, ryu.iida, torisawa}@nict.go.jp

Abstract

In this paper, we propose a method for why-question answering (why-QA) that uses an adversarial learning framework. Existing why-QA methods retrieve *answer passages* that usually consist of several sentences. These multi-sentence passages contain not only the reason sought by a why-question and its connection to the why-question, but also redundant and/or unrelated parts. We use our proposed *Adversarial networks for Generating compact-answer Representation* (AGR) to generate from a passage a vector representation of the non-redundant reason sought by a why-question and exploit the representation for judging whether the passage actually answers the why-question. Through a series of experiments using Japanese why-QA datasets, we show that these representations improve the performance of our why-QA neural model as well as that of a BERT-based why-QA model. We show that they also improve a state-of-the-art distantly supervised open-domain QA (DS-QA) method on publicly available English datasets, even though the target task is not a why-QA.

1 Introduction

Why-question answering (why-QA) tasks retrieve from a text archive answers to such why-questions as “Why does honey last such a long time?” Previous why-QA methods retrieve from a text archive *answer passages*, each of which consists of several sentences, like A in Table 1 (Girju, 2003; Higashinaka and Isozaki, 2008; Oh et al., 2012, 2013, 2016, 2017; dos Santos et al., 2016; Sharp et al., 2016; Tan et al., 2016; Verberne et al., 2011), and then determine whether the passages answer the question. A proper answer passage must contain (1) a paraphrase of the why-question (e.g., the underlined texts in Table 1) and (2) the *reasons* or the *causes* (e.g., the bold texts in Table 1) of the events described in the why-question, both of which are often written in multiple non-adjacent sentences. This multi-sentenceness implies that the answer passages often contain redundant parts that are not directly related to a why-question or its reason/cause and whose presence complicates the why-QA task. Highly accurate why-QA methods should be able to find the exact reason sought by a why-question in an answer passage without being distracted by redundancy.

Q	Why does honey last a long time?
A	While excavating Egypt’s pyramids, archaeologists have found pots of honey in an ancient tomb: thousands of years old and still preserved. <u>Honey can last a long time</u> **due to three special properties.** Its average pH is 3.9, which is quite acidic. **Such high level of acidity** is certainly hostile and **hinders the growth of many microbes**. Though honey contains around 17–18% water, **its water activity is too low to support the growth of microbes**. Moreover **honey contains hydrogen peroxide**, which is thought to help **prevent the growth of microbes in honey**. Despite these properties, honey can be contaminated under certain circumstances.
C	Because its acidity, low water activity, and hydrogen peroxide together hinder the growth of microbes.

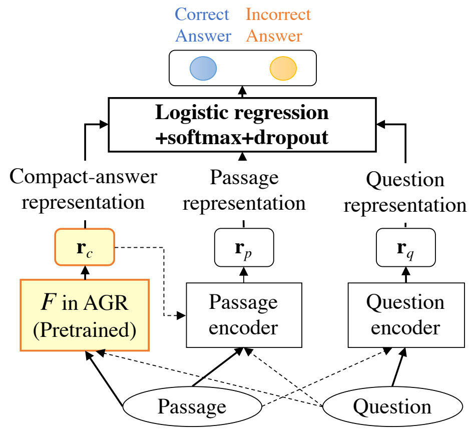
Table 1: Answer passage A to why-question Q and its compact answer C

In this paper, we train a neural network (NN) to generate, from an answer passage, a vector representation of the non-redundant reason asked by a why-question, and exploit the generated vector representation as evidence for judging whether the passage answers the why-question. This idea was inspired by Ishida et al. (2018), who used a seq2seq model to automatically generate such *compact answers* as C in Table 1 from the answer passages retrieved by a why-QA method. Compact answers are sentences or phrases that express the reasons for a given why-question without redundancy. If we can use such automatically generated compact-answers to support a why-QA method in finding the exact reason of a why-question in these passages, why-QA accuracy may be improved. We actually tried this idea in a preliminary study in which we generated a compact answer from a given question-passage pair by using the compact-answer generation method of Iida et al. (2019) and used the generated compact-answer along with the given question-passage pair to find proper answer passages. However, we were disappointed by the small performance improvement, as shown in our experimental results.

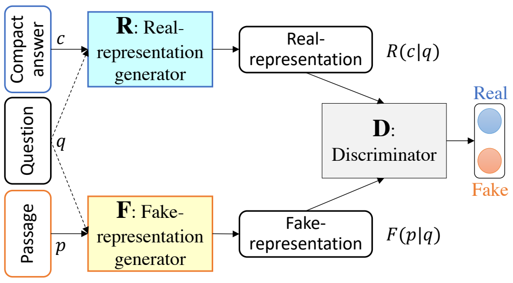
We chose an alternative approach. Instead of generating a *compact answer* of an answer passage as word sequences, we devised a model to generate a *compact-answer representation*, which is a vector representation for a compact answer, from an answer passage. Inspired by the generative adversarial network (GAN) approach (Goodfellow et al., 2014), we developed an adversarial network called the *Adversarial networks for Generating compact-answer Representation* (AGR). Like the original GAN, an AGR is composed of a generator and a discriminator: the *generator network* is trained for generating (from answer passages) *fake representations* to make it hard for the *discriminator network* to distinguish these fake representations from the *true representations* derived from manually created compact-answers.

