Abstract

In existing studies on Visual Question Answering (VQA), which aims to train an intelligent system to be able to answer questions about images, the answers corresponding to the questions consists of short, almost single words. However, considering the natural conversation with humans, the answers would more likely be sentences, rather than single words. In such a situation, the system needs to focus on a keyword, i.e., the most important word in the sentence, to answer the question. Therefore, we have proposed a novel keyword extraction method for VQA. Because collecting keywords and full-sentence annotations for VQA can be highly costly, we perform the keyword extraction in an unsupervised manner. Our key insight is that the full-sentence answer can be decomposed into two parts: the part contains new information for the question and the part only contains information already included in the question. Since the keyword is considered as the part which contains new information as the answer, we need to identify which words in the full-sentence answer are the part of new information and which words are not. To ensure such decomposition, we extracted two features from the full-sentence answers, and designed discriminative decoders to make each feature to include the information of the question and answers respectively. We conducted experiments on existing VQA datasets, which contains full-sentence annotations, and show that our proposed model can correctly extract the keyword without any keyword annotations.

1. Introduction

Visual recognition systems are one of the most active research fields, and it is expected to apply that research to systems running in the real world, for example robots. In the typical setting, the visual recognition system is trained in a fully supervised way. That is, all object classes to be recognized are already included in the training data, and the classes that are not included in the training data are out of the scope of recognition. However, there is an innumerable number of object classes in the real world; training all of them is impossible. The one of the promising way to solve this problem is learning by asking, i.e., generating questions to humans about unknown objects, and learning new knowledge from the human response [26, 22, 18].

Regarding learning from questions about an image, Visual Question Answering (VQA) [2, 9] is a well-known vision and language task, which aims to develop a system that can answer a question about an image. A typical dataset used in VQA is the VQA v2 dataset [9]. The VQA v2 dataset is used in VQA challenge competition, and various models [8, 35, 25, 34, 13] have been proposed to improve the performance of VQA. The answers in the VQA v2 dataset are basically a single word. Therefore, the diversity of the answers is limited. The typical VQA model is built as a multi-class classification model, using about 3,000
most frequent answers in the dataset.

However, considering the actual conversation with humans, the answers are rarely expressed by a single word, and instead are often expressed as sentences. In fact, the average length of answers is 6.5 words in the VisDial v1.0 dataset [6], which is a dataset of natural conversations about an image. This is much longer than the average length of the answers in the VQA v2 dataset – 1.2 words.

This suggests that human answers are likely to be obtained in the form of sentences when asking humans questions in a natural scenario. However, to use the obtained answers for various tasks, such as object class recognition and relationship detection, it is desirable that the answer be represented by a single word, like a class label. Therefore, it is necessary to extract the word that is most relevant, to the question and the image from the answer, from the full-sentence answer. In this paper, we tackle the task of extracting keywords when the answers from VQA are obtained in the form of sentences (Full-sentence VQA). To extract a keyword from a full-sentence VQA, the simplest method is to construct a dataset with full-sentence answers and keyword annotations, and train the model in a supervised manner. However, the cost of constructing such a dataset and annotating the keywords in the answer is very high. If we can train a keyword extraction model in an unsupervised manner, using a dataset with only full-sentence answers (w/o keyword annotations), we can save the high cost of collecting keyword annotations.

In this paper, we propose an unsupervised keyword extraction model using a full-sentence VQA dataset, which contains no keyword annotations.

Here, our main idea is based on an intuition that the keyword is the most informative word in the full-sentence answer, and contains the information not included in the question. That is to say, the full-sentence answer can be decomposed into the keyword information that is not included in the question, and the information that is already included in the question. For example, in the answer “The candles are in front of the bear,” the word “candles” is the keyword, while the last part “The something is in front of the bear” is information already included in the question, that is, there is no new information besides the question. We intend to extract two features from the full-sentence answer, which represent the information of the keyword and the information derived from the question, respectively. To ensure these two features to discriminatively contain the keyword information and the question information, we intend to reconstruct the original answers from the keyword features, and the original questions from the question features.

