Image-Based Question Answering with Visual Semantic Embeddings

by

Mengye Ren

Supervisor: Richard S. Zemel
April 10, 2015

Abstract

Computers understanding complex objects in an image and interacting with human through natural language are two open areas of research in computer vision and computational linguistics. A three-year-old child can describe what he/she sees and answer questions regarding to the visual elements, but computers have an extraordinarily hard time handling these simple tasks. Learning and applying visual and semantic knowledge in a computer program will mean a big step in the field of artificial intelligence.

This work, image-based question answering with visual semantic embeddings, aims to let computers jointly learn both vision and natural language through a question-answering (QA) task. In recent years, we have evidenced major breakthroughs in deep learning, machine learning models of deep neural networks, on applications such as object recognition and natural language processing. Inpired by previous approaches using convolutional and recurrent neural networks [1, 2, 3, 4], we combined these learning architectures to create a common embedding space – a visual semantic embedding to represent both images and sentences. We designed models that learn to perform question-answering task within this common embedding space.

Image-based question answering is still a fairly new field. We show that our proposed models achieved more than 1.5 times better accuracy compared to the previous effort [5] on the same dataset [5]. To further evaluate our models and to contribute more data to the research community, we are releasing another dataset that is 20 times larger by converting an image description dataset [6] into question-answer forms. We hope that our work can encourage more follow-up research to achieve better results than ours.

The five chapters will be organized as follows: Chapter 1 will give a brief introduction on the motivation of developing image-based question answering technology. Chapter 2 will cover both the technical backgrounds of our proposed methods, i.e. neural networks and word embeddings, and previous attempts on this very topic. Chapter 3 will explain our neural network medels and our dataset generation algorithms in detail and present the results. Chapter 4 will discuss the significance of our results and Chapter 5 will present some future directions of this research.

Acknowledgements

This thesis would not have been possible without the guidance and support of many people.

First, I would like to thank my advisor Dr. Richard Zemel. Dr. Zemel brought me into the research gate of machine learning when I still did not know anything about it. He suggested such an interesting research project to me that I had never believed I could work on it. Throughout the final year of my undergraduate studies, he always encouraged me on any little progress I made. He gave me many new directions and resources that lead to results of this research.

I owe many thanks to my mentor Ryan Kiros, who is a PhD student in the machine learning group. Ryan has spent a lot of time helping me on this thesis, and has inspired me with a lot of his brilliant ideas on the model design. His energetic attitude also influenced me. Without Ryan's guidance, I certainly would not have achieved as many results.

I also want to thank Dr. Ruslan Salakhutdinov for a helpful discussion on the bidirectional models, and Nitish Srivastava for his support on the Toronto Conv Net. Nitish's excellent Conv Net software and trained models really made my research easy to get started. Of course, my research would not have been feasible without the computing resources. Dr. Zemel did not have any hesitation assigning me with all the computing resources I needed, and I am thankful to Dr. Relu Patrascu for setting up my account and access to the computing clusters.

I would like to thank everyone in the machine learning group at the University of Toronto. I benefited a lot from attending the weekly cookie talks and paper discussions. They really broadened my knowledge on the cutting-edge research in the machine learning community.

I would like to say thank you to all of my friends in Engineering Science. I enjoyed the four years of fruitful undergraduate studies with you and I will be very proud of being an Engineering Science alumnus.

Finally, I would like to thank my parents Yue Li and Lizhi Ren. I thank their love and support to let me explore in the area of my interest and passion.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vision and language are two major inputs of human perception. Everyday, we see images through eyes and listen to speech through ears, and often the two are associated and complementary to each other. Speakers use visual elements such as projector slides to make their lectures easier to understand, and photographers use captions and descriptions to convey deeper thoughts of their photos.

However, compared to humans, computers are extremely poor at vision and language. A human can recognize objects in each part of the image and describe them using his/her own words, but it is still a very difficult task for computers. In recent years, computational models, pariticularly large-scale deep convolutional and recurrent neural networks, have achieved eminent success in the field of object recognition and natural language processing [1, 2, 3, 4]. The achievement from previous research has formed the building blocks of learning higher level representations of vision and language.