We combined the generator network in the AGR with an extension of the state-of-the-art why-QA method (Oh et al., 2017). Our evaluation against a Japanese open-domain why-QA dataset, which was created using general web texts as a source of answer passages, revealed that the generator network significantly improved the accuracy of the top-ranked answer passages and that the combination significantly outperformed several strong baselines, including a combination of a generator network and a BERT model (Devlin et al., 2019). This combination also outperformed a vanilla BERT model, suggesting that the generator network in our AGR may be effective even if it is combined with many types of NN architectures. Another interesting point is that the performance improved even when we replaced, as the inputs to AGR, the word embedding vectors that represent an answer passage, with a *random vector*. This observation warrants further exploration in our future work.

Finally, we applied our AGR to a distantly supervised open-domain QA (DS-QA) task (Chen et al., 2017), which is an extension of a machine-reading task, to check whether it is applicable to other datasets. We combined our generator network with a state-of-the-art DS-QA method, OpenQA (Lin et al., 2018), and used a generated compact-answer representation from a given passage as evidence to 1) select relevant passages from the retrieved ones and 2) find an answer from the selected passages. Although the task was not our initial target (why-QA) and the answers in the DS-QA task were considerably shorter than those in the why-QA, experiments using three publicly available datasets (Quasar-T (Dhingra et al., 2017), SearchQA (Dunn et al., 2017), and TriviaQA (Joshi et al., 2017)) revealed that the generator network improved the performance in most cases. This suggests that AGR may be applicable to many QA-like tasks.



(a) Our why-QA model



(b) AGR and its three subnetworks F, R, and D

Figure 1: System architecture

2 Why-QA Model

Figure 1 illustrates the architecture of our why-QA model and the AGR. Our why-QA model computes the probability that a given answer passage describes a proper answer to a given why-question using the representations of a question, an answer passage, and a compact answer. The probability (the why-QA model's final output) is computed from these representations by our answer selection module, which is a logistic regression layer with dropout and softmax output.

The representations of why-questions and answer passages are generated by Convolutional Neural Networks (CNNs) (Collobert et al., 2011; LeCun et al., 1998) that (1) are augmented by two types of attention mechanisms, *similarity-attention* (dos Santos et al., 2016; Tan et al., 2016) and *causality-attention* (Oh et al., 2017), and (2) are given two types of word embeddings, *general word embeddings* computed by *word2vec* (Mikolov et al., 2013) using Wikipedia and *causal word embeddings* (Sharp et al., 2016). Note that in computing a question's representation, the answer passage is given to the question encoder to guide the computation. Likewise the passage encoder is given the question and the representation of the compact answer. We represent these information flows with dotted arrows in Fig. 1(a).

The representations of compact answers are created by a generator network called a *fake-representation generator* (F in Fig. 1(a)), which is pre-trained in an adversarial learning manner (Fig. 1(b)). During the training of the whole why-QA model, the generator's parameters are fixed and no further fine-tuning is conducted.

In the next section, we describe our main contribution: the AGR and the fake-representation generator. The entire why-QA model can be seen as an extension of the state-of-the-art why-QA method (Oh et al., 2017). Its details are described in Section A of the supplementary materials.

3 Adversarial Networks for Generating Compact-answer Representation

3.1 Adversarial training

Generative adversarial networks (GANs) (Goodfellow et al., 2014) are a framework for training generative models based on game theory. Unlike the original GANs, which generate such *data samples* as images and compact answers from noise, our AGR generates *useful vector representations* from *meaningful text passages*. To clarify the difference, we explain our AGR with three subnetworks: two generators, F and R, and a discriminator, D, as in Fig. 1(b). Generator F takes as input passage p drawn from prior passage distribution d_p and outputs vector $\bar{\mathbf{p}}$ as a *fake representation* of a compact answer. We call F a *fake-representation generator*. R, which we call a *real-representation generator*, is given sample c taken from manually created compact-answers and provides vector $\bar{\mathbf{c}}$ as a *real-representation* of the sampled compact-answer. Discriminator D has to distinguish fake-representation $\bar{\mathbf{p}}$ from real-representation $\bar{\mathbf{c}}$ of a compact answer. These three networks play an adversarial minimax game; fake-representation generator F creates a fake compact-answer representation that is hard for the discriminator to distinguish from the representations of manually created compact-answers, and discriminator D and generator R simultaneously try to avoid being duped by generator F. These processes should allow generator F to learn how to generate a representation of a proper compact-answer from an answer passage.

In addition, since passage p and compact answer c are dependent on question q, the generation of the compact-answer representations by F and R is conditioned by question q, like in the conditional GANs (Mirza and Osindero, 2014). We trained our AGR with the following minimax objective:

$$\min_{F} \max_{D,R} V(D,F,R) = \mathbb{E}_{c\sim d_c(c)}[\log D(R(c|q))] + \mathbb{E}_{p\sim d_p(p)}[\log(1 - D(F(p|q))].$$

3.2 Generator and discriminator

In our implementation, both F and R are networks with identical structure called Encoder. They are defined as follows, where p, c, and q are respectively an answer passage, a manually created compact-answer, and a why-question:

$$F(p|q) = \text{Encoder}(p; \theta_F, q)$$
$$R(c|q) = \text{Encoder}(c; \theta_R, q)$$

Here θ_F and θ_R represent the parameters of networks F and R. The details of Encoder are described below.