Given an image, its corresponding question, and full-sentence answer, our proposed model extracts the keyword of the answer by decomposing the keyword information and the question information in the answer. Our model consists of the three modules: the encoder module $E$, the attention scoring module $S$, and the decoder module $D$.

The encoder $E$ encode the image and question features into joint features. The attention scoring module $S$ weights each words, according to its importance, and calculates its weighted sum to generate two features, that represent the keyword and the question information, respectively. This module is inspired by the researches on the attention mechanisms, particularly by the Transformer [27]. Finally, the decoder module $D$ reconstructs the original sentences from the output of $S$.

We conducted experiment on two datasets, GQA [11] and FSVQA [23], which contain both full-sentence and single-word answers. Note that annotation in both datasets is not provided by humans. The questions and answers in the GQA dataset are automatically generated from scene graphs, and the full-sentence answers in the FSVQA dataset are generated from single-word answers, in the VQA v1
dataset [2], by applying linguistic rules.

To the best of our knowledge, our work is the first attempt at extracting a keyword from full-sentence VQA in an unsupervised manner.

Overall, the main contributions of our work are

1. We proposed a novel task of extracting keywords from the full-sentence VQA without any keyword annotations.
2. We designed a novel, unsupervised keyword extraction model, based on the disentanglement of the full-sentence answer, and the reconstruction of the original answer and the question.
3. We conducted experiments on two VQA datasets, and provided both qualitative and quantitative results that shows the effectiveness of our model.

2. Related works

2.1. Unsupervised Keyword Extraction for Text

There is much research on keyword extraction from text, both in a supervised and unsupervised manner. Our work is one of the unsupervised methods, so we focus on unsupervised methods here.

Unsupervised keyword extraction methods are roughly categorized into two approaches: graph-based methods and statistical methods.

In graph-based methods [17], the target document is regarded as a graph, with each word as a node, and the relationship between words (e.g., co-occurrence [17, 28]) is applied as an edge weight. However, these methods need to construct a graph from the target document, and use a relationship such as co-occurrence between words as edges. Therefore, the document is required to have a certain length. Here, we try to extract keywords from full-sentence VQA dataset, in which the average word length of the answers is about 10 words. Therefore, graph-based methods are not suitable for our target dataset.

The statistical methods rely on statistics obtained from a document, such as the term frequency and the position of a word. The most basic method of the statistical method is TF-IDF [20], which calculates the term frequency from the target document, and the inverse document frequency from the large corpus, and scores each word in the target document. Other statistical methods consider different features, like the co-occurrence of the candidate phrases [21], which is obtained by splitting the document by delimiters and stopwords, but this is not suitable for short text because of the usage of the co-occurrence.

2.2. Attention

The attention mechanism is a technique originally proposed in machine translation [4], which aims to focus on the most important part of the input sequences for a task. The attention mechanism has been applied to various tasks, such as document classification [33], question answering [29, 24], image captioning [32, 16], and VQA [12, 1, 31, 35].

In the following part, we describe the research on the attention mechanism, in natural language processing, related to this work. In general, the attention mechanism is described as learning the mapping between a query and key-value pairs. In machine translation, the decoder’s hidden state is treated as a query, and the key and value vectors are generated from the encoder’s hidden state, and the attention is computed as the weighted sum of the value vector, dependent on the query and the key vectors.

Transformer [27] is one of the most popular attention mechanisms for machine translation, which enables machine translation without using recurrent neural networks, but using a self-attention mechanism and feed-forward networks.

Another line of research uses attention for supervised keyword extraction [30]. Its model was trained for document classification, and extracted the word, that seemed to be important for the task, as the keyword. This system requires additional annotations of document class labels, to train the model, whereas our study aims to extract keywords without any additional annotations, a significant difference.

2.3. Visual Question Answering

VQA is a well-known task of learning from questions and answers about images. In VQA, the most representative dataset is VQA v2 [9], and much research uses this dataset for performance evaluations. In general, VQA is regarded as a classification task, with target classes of $k$ frequent answers [12, 25]. In the VQA v2 dataset, the average number of words in an answer is only 1.2 words, and the variety of the answers is relatively limited. Therefore, by setting the number of target classes $k$ to about 3,000, about 93% of the questions can be covered [25].