In this work, we hope to design machine learning models that can jointly learn vision and language through a question-answering (QA) task. The computer is given a set of images and questions regarding the images, and is expected to output correct answers through a supervised learning scheme. Machines need to combine knowledge from both image and words in order to achieve good performance.

Building an image-based question answering system has profound implication in the field of artificial intelligence (AI). Natural language human-computer interation eliminates the need for prerequisite knowledge of operating electronic devices, which will make technologies more accessible. Exchanging image understanding through natural language can also bring benefits to users with visual disabilities. Image-based question answering technology will trigger countless AI applications and can be directly deployed in robots and wearables to assist humans on daily basis.

Tackling the problem is not an easy task, due to multiple layers of complexity. Not only does the model need to parse the scenes and objects in the image, it also needs to understand the question and generate the relevant answer. Compared to image description generation, image QA requires answers to be targeted to the questions with higher precision. Image QA is still a fairly new field. The current available datasets were just released last December [9]. The authors also presented their approach to this problem, which can be improved by a large margin. In short, there is a lack of large and high-quality datasets as well as competitive models on this problem.

Our goal is to design a machine learning model that learns a representation combining both images and sentences. Unlike the existing approach, we formulate the problem using embedding spaces. First, we need to convert the images and questions into vector forms so that they can be sent as inputs to a neural network. We propose to use the state-of-the-art convolutional neural networks to obtain the visual features, and use recurrent neural networks to obtain sentence-level embedding. We will cover the basics of neural networks and embedding models in Chapter 2. Now, we can reduce the problem into a multi-class classification problem by assuming only single-word answers. We aim to learn a function that outputs either the answer class or the nearest neighbour of the correct answer in the semantic embedding space. Lastly, as the currently available dataset may be insufficient for our approach, we propose an automatic question generation algorithm to synthesize QA pairs from a large image description dataset.

In conclusion, solving the problem of image-based question answering will be a meaningful step in the field of machine learning and AI. We hope this work will invite more research in the future linking computer vision and natural language knowledge in a broader sense.

1.1 Overview of the thesis

Chapter 2 will introduce the background of our proposed methodology, including the very basics of neural networks and word embeddings as well as more recent research on jointly learning image and text with neural network based embedding models. Chapter 3 will describe our proposed approaches and results. We will explain the models in detail with their performance. We will present an algorithm that helps us collecting a large scale image QA dataset. Chapter 4 will discuss the experimental results in depth, and lastly Chapter 5 will propose future work on our line of research.

Chapter 2

Background

Deep architecture and recurrent topology have been significantly developed over the last decade that allow neural networks to automatically extract abstract and useful features. They are as of today the best tools to solve problems such as image detection [2, 10], speech recognition [11, 12], and language modelling [13, 14], outperforming many known machine learning models with hand-crafted feature-extractors. Since our model formulation relies mainly on deep neural networks, this chapter will begin with the basics of neural networks.

2.1 Neural networks

Artificial neural networks are computational models that consist of interconnected adaptive units [15]. A set of input is feeded to each unit, also called artificial neuron, with certain weights assigned.

Each neuron takes the weighted sum of the input, with an optional bias term, and applies an activation function. It is an intuition borrowed from neuroscience: a neuron is fired after the electric current passes some threshold.