Discriminator $D(\mathbf{r})$ takes as input $\mathbf{r}$, either the output of $F(p|q)$ or that of $R(c|q)$, and computes the probability that given representation $\mathbf{r}$ comes from a real compact-answer using a feedforward network with two hidden layers (100 nodes in the first layer and 50 in the second layer) and a logistic regression layer on top of the hidden layers. We used sigmoid outputs by the logistic regression layer as the output probability.

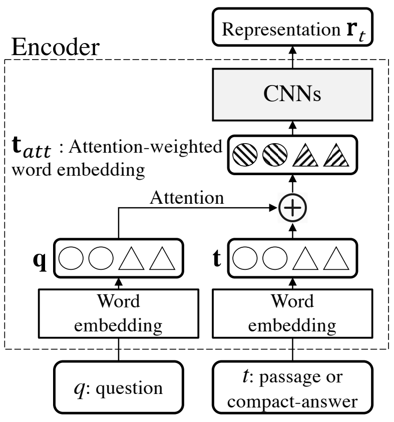

Representation $\mathbf{r}_t$
Encoder
CNNs
$\mathbf{t}_{att}$: Attention-weighted word embedding
Attention
q
t
Word embedding
Word embedding
q: question
t: passage or compact-answer

Figure 2: Architecture of generators F and R

3.3 Encoder

Figure 2 illustrates the architecture shared by our fake-representation generator F and real-representation generator R, namely, Encoder$(t; \theta, q)$, where θ is a set of parameters, q is a why-question, and t is either an answer passage or a manually created compact-answer. Encoder$(t; \theta, q)$ first represents question q and passage/compact-answer t with pre-trained word embeddings, which are supplemented with attention mechanisms. The resulting attention-weighted word embeddings are given to convolutional neural networks (CNNs) that generate a single feature vector, which is an output/value of Encoder$(t; \theta, q)$.

In the following, we give an overview of the word embeddings, the attention mechanisms, and the CNNs used in Encoder$(t; \theta, q)$. All of these techniques were proposed by previous works. Further details are given in Section B of the supplementary materials.

3.3.1 Word embeddings

The pre-trained word embeddings used in Encoder$(t; \theta, q)$ were obtained by concatenating two types of d-dimensional word embeddings ($d = 300$ in this work): *general word embeddings* and *causal word embeddings*.

General word embeddings are widely used embedding vectors (300 dimensions) that were pre-trained for about 1.65 million words by applying *word2vec* (Mikolov et al., 2013) to about 35 million sentences from Japanese Wikipedia (January 2015 version).

Causal word embeddings (Sharp et al., 2016) were proposed for representing the *causal associations* between words. Sharp et al. (2016) created a set of cause-effect word pairs by paring each content word in a cause part with each content word in an effect part of the same causality expression, such as "Volcanoes erupt *because* magma pushes through vents and fissures." In this work, we extracted 100 million causality expressions from 4-billion Japanese web pages using the causality recognizer of Oh et al. (2013). Then, following Sharp et al. (2016), we trained 300-dimensional causal word embeddings for about 1.85 million words by applying the generalized skip-gram embedding model of Levy and Goldberg (2014) to the causality expressions.

3.3.2 Attention

We also applied two types of attention mechanisms to the above word embeddings. The first type of attention, *similarity-attention* (dos Santos et al., 2016; Tan et al., 2016), was used for estimating the similarities between words in question q and those in passage/compact-answers t and focusing on the attended words as those that directly indicate the connection between the question and passage/compact-answers. Basically, the mechanism computes the cosine similarity between the embeddings of the words in q and t, and uses it for producing attention feature vector $a^s_j \in \mathbb{R}$ for word t_j in passage/compact-answers.

Another attention mechanism, *causality-attention* (Oh et al., 2017), was proposed for focusing on passage words causally associated with question words. They used normalized point-wise mutual information to measure the strength of the causal associations with the causality expressions used for creating the causal embeddings. The scores are used for producing causality-attention feature vector a^c_j for word t_j.

Finally, we form two attention feature vectors, $\mathbf{a}^s = [a^s_1, \cdots, a^s_{|t|}]$ and $\mathbf{a}^c = [a^c_1, \cdots, a^c_{|t|}]$, concatenate them into $\mathbf{a} = [\mathbf{a}^s; \mathbf{a}^c] \in \mathbb{R}^{2\times|t|}$, and produce attention-weighted word embedding $\mathbf{t}_{att}$ of given text t, which is either an answer passage or a compact answer:

$$\mathbf{t}_{att} = \mathrm{ReLU}(\mathbf{W}_t \mathbf{t} + \mathbf{W}_a \mathbf{a})$$

where $\mathbf{W}_t \in \mathbb{R}^{2d\times 2d}$ and $\mathbf{W}_a \in \mathbb{R}^{2d\times 2}$ are trainable parameters, $\mathbf{t}$ is the representation of text t, and ReLU represents the rectified linear units.