However, in natural question answering by humans, the answers will be expressed as a sentence rather than a single word. For example, in the Visual Dialog dataset [6], which collects natural dialogue about images, human answers to the questions are often sentences, and the average number of the words in the answer is 6.5 words, which is larger than VQA v2 dataset average.

The FSVQA [23] is a VQA dataset with answers in the form of full sentences. In this dataset, the full-sentence answers are automatically generated, by applying the rule-based natural language processing to the questions and single-word answers, in the VQA v1 dataset [2].

The GQA dataset [11] is recently proposed and is a dataset also with automatically generated full-sentence answers. The questions and answers (both single-word and
3. Model

First, we present the overview of our proposed model. Then, we describe the details of each module of the model and their objective functions.

3.1. Overview

The overview of the model is shown in Figure 3. This model consists of the encoder $E$, the attention scoring modules $S_a$ and $S_q$, and the decoder modules $D_{all}$, $D_a$, and $D_q$.

As the input to the model, we use an image $I$ and the corresponding question $Q$ and the full-sentence answer $A = \{w_1, w_2, \ldots, w_n\}$. Here, $w_i$ represents the $i$-th word in the full-sentence answer.

Given $I$ and $Q$, $E$ extracts image and question features, and integrates them into joint features $f_j$, i.e., $E(I, Q) = f_j$.

Next, $S_a$ and $S_q$ use $f_j$ and $A$ as input, and output the weight vectors, $a_k$ and $a_q$. Here, $a_k = \{a_{k1}, a_{k2}, \ldots, a_{kn}\}$ and $a_q = \{a_{q1}, a_{q2}, \ldots, a_{qn}\}$ for each word in $A$. We denote $a_i \in (0, 1)$ as the weight score of the $i$-th word in $A$.

Then, we consider the keyword vector $f_k$ as the embedding vector of the word with the highest weight score in $a_k$, and the question information vector $f_q$ as the weighted sum of the embedding vectors of $A$ corresponding to the weight score $a_q$.

Next, $D_{all}$ reconstructs the original full-sentence answer, using $f_k$ and $f_q$ using LSTM. We intend $f_k$ to be the keyword vector in the full-sentence answer, i.e., the most informative word in the full-sentence answer, and $f_k$ to represent the question information. However, $D_{all}$ only ensures both features to have the information of the full-sentence answer. To separate them, we designed the additional decoders, $D_a$ and $D_q$. $D_a$ reconstructs the BoW features of the answer from $f_k$, and $D_q$ reconstructs the BoW features of the question, from $f_q$ with auxiliary vectors. Our intention in this operation is to make $f_k$ and $f_q$ representative features for the full-sentence answer and the question, respectively.

The entire model is trained to minimize the error between the reconstructed sentences $A_{recon}$ and the original full-sentence answers, and the error between the BoW fea-
tures of the full-sentence answers and the questions.

3.2. Encoder

This module \( E \) encodes the image \( I \) and the question \( Q \) and obtains the image feature \( f_I \), the question feature \( f_Q \), and the joint feature \( f_J \). To generate \( f_I \), we use the image feature extracted from a deep CNN, which is pre-trained on a large scale image recognition dataset. For \( f_Q \), we converted each word token into a word embeddings and average them. Then, we perform \( l_2 \) normalization to both features. Finally, those features are concatenated to the joint feature \( f_J \in \mathbb{R}^{d_I} \), i.e., \( E(I, Q) = f_J = [f_I; f_Q] \), where \( d_J \) is the dimension of the joint feature, and \([;]\) indicates concatenation. Note that we did not update the model parameters of \( E \) during training.

3.3. Attention Scoring Module

This module takes \( f_J \) as input, and weights each words in the full-sentence answer. We used two of these modules, \( S_a \) and \( S_q \). \( S_a \) computes the weights, depending on which word is important for the full-sentence answer, and \( S_q \) computes the weights based on the question. \( S_a \) and \( S_q \) have almost the same structure. Therefore, we show the details of \( S_a \) first, and then describe the difference between \( S_a \) and \( S_q \).