$$z = \mathbf{w}^\top \mathbf{x} + b \tag{2.1}$$

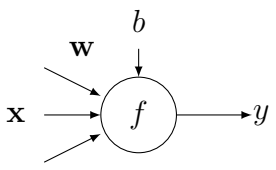


Figure 2.1: An artificial neuron

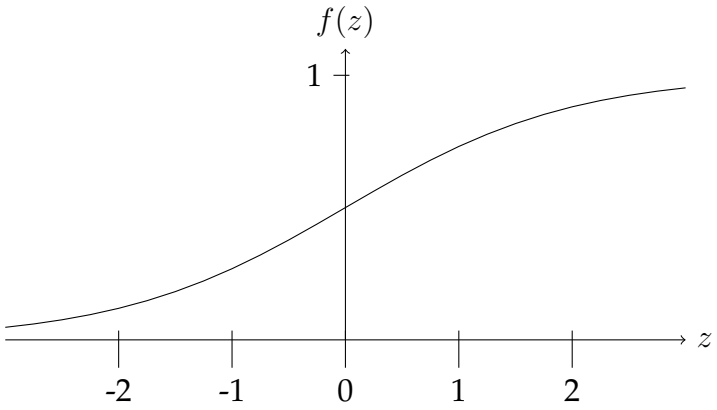


Figure 2.2: The sigmoid function

$\mathbf{x}$ is an input vector with dimension D, and $\mathbf{w}$ is a weight vector also of dimension D, and b is a scalar bias term.

$$y = f(z) = f(\mathbf{w}^\top \mathbf{x} + b) \tag{2.2}$$

A common activation function are the sigmoid function: $f(z) = \dfrac{1}{1+e^{-z}}$, and the hyperbolic tangent function: $f(z) = \tanh(z)$. As shown in 2.2, the sigmoid function is in its "on" state with output to be approximately one on the far right, and in its "off" state with output to be approximately zero on the far left. There is a linear region around the origin where the output climbs from zero to one.

2.1.1 Feedforward neural networks

A feedforward neural network is composed of layers of parallel neurons that process input and transmits data to the next layer. In terms of network topology, a typical feedforward neural network can be decomposed into three types of layers: one input layer, possibly multiple hidden layers, and one output layer. The input layer contains simply the original data samples. The hidden layer and the output layer neurons contain activation functions. We present equations for a single hidden layer network below.

$$\begin{aligned} \mathbf{h} &= f^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \\ \mathbf{y} &= f^{(2)}(\mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}) \end{aligned} \tag{2.3}$$

$f^{(1)}$ and $f^{(2)}$ are activation functions for the hidden layer and the output layer, which can be different functions. $\mathbf{W}^{(1)}$ is the weight matrix from the input layer to the hidden layer with dimension $H \times D$, where D is the input dimension and H is the hidden dimension. $\mathbf{x}$ is the

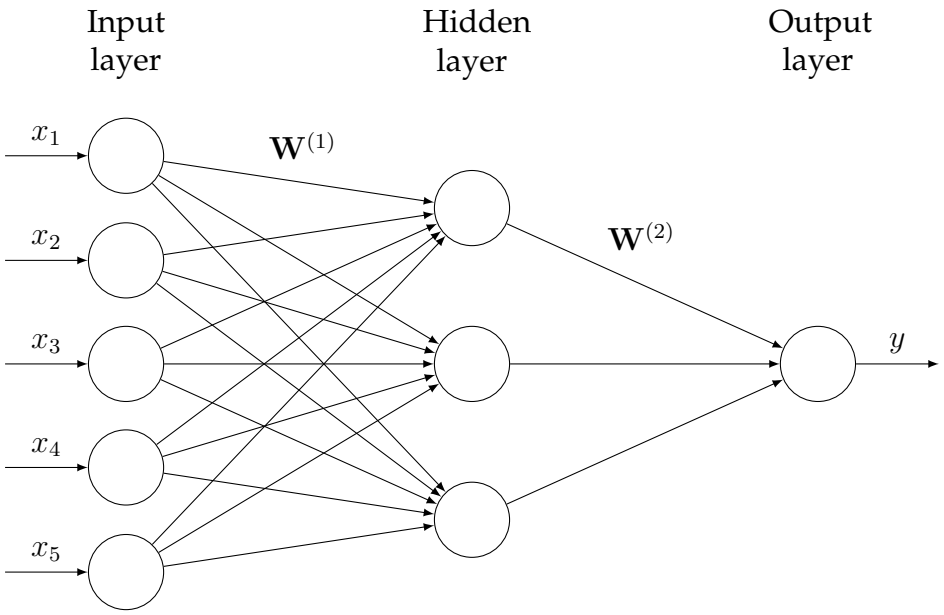


Figure 2.3: A feedforward net with one hidden layer (adapted from [7])

input vector of the neural network and $\mathbf{y}$ is the output vector. With a nonlinear activation function such as the sigmoid function, a single hidden layer neural network is proved to have capability of universal function approximation [16].