3.3.3 CNNs

$\mathbf{t}_{att}$ is given to CNNs to generate final representation $\mathbf{r}_t$ of a given passage/compact-answer t.

The CNNs resembles those in Kim (2014). Convolutions are performed over the word embeddings using both multiple filters and multiple filter windows (e.g., sliding over 1, 2, or 3 word windows at a time and 100 filters for each window). An average pooling operation is applied to the convolution results to generate representation $\mathbf{r}_t$, which is the output/value of Encoder$(t; \theta, q)$; $\mathbf{r}_t = \text{Encoder}(t; \theta, q)$. In our experiments, we set the dimension of representation $\mathbf{r}_t$ to 300.

4 Why-QA Experiments

4.1 Datasets

We used three datasets, $WhySet$, $CmpAns$, and $AddTr$, for our why-QA experiments. $WhySet$ and $AddTr$ were used for training and evaluating the why-QA models, while $CmpAns$ was used for training AGR.

The $WhySet$ dataset, which was used in previous works for why-QA (Oh et al., 2012, 2013, 2016, 2017), is composed of 850 Japanese why-questions and their top-20 answer passages (17,000 question-passage pairs) obtained from 600 million Japanese web pages using the answer-retrieval method of Murata et al. (2007), where a question-passage pair is composed of a single-sentence question and a five-sentence passage. The label of each question-answer pair (i.e., correct answer and incorrect answer) was manually annotated (See Oh et al. (2013) for more details). Oh et al. (2013) selected 10,000 question-passage pairs as training and test data in 10-fold cross-validation (9,000 for training and 1,000 for testing) and used the remainder (7,000 question-passage pairs) as additional training data during the 10-fold cross-validation. We followed the settings and, in each fold, we selected 1,000 pairs from the 9,000 pairs for training to use as development data for tuning hyperparameters. Note that there are no shared questions in the training, development, or test data.

For training the AGR, we used $CmpAns$, the training data set created in Ishida et al. (2018) for compact-answer generation; $CmpAns$ consists of 15,130 triples of a why-question, an answer passage, and a manually-created compact answer. These cover 2,060 unique why-questions. Note that there was no overlap between the questions in $CmpAns$ and those in $WhySet$. $CmpAns$ was created in the following manner: 1) human annotators manually came up with open-domain why-questions, 2) retrieved the top-20 passages for each why-question using the open-domain why-QA module of a publicly available web-based QA system WISDOM X (Mizuno et al., 2016; Oh et al., 2016), and 3) three annotators created (when possible) a compact answer for each of the retrieved passages. The passages for which no annotator could create a compact answer were discarded, and were not included in the 15,130 triples mentioned previously. The average lengths of questions, passages, and compact answers in $CmpAns$ were 10.5 words, 184.4 words, and 8.3 words, respectively.

Finally, we created additional training data $AddTr$ for training the why-QA models. If an annotator could write a compact answer for a question and an answer passage, she/he probably recognized the passage as a proper answer passage to the question. Based on this observation, we built $AddTr$ from $CmpAns$ by applying a majority vote. We only gave a correct answer label to a question and a passage if at least two of the three annotators wrote compact answers, and it received an incorrect answer label otherwise. $AddTr$ has 10,401 pairs in total. We used $AddTr$ as additional training data for baselines that lack a mechanism for generating compact-answer representations, for a fair comparison with other methods that use $CmpAns$ for such mechanisms.

We processed all the data with MeCab[1], a morphological analyzer, to segment the words.

4.2 Training details

In our proposed methods and their variants, all the weights in the CNNs were initialized using He's method (He et al., 2015), and the other weights in our why-QA model were initialized randomly with a uniform distribution in the range of (-0.01, 0.01). For the CNN-based components, we set the window size of the filters to "1,2,3" with 100 filters each[2]. We used dropout (Srivastava et al., 2014) with probability 0.5 on the final logistic regression layer. All of these hyper-parameters were chosen with our development data. We optimized the learned parameters with the Adam stochastic gradient descent (Kingma and Ba, 2015). The learning rate was set to 0.001, and the batch size

[1]http://taku910.github.io/mecab/

[2]We examined all of the following combinations of number of filters and filter window size: any of {25,50,75,100} for the former, and any of {"1,2,3", "2,3,4", "1,2,3,4"} for the latter.

for each iteration was set to 20.

4.3 Compared methods

We tried three schemes for training our AGR in our proposed method. In the first scheme, pairs of passages and compact answers in $CmpAns$ were given to fake-representation generator F and real-representation generator R as their inputs. We called the fake-representation generator trained in this way F_{OP} and referred to our proposed method using F_{OP} as `Ours(OP)`. In the second scheme, we randomly sampled five-sentence passages that contain some clue words indicating the existence of causal relations, such as "because," from 4-billion web pages and fed them to fake-representation generator F. We fed the same number of the sampled passages as in $CmpAns$ for fair comparison. We refer to the method trained by this scheme as `Ours(RP)`. In the final scheme, we replaced the word embeddings for the passages given to fake-representation generator F with random vectors and used similarity-attention but not causality-attention. The fake-representation generator trained in this way is called F_{RV}, and our proposed method using F_{RV} is called `Ours(RV)`. This scheme is more similar to the original GAN than the others because the fake-representation generator is given *random noises*.