The weight scoring in these modules is based on the attention mechanism used in Transformer [27]. First, we encode each word in the full-sentence and create the full-sentence answer vector \( f_A = \{w^n_1, w^n_2, \ldots, w^n_d\} \in \mathbb{R}^{n \times d_a} \), where \( w^n_i \) denotes the embedding vector of the \( i \)-th word, \( n \) is the length of the full-sentence answer, and \( d_a \) is the dimension of the word embedding vector. To represent the order of the words in the full-sentence answer, we applied positional encoding to \( f_A \). Specifically, before feeding \( f_A \) into scoring modules, we add positional embedding vectors to \( f_A \), same as introduced in BERT [7].

We describe our attention mechanism as a mapping between Query and Key-Value pairs. First, we calculate Query vector \( Q \in \mathbb{R}^{h} \), Key vector \( K \in \mathbb{R}^{h \times n} \), and Value vector \( V \in \mathbb{R}^{h \times n} \).

\[
Q = W_q f_J \\
K = W_k f_A \\
V = W_v f_A
\]

where \( W_q \in \mathbb{R}^{h \times d_I} \), \( W_k \in \mathbb{R}^{h \times d_a} \), \( W_v \in \mathbb{R}^{h \times d_v} \) are learned weights. Then, the attention weight vector \( a_k = \{a_1, a_2, \ldots, a_n\} \in \mathbb{R}^n \), where \( a_i \) is the weighted score of the \( i \)-th word, is computed as the product of \( Q \) and \( K \), as shown below.

\[
a_k = K^T Q
\]

Then, we choose the word that has the highest weighted score as the keyword of the full-sentence answer:

\[
f_k = \arg\max_{a_k} (f_A)
\]

However, the argmax operation is non-differentiable. Therefore, we use an approximation of this operation by softmax with temperature.

\[
f_k = f_A \text{softmax}(\frac{a_k}{\tau})
\]

where \( \tau \) is a temperature parameter and, as \( \tau \) approaches 0, the output of the softmax function becomes one-hot distribution.

\( S_q \) has the same structure as \( S_a \) up to the point of computing the attention weight vector \( a_q \). For the keyword vector, we have the intention to focus on the specific word in the full-sentence answer. Therefore, we use the softmax with temperature. However, for the question vector, we do not need to concentrate on a word. Therefore, we calculate the question vector as the weighted sum of the attention score:

\[
f_q = f_A \text{softmax}(a_q)
\]

Then, we applied one layer feed-forward neural network followed by layer normalization [3] to the output of this module \( f_k, f_q \).

3.4. Decoder

3.4.1 Entire Decoder

In the entire decoder \( \mathbf{D}_{all} \), we reconstruct the full-sentence from the output of the attention scoring modules \( f_k \) and \( f_q \), i.e., \( \mathbf{A}_{recon} = \mathbf{D}_{all}(f_k, f_q) \), where \( \mathbf{A}_{recon} \) denote the reconstructed full-sentence answer. We use an LSTM as the sentence generator. As the input to the LSTM at each step \((x_t)\), we concatenate \( f_k \) and \( f_q \) to the output of the previous step:

\[
x_0 = W_{x_0}[f_k; f_q]
\]

\[
x_t = W_x [f_k; f_q; s_{t-1}]
\]

where \( s_{t-1} \) is the output of the LSTM at the \( t-1 \) step, and \( W_{x_0} \) and \( W_x \) are the learned parameters.

The objective for \( \mathbf{D}_{all} \) is defined by the cross-entropy loss:

\[
L_{all} = -\sum_{t=1}^{n} \log(p(s_t = \hat{s}_{t|t-1}^\text{ans} | \hat{s}_{1:t-1}^\text{ans}))
\]

where \( \hat{s}_{t|t-1}^\text{ans} \) is the ground-truth full-sentence answer.
**Word dropout** We applied word dropout [5], which is a method of masking words of input sentences to LSTM with a specific probability. This forces the decoder to generate sentences relying not on the previous word, but on the \( f_k \) and \( f_q \).

### 3.4.2 Discriminative Decoders

\( D_{all} \) tries to reconstruct the full-sentence answer from \( f_k \) and \( f_q \). Therefore, \( D_{all} \) enables the feature vectors to contain the information of the answer. However, we intend to represent the information of the answer's keyword in \( f_k \), and the information of the question in \( f_q \). Therefore, we design the discriminative decoders, \( D_a \) and \( D_q \), to generate \( f_k \) and \( f_q \), respectively, thus capturing the desired information separately.

\( D_a \) reconstructs the full-sentence answer, and \( D_q \) reconstructs the question. This reconstruction is performed, not with the sentence as the target, but with the BoW features of the sentence as the target. This is because we intend to focus on the content of the sentence, not the sequential information of the sentence. As an alternative, we also considered the reconstruction of the sentence, but that by LSTM is difficult to train. Therefore, we chose BoW features as the target of the reconstruction. BoW feature \( b \in \mathbb{R}^{n_a} \) is represented as a vector whose \( i \)-th elements is \( N_i/L_s \), where \( n_a \) is the size of the vocabulary, \( N_i \) is the number of occurrences of the \( i \)-th word, and \( L_s \) is the number of the words in the sentence.

The input to these discriminative decoders is not only feature vectors, but also auxiliary vectors, which is the additional features that helps the reconstruction. Specifically, the auxiliary vector for \( D_a \) is the average of the word embedding vectors in the question, \( f_Q \), and, for \( D_q \), the auxiliary vector is the image feature \( f_i \).

We build the decoder as the fully-connected layers:

\[
y_a = W_A[f_k; f_Q] + B_A
\]

\[
y_q = W_Q[f_k; f_i] + B_Q
\]

The loss function for the discriminative decoder is the cross entropy loss between the ground-truth BoW features and the predicted BoW features:

\[
L_a = -\sum_{i=1}^{n_a} b_a[i] \log y_a[i]
\]

\[
L_q = -\sum_{i=1}^{n_q} b_q[i] \log y_q[i]
\]

where \( b \) denotes the ground-truth of the BoW features, \( n_a \) and \( n_q \) are the size of the vocabulary for the answer and the question, respectively.

### 3.5. Full Objectives

Finally, the overall objective function for our model is written as

\[
L = \lambda_{all} L_{all} + \lambda_a L_a + \lambda_q L_q
\]

where \( \lambda_{all} \), \( \lambda_a \) and \( \lambda_q \) are hyper-parameters that balance each loss function.

### 3.6. Implementation Details

In the encoder \( E \), we extract image features with a size of \( 2048 \times 14 \times 14 \), from the pool-5 layer of the ResNet152 [10], pre-trained on ImageNet, and apply global pooling to obtain 2048-dimensional features. For the question word encoding, we used 300-dimensional GloVe embeddings [19], which are pre-trained on the Wikipedia/Gigaword corpus\(^1\).

In the attention scoring module, the embedding matrix, to convert each word in the full-sentence answer into \( f_A \), is initialized with the pre-trained GloVe embeddings. The temperature parameter \( \tau \) is gradually annealed using the schedule \( \tau_i = \max(\tau_0 e^{-\frac{i}{\tau}}, \tau_{\text{min}}) \), where \( i \) is the overall training iteration, and other parameters are set as \( \tau_0 = 0.5, \tau = 3.0 \times 10^{-5}, \tau_{\text{min}} = 0.1 \). The LSTM in the \( D_{all} \) has a hidden state of 1024 dimensions. The word dropout rate is set to 0.25.

We used Adam [14] optimizers to train the model, which has an initial learning rate of \( 1.0 \times 10^{-3} \). The loss parameters \( \lambda_{all}, \lambda_a \) and \( \lambda_q \) are set to 1.0, 1,000, 1,000, respectively.