We implemented and evaluated the following four why-QA models in previous works as baselines, using the same dataset as ours:

1) Oh et al. (2013): an SVM-based model that uses bag-of-word and causality features;

2) Sharp et al. (2016): a CNN model that represents a question and its answer passage separately by using cause and effect word-embedding vectors, which were trained with our causality expressions by the same way as in Sharp et al. (2016);

3) Tan et al. (2016): an LSTM model with similarity-attention;

4) Oh et al. (2017): the state-of-the-art neural model for Japanese why-QA that uses causality-attention and causality expressions.

We also evaluated nine baseline neural models, four of which are BERT-based models (`BERT`, `BERT+AddTr`, `BERT+F`$_{\mathrm{OP}}$, and `BERT+F`$_{\mathrm{RV}}$), to show the effectiveness of our why-QA model and AGR. They are listed in Table 2.

Method	Description
`BASE`	Proposed method from which we removed fake-representation generator F.
`BASE+AddTr`	`BASE` that used both $WhySet$ and $AddTr$ as its training data.
`BASE+CAns`	On top of `BASE`, it additionally used real-representation generator R to encode compact answers, which were generated by the compact-answer generator of Iida et al. (2019). R was trained alongside the why-QA model using $WhySet$ and the compact-answer generator was pre-trained with $CmpAns$.
`BASE+CEnc`	On top of `BASE`, it additionally used the encoder in the compact-answer generator of Iida et al. (2019) to create compact-answer representation. The encoder was pre-trained with $CmpAns$.
`BASE+Enc`	Same as `Ours(OP)` except that the fake-representation generator was trained in a supervised manner alongside the why-QA model using $WhySet$ and $AddTr$ as the training data.
`BERT`	Same as `BASE` except that the CNN-based encoders for questions and passages were replaced with the BERT (Devlin et al., 2019).
`BERT+AddTr`	`BERT`, which used both $WhySet$ and $AddTr$ as its training data.
`BERT+F`$_{\mathrm{OP}}$	On top of `BERT`, it additionally used compact-answer representation produced by F_{OP} for answer selection.
`BERT+F`$_{\mathrm{RV}}$	Same as `BERT+F`$_{\mathrm{OP}}$ except that it used F_{RV} instead of F_{OP} for producing compact-answer representation.

Table 2: Baseline neural models

To pre-train the BERT-based models, we used a combination of sentences extracted from Japanese Wikipedia articles (August 2018 version) and causality expressions automatically recognized from a causality recognizer (Oh et al., 2013). This data mix consists of 75% of sentences extracted from Wikipedia (14,675,535 sentences taken out of 784,869 articles randomly sampled) and 25% of cause and effect phrases taken from causality expressions (4,891,846 phrases from 2,445,923 causal relations). This ratio was determined through preliminary experiments using the development data. For the pre-training parameters, we followed the settings of BERT$_{\mathrm{BASE}}$ in Devlin et al.

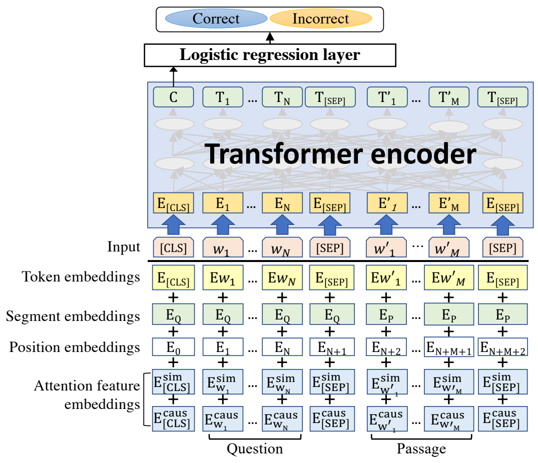


Figure 3: Architecture of BERT: E represents input embedding and T_i represents contextual representation of token i. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. for separating questions/passages).

(2019)[3] except for the batch size of 50. We ran 3 epochs with the learning rate of 1e-5 for fine-tuning the BERT-based models[4].

A BERT-based model, BERT, takes a question-passage pair as input and computes the input representation using token, segment, position, and attention feature embeddings (Fig. 3). For the input representation computation, the original BERT only used the token, segment, and position embeddings, while BERT additionally used the attention feature embeddings[5] to exploit the same similarity-attention and causality-attention features used in our proposed method. We used the attention feature embeddings during the fine-tuning and testing, but not during the pre-training of the BERT-based model. The attention feature embeddings for answer passages (i.e., $E^{sim}_{w'_1}, \cdots, E^{sim}_{w'_M}$, and $E^{caus}_{w'_1}, \cdots, E^{caus}_{w'_M}$) were computed from the same attention feature vectors, $\mathbf{a}^s$ and $\mathbf{a}^c$, as those in our proposed methods; those for the other parts (i.e., questions, [CLS], and [SEP]) were computed from a zero vector (indicating no attention feature). The transformer encoder processed the input representation to generate a representation for a question-passage pair and passed the generated representation to the logistic regression layer for answer selection. BERT was trained with the training data in $WhySet$. BERT+AddTr is the same as BERT except that it additionally used *AddTr* as training data. On top of BERT, BERT+F$_{OP}$ and BERT+F$_{RV}$ additionally used the compact-answer representation produced by our fake-representation generator for answer selection by giving it to the final logistic regression layer.

[3]12-layers, 768 hidden states, 12 heads and training for 1-million steps with the warmup rate of 1% using Adam optimizer with the learning rate of 1e-4.

[4]We tested all the combinations of epochs {1, 2, 3, 4, 5} and learning rates of {1e-5, 2e-5, 3e-5} and chose the one that maximized the performance on the development data in $WhySet$.

[5]We also evaluated a BERT-based model that did not use the attention feature embeddings, but its P@1 (41.4) was much lower than that of BERT (51.2).

	P@1	MAP
Oh et al. (2013)	41.8	41.0
Sharp et al. (2016)	33.2	32.2
Tan et al. (2016)	34.0	33.4
Oh et al. (2017)	47.6	45.0
BASE	51.4	50.4
BASE+AddTr	52.0	49.3
BASE+CAns	51.8	50.3
BASE+CEnc	52.4	51.5
BASE+Enc	52.2	50.6
BERT	51.2	50.8
BERT+AddTr	51.8	51.0
BERT+F$_{OP}$	53.4	51.2
BERT+F$_{RV}$	53.2	50.9
Ours(OP)	**54.8**	**52.4**
Ours(RP)	53.4	51.5
Ours(RV)	54.6	51.8
Oracle	60.4	60.4

Table 3: Why-QA performances

4.4 Results

Table 3 shows the performances of all the methods in the Precision of the top answer (P@1) and the Mean Average Precision (MAP) (Oh et al., 2013). Note that the Oracle method indicates the performance of a *fictional* method that ranks the answer passages perfectly, i.e., it locates all the m correct answers to a question in the top-m ranks, based on the gold-standard labels. This performance is the upper bound of those of all the implementable methods.

Our proposed method, Ours(OP), outperformed all the other methods. Our starting point, i.e., BASE, was already superior to the methods in the previous works. Compared with BASE and BASE+AddTr, neither of which used compact-answer representations or fake-representation generator F, Ours(OP) gave 3.4% and 2.8% improvement in P@1, respectively. It also outperformed BASE+CAns and BASE+CEnc, which generated compact-answer representations in a way different from the proposed method, and BASE+Enc, which trained the fake-representation generator without adversarial learning. These performance differences were statistically significant ($p < 0.01$ by the McNemar's test).

`Ours(OP)` also outperformed all the BERT-based models but an interesting point is that fake-representation generator F boosted the performance of the BERT-based models (statistically significant with $p < 0.01$ by the McNemar's test). These results suggest that AGR is effective in both our why-QA model and our BERT-based model.

4.5 Deeper analysis on the output of F

Another interesting point is that `Ours(RV)`, in which fake-representation generator F_{RV} was trained using random vectors, achieved almost the same performance as that of `Ours(OP)`. This result was puzzling, so we first checked whether F_{RV}'s output was not just random noise (which could prevent the why-QA model from overfitting) by replacing in `Ours(RV)` the output of F_{RV} by random vectors. Although we sampled the random vectors from different distribution types with various ranges, we obtained at best similar performance to that of `BASE`: 51.6 in P@1. This result confirms that it is not trivial to mimic F_{RV} using random vectors at least.

We investigated the F_{RV}'s output to check whether it actually *focused* on the compact answer in a given passage. We computed the following three representation sets from a gold set of 3,608 triples of why-questions, answer passages and manually created compact-answers that do not overlap with $CmpAns$:

- $\{\mathbf{r}_i^{org}\}$: F_{RV}'s output with the pairs of a why-question and an answer passage in the gold set as its input;
- $\{\mathbf{r}_i^{in}\}$: F_{RV}'s output for the same input as $\{\mathbf{r}_i^{org}\}$, where we replaced the *word embeddings* of all the content words in the answer passages that *also appeared in* the associated gold compact-answers with *random vectors*;
- $\{\mathbf{r}_i^{out}\}$: F_{RV}'s output for the same input as $\{\mathbf{r}_i^{org}\}$, where we replaced the *word embeddings* of all the content words in the answer passages that *did not appear in* the associated gold compact-answers with *random vectors*[6].