### 4. Experimental Setup

#### 4.1. Dataset

We conducted experiments on two datasets: GQA [11] and FSVQA [23]. In Table 1, we present the basic statistics of both datasets, and those of the VQA v2 dataset for reference.

\[^1\]http://nlp.stanford.edu/projects/glove/

| Dataset | number of QA | avg. len. answer | dataset annotation |
|---------|--------------|------------------|-------------------|
| GQA     | train 943,000 | 6.69             | automatically     |
|         | val 132,062   | 6.70             | automatically     |
|         | all 1,075,062 | 6.70             |                   |
| FSVQA   | train 139,038 | 6.11             | automatically     |
|         | val 68,265    | 6.07             |                   |
|         | all 207,303   | 6.10             |                   |
| VQA v2  | train 443,757 | 1.16             | manually          |
|         | val 214,354   | 1.16             |                   |
|         | all 658,111   | 1.16             |                   |

Table 1. Basic statistics of the dataset we used. For reference, the statistics of VQA v2 dataset are also provided. Note that the value of FSVQA dataset are after pre-processing.
Table 2. Performance of the keyword extraction on GQA and FSVQA. Higher is better for Accuracy and lower is better for Mean Rank. We conducted experiments three times for our methods. Note that TF-IDF is a deterministic algorithm, so we conducted experiments only one time.

| Model                  | GQA       | FSVQA     |
|------------------------|-----------|-----------|
|                        | Accuracy  | Mean rank | Accuracy  | Mean rank |
| TF-IDF                 | 0.275     | 2.86      | 0.278     | 3.22      |
| Ours                   | 0.429 ± 0.03 | 2.04 ± 0.10 | 0.351 ± 0.04 | 2.38 ± 0.14 |
| Ours w/o D_q           | 0.318 ± 0.02 | 2.44 ± 0.04 | 0.298 ± 0.03 | 2.67 ± 0.05 |
| Ours w/o D_a, D_q      | 0.350 ± 0.06 | 3.01 ± 0.49 | 0.347 ± 0.05 | 2.49 ± 0.21 |
| Ours with LSTM D_a, D_q| 0.329 ± 0.01 | 2.36 ± 0.04 | 0.347 ± 0.01 | 3.35 ± 0.11 |

Table 3. The performance per question types. We consider the first two words of each question as the question types, and we show the experimental results for top 10 frequent question types. The left table shows the results of GQA, and the right table shows the results of FSVQA.

**GQA** The GQA dataset [11] contains 22M questions and answers, which are generated from the image scene graph. The questions and answers are automatically generated from the scene graphs, and the answers contain both the single word answers and the full-sentence answers. The 22M questions and answers in GQA have unbalanced answer distributions. Therefore, we use the balanced version of this dataset, which is down-sampled from the original dataset and contains 1.7M questions. As pre-processing, we removed the periods, commas, and question marks.

**FSVQA** The FSVQA dataset [23] contains 370K questions and full-sentence answers. This dataset is built by applying rule-based processing to the VQA v1 dataset [2], and captions in the MSCOCO dataset [15], to obtain the full-sentence answers. There are 10 annotations per question in the VQA v1 dataset. Therefore, one of those annotations is chosen to create full-sentence answers. Specifically, the one annotation that has the highest frequency of the 10 annotations is chosen. If the frequencies are all equal, one annotation is chosen randomly. To obtain the single word answer corresponding to the full-sentence answer, we filter the questions for which we cannot determine the highest frequency annotation. After this process, we obtained 139,038 questions for a train split, and 68,265 questions for a validation split.

### 4.2. Settings

We measure the performance of our model by the accuracy of the keyword and the Mean Rank. Mean Rank is the average rank of the correct keyword when sorting each word in order of the importance score. In addition, we report the accuracy per question types. Here, we categorize all questions according to the first two words, and we show the top 10 frequent question types for this categorization.

We report the keyword extraction result by TF-IDF baseline as a comparison method. We conduct an ablation study to show the importance of D_a and D_q. In addition, we change the reconstruction method, from BoW estimation to the original sentence generation, using LSTM.
### 5. Experimental Results

#### 5.1. Results and Discussions

The experimental results are shown in Table 2. Compared with TF-IDF method, our model achieves the better performance for all metrics and datasets. As can be seen in the results of the ablation study, even without $DA$ and $Dq$, our model has much better performance than TF-IDF. When using LSTM in $DA$ and $Dq$, the accuracy and mean rank becomes worse than in our proposed model, which reconstructs the BoW in those modules. We believe this is because the reconstruction of the sentence with LSTM needs to manage the sequential information in the sentence, and that is more difficult than BoW estimation. We intended to focus on the contents of the sentence. Therefore, the BoW is more suitable for these modules.