If F_{RV} perfectly focuses on the gold standard compact-answers, for each question-passage pair, $\mathbf{r}_i^{out}$ should be the same as $\mathbf{r}_i^{org}$ and $\mathbf{r}_i^{in}$ should significantly differ from $\mathbf{r}_i^{org}$. Next we computed the average Euclidian distance among $\{\mathbf{r}_i^{org}\}$, $\{\mathbf{r}_i^{in}\}$ and $\{\mathbf{r}_i^{out}\}$. The average distance (2.67) between $\{\mathbf{r}_i^{org}\}$ and $\{\mathbf{r}_i^{out}\}$ was much smaller than the average distance (13.3) between $\{\mathbf{r}_i^{org}\}$ and $\{\mathbf{r}_i^{in}\}$. Note that we replaced the word embeddings for much more words with random vectors in the computation of $\{\mathbf{r}_i^{out}\}$ than those in the computation of $\{\mathbf{r}_i^{in}\}$ (38.1 words vs. 5.6 words). This implies that the distance between $\{\mathbf{r}_i^{org}\}$ and $\{\mathbf{r}_i^{out}\}$ might be much larger than that between $\{\mathbf{r}_i^{org}\}$ and $\{\mathbf{r}_i^{in}\}$ if F_{RV} focused equally on every answer passage word. However, the actual results suggest that this is not the case. Although we cannot draw decisive conclusions due to the complex nature of neural networks, we believe from the results that F_{RV} does actually focus more on words that are a part of a compact answer than on other words. We also computed $\{\mathbf{r}_i^{org}\}$, $\{\mathbf{r}_i^{in}\}$, and $\{\mathbf{r}_i^{out}\}$ with fake-representation generator F_{OP} in the same way and observed the same tendency.

[6]For both $\mathbf{r}_i^{in}$ and $\mathbf{r}_i^{out}$, we never replaced the word embeddings for the words that also appeared in the question.

	Train	Dev	Test
Quasar-T	37,012	3,000	3,000
SearchQA	99,811	13,893	27,247
TriviaQA	87,291	11,274	10,790*
SQuAD v1.1	87,599	10,570*	NA

Table 4: Number of questions in each dataset: datasets marked with $*$ were not used in this experiment

5 DS-QA Experiments

We tested our framework on another task, the distantly supervised open-domain question answering (DS-QA) task (Chen et al., 2017), to check its generalizability. Table 4 shows the statistics for the datasets used in this experiment. The first three, Quasar-T, SearchQA, and TriviaQA provided by Lin et al. (2018), were used for training and evaluating DS-QA methods. The training data of SQuAD v1.1 (Rajpurkar et al., 2016) was used for training our AGR. The SQuAD dataset consisted of the triples of a question, an answer, and a paragraph that includes the answer. We assume that the answers are *our compact answers*, although the answers in the dataset are consecutive short word sequences (2.8 words on average), whose majority are noun phrases, unlike the compact answers for our why-QA experiment, i.e., sentences or phrases (8.3 words on average).

We trained our AGR with all the triples of a question, an answer, and a paragraph in the training data of SQuAD-v1.1 under the same settings for the AGR's hyperparameters as in our why-QA experiment except that we use neither causal word embeddings nor causality-attention. In this experiment, we used the AGR training schemes for `Ours(OP)` and `Ours(RV)`. We used the 300-dimensional GloVe word embeddings learned from 840 billion tokens in the web crawl data (Pennington et al., 2014), as general word embeddings. Then we combined the resulting fake-representation generator F in the AGR with the state-of-the-art DS-QA method, OpenQA (Lin et al., 2018)[7]. We also used the hyperparameters presented in Lin et al. (2018).

OpenQA is composed of two components: a *paragraph selector* to choose relevant paragraphs (or answer passages in our terms) from a set of paragraphs and a *paragraph reader* to extract answers from the selected paragraphs. For identifying answer a to given question q from set of paragraphs $P = \{p_i\}$, the paragraph selector and the paragraph reader respectively compute probabilities $Pr(p_i|q,P)$ and $Pr(a|q,p_i)$, and final output $Pr(a|q,P)$ is obtained by combining the probabilities. We introduced $\mathbf{c}_i$, which is a compact-answer representation generated by fake-representation generator F with question q and paragraph p_i as its input, to the computation of the probabilities as follows:

$$Pr(a|q,P,C) = \sum_i Pr(a|q,p_i,\mathbf{c}_i)Pr(p_i|q,P,\mathbf{c}_i)$$

In the original OpenQA, the paragraph selector and the reader use bidirectional stacked RNNs for encoding paragraphs, where word embeddings $\mathbf{p}_i$ of a paragraph is used as the input. In our implementation, we computed attention-weighted embedding $\bar{\mathbf{p}}_i$ of a paragraph by using compact-answer representation $\mathbf{c}_i$. Given word embedding $\mathbf{p}_i^j$ for the j-th word in paragraph p_i, its attention-weighted embedding $\bar{\mathbf{p}}_i^j$ was computed by using a bilinear function (Sutskever et al., 2009):

$$\bar{\mathbf{p}}_i^j = \text{softmax}_j(\mathbf{p}_i^\mathsf{T}\mathbf{M}\mathbf{c}_i)\mathbf{p}_i^j,$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a trainable matrix, $\text{softmax}_j(\mathbf{x})$ denotes the j-th element of the *softmaxed* vector of $\mathbf{x}$, and $d = 300$. We gave $[\mathbf{p}_i^j; \bar{\mathbf{p}}_i^j]$, a concatenation of $\mathbf{p}_i^j$ and $\bar{\mathbf{p}}_i^j$, as the word embedding of the j-th word in paragraph p_i to the bidirectional stacked RNNs.

[7]https://github.com/thunlp/OpenQA

	Quasar-T		SearchQA		TriviaQA	
	EM	F1	EM	F1	EM	F1
R^3	35.3	41.7	49.0	55.3	47.3	53.7
`OpenQA`	42.2	49.3	58.8	64.5	48.7	**56.3**
`Ours(OP)`	**43.2**§	49.7	59.6†	**65.3**†	**49.6**†	54.8
`Ours(RV)`	42.9§	**49.9**	**59.7**†	**65.3**†	**49.6**†	54.7

Table 5: DS-QA performances: Result of TriviaQA dataset is on its development data. § and † respectively indicate statistical significance of difference in performance between `Ours(·)` and `OpenQA` by the McNemar's test with $p < 0.05$ and $p < 0.01$

Table 5 shows the performances of the four DS-QA methods: R^3 (Wang et al., 2018), `OpenQA` (Lin et al., 2018), `Ours(OP)`, and `Ours(RV)` evaluated against the Quasar-T, SearchQA and TriviaQA datasets. All the methods were evaluated with EM and F1 scores, following Lin et al. (2018). EM measures the percentage of predictions that exactly match one of the ground-truth answers and F1 is a metric that loosely measures the average overlap between the prediction and ground-truth answer. Note that both `Ours(OP)` and `Ours(RV)` outperformed both previous methods, R^3 and `OpenQA`, except for the F1 score for the TriviaQA dataset. Some of the improvements over the previous state-of-the-art method, `OpenQA`, were statistically significant. These findings suggest that our framework can be effective for tasks other than the original why-QA and the other datasets.

6 Conclusion and Future Work

We proposed a method for why-question answering (why-QA) that used an adversarial learning framework. It employed adversarial learning to generate vector representations of reasons or *true answers* from answer passages and exploited the representations for judging whether the passages are proper answer passages to the given why-questions. Through experiments using Japanese why-QA datasets, we showed that this idea improved why-QA performance. We also showed that our method improved the performance in a distantly supervised open-domain QA task.

In our why-QA method, causality expressions extracted from the web were used as background knowledge for computing causality-attention/embeddings. As a future work, we plan to introduce a wider range of background knowledge including another type of event causality (Hashimoto et al., 2012, 2014, 2015; Kruengkrai et al., 2017).

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, pages 76–83.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA. MIT Press.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 619–630.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI-15)*, pages 2396–2403.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 987–997.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1026–1034, Washington, DC, USA. IEEE Computer Society.

Ryuichiro Higashinaka and Hideki Isozaki. 2008. Corpus-based question answering for why-questions. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 418–425.

Ryu Iida, Canasai Kruengkrai, Ryo Ishida, Kentaro Torisawa, Jong-Hoon Oh, and Julien Kloetzer. 2019. Exploiting background knowledge in compact answer generation for why-questions. In *Proceedings of Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19)*.

Ryo Ishida, Kentaro Torisawa, Jong-Hoon Oh, Ryu Iida, Canasai Kruengkrai, and Julien Kloetzer. 2018. Semi-distantly supervised neural model for generating compact answers to open-domain why questions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 5803–5811. AAAI Press.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, pages 1–13.

Canasai Kruengkrai, Kentaro Torisawa, Chikara Hashimoto, Julien Kloetzer, Jong-Hoon Oh, and Masahiro Tanaka. 2017. Improving event causality recognition with multiple background knowledge sources using multi-column convolutional neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 1736–1745.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Junta Mizuno, Masahiro Tanaka, Kiyonori Ohtake, Jong-Hoon Oh, Julien Kloetzer, Chikara Hashimoto, and Kentaro Torisawa. 2016. WISDOM X, DISAANA and D-SUMM: Large-scale NLP systems for analyzing textual big data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 263–267.

Masaki Murata, Sachiyo Tsukawaki, Toshiyuki Kanamaru, Qing Ma, and Hitoshi Isahara. 2007. A system for answering non-factoid Japanese questions by using passage retrieval weighted based on type of answer. In *Proceedings of NTCIR-6*.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Ryu Iida, Masahiro Tanaka, and Julien Kloetzer. 2016. A semi-supervised learning approach to why-question answering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3022–3029.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Takuya Kawada, Stijn De Saeger, Jun'ichi Kazama, and Yiou Wang. 2012. Why question answering using sentiment analysis and word classes. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 368–378.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-question answering using intra- and inter-sentential causal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1733–1743.

Jong-Hoon Oh, Kentaro Torisawa, Canasai Kruengkrai, Ryu Iida, and Julien Kloetzer. 2017. Multi-column convolutional neural networks with causality-attention for why-question answering. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 415–424.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 138–148.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Joshua B. Tenenbaum, and Ruslan R Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in Neural Information Processing Systems 22*, pages 1821–1828. Curran Associates, Inc.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany. Association for Computational Linguistics.

Suzan Verberne, Hans van Halteren, Daphne Theijssen, Stephan Raaijmakers, and Lou Boves. 2011. Learning to rank for why-question answering. *Inf. Retr.*, 14(2):107–132.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 5981–5988.