The results for the accuracy per question types are shown in Table 3. For most question types, our method shows higher accuracy and lower mean rank than the compared methods. The TF-IDF method tends to choose unusual words as keywords. Therefore, the accuracy drops to almost zero in some question types. However, even in such cases, our method can extract keywords correctly.

#### 5.2. Qualitative Results

We show some examples in Figure 4. The upper side is the result for GQA, and the lower side is the result for FSVQA. And we provide the heatmap of the attention weight score from our attention scoring module, $Sa$. Note that the attention score is normalized to improve visibility. The left two samples are the successful cases and the right two samples are the failure cases. As can be seen even in failure cases, our model can extract keywords correctly.

### 6. Conclusion

In this paper, we proposed the novel task of unsupervised keyword extraction from full-sentence VQA, and we designed the novel model to tackle this task, based on the information decomposition of the full-sentence answer and the reconstruction of the answer and the question. Both qualitative and quantitative experiments show that our model successfully extract the keyword of the full-sentence answer without any keyword supervision.

Our future work includes utilizing extracted keywords to other tasks, such as VQA or object classification. Also, our work could be combined with recent works on VQG. With our work, an intelligent system can ask humans about unseen objects in the image, obtain the answer and learn new knowledge from it, even if the answer consists of more than a single word.

### Acknowledgement

This work was partially supported by JST CREST Grant Number JPMJCR1403, and partially supported by JSPS KAKENHI Grant Number JP19H01115.
References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In CVPR, 2018. 3

[2] Stanislav Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In ICCV, 2015. 1, 3, 7

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016. 5

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In ICLR, 2015. 5

[5] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In CoNLL, 2016. 6

[6] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Jose M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In ICLR, 2017. 2, 3

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, 2019. 5

[8] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In EMNLP, 2016. 1

[9] Yash Goyal, Tejas Khot, Douglas Simmons-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In CVPR, 2017. 1, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016. 6

[11] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In CVPR, 2019. 2, 3, 6, 7

[12] Vahid Kazemi and Ali Elqursh. Show, ask, attend, and answer: A strong baseline for visual question answering. arXiv preprint arXiv:1704.03162, 2017. 3

[13] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In NeurIPS, 2018. 1

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015. 6

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014. 7

[16] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In CVPR, 2017. 3

[17] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In EMNLP, 2004. 3

[18] Ishan Misra, Ross Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. Learning by asking questions. In CVPR, 2018. 1

[19] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In EMNLP, 2014. 6

[20] Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. In the first instructional conference on machine learning, 2003. 3

[21] Stuart Rose, Duve Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. Text mining: applications and theory, 1:1–20, 2010. 3

[22] Tingke Shen, Amlan Kar, and Sanja Fidler. Learning to caption images through a lifetime by asking questions. In CVPR, 2019. 1

[23] Andrew Shin, Yoshitaka Ushiku, and Tatsuya Harada. The color of the cat is gray: 1 million full-sentences visual question answering (fsvqa). arXiv preprint arXiv:1609.06657, 2016. 2, 3, 6, 7

[24] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In NeurIPS, 2015. 3

[25] Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. In CVPR, 2018. 1, 3

[26] Kohei Uehara, Antonio Tejero-De-Pablos, Yoshitaka Ushiku, and Tatsuya Harada. Visual question generation for class acquisition of unknown objects. In ECCV, 2018. 1

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NeurIPS, 2017. 2, 3

[28] Xiaojun Wan and Jianguo Xiao. Single Document Keyphrase Extraction Using Neighborhood Knowledge. 2008. 3

[29] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. In ICLR, 2015. 3

[30] Xing Wu, Zhikang Du, and Yike Guo. A visual attention-based keyword extraction for document classification. Multimedia Tools and Applications, 77(19):25355–25367, 2018. 3

[31] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In ECCV, 2016. 3

[32] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In ICMIL, 2015. 3

[33] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In NAACL-HLT, 2016. 3

[34] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. In CVPR, 2019. 1

[35] Zhou Yu, Jun Yu, Jianping Fan, and Dacheng Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In ICCV, 2017. 1, 3