2.1.2 Error functions

We first define the error function as some measure of cost from the network output to the desired output. For example, the sum of square error is defined as following:

$$E(\mathbf{y}) = \sum_i (t_i - y_i)^2 \tag{2.4}$$

$\mathbf{y}$ is the network output, and $\mathbf{t}$ is the desired output, or the ground truth in the training dataset. i denotes the index of the output element. For classification problems, usually the output is the probability of an example belonging to a class. It is more common to use the cross-entropy error function for this type of problems. For binary classification, the cross-entropy is defined as following:

$$E(\mathbf{y}) = \sum_i -t_i \log(y_i) - (1 - t_i) \log(y_i) \tag{2.5}$$

t and y have the same definition as above. $t = 1$ means the example belongs to class 1 and 0

means class 0. If the output of the network is exactly the same with the ground truth, the value of the error function will be zero, and if the output is exactly the opposite, the value of the error function goes to infinity. For multiple class problems, the activation function is usually a softmax function, also called normalized exponential:

$$\mathbf{y} = \frac{e^{\mathbf{z}}}{\sum_i e^{z_i}} \tag{2.6}$$

where z is the activation value before output and i denotes the index of the output element. In this way, the class with the maximum probability is preserved and the function is differentiable. The multi-class cross-entropy error function is defined as following:

$$E(\mathbf{y}) = \sum_i -t_i \log(y_i) \tag{2.7}$$

$\mathbf{t}$ is a vector with all zeros except one entry with the correct class of the example, and $\mathbf{y}$ is a vector of the network predicted class probability. This can be seen as an extension to the two-class case.

2.1.3 Training neural networks

The training of neural networks is usually achieved through backpropagation. Equivalently speaking, learning can be regarded as a minimization problem of some error function, and backpropagation a first-order gradient descent method. The calculus chain rule is used to derive the error derivative with regard to weights in each layer. For each update, we take a small step of the gradient towards lower value of the error function.

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_i \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{w}} \tag{2.8}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t \tag{2.9}$$

γ is also called the learning rate or the step size. If we have want to further backpropagate to previous layers of neurons, we need to use the chain rule again to calculate the error derivative with regard to the input of the current layer.

$$\frac{\partial E}{\partial x_j} = \sum_i \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial x_j} \tag{2.10}$$

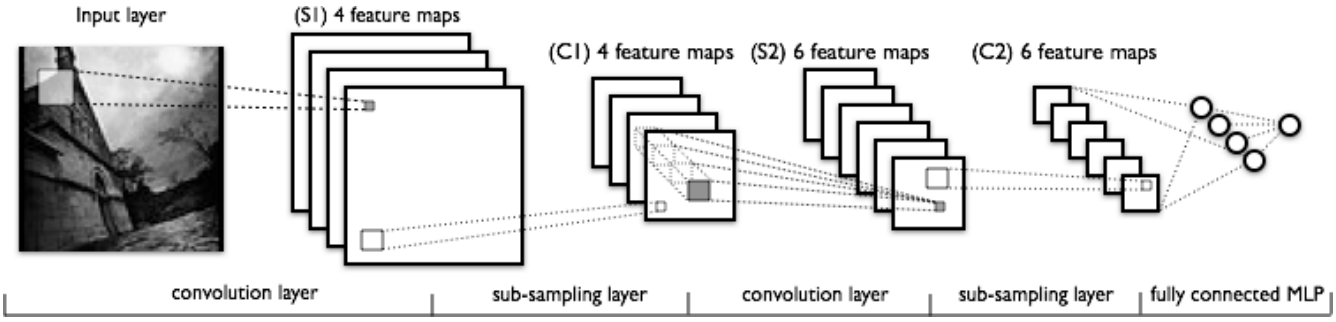


Figure 2.4: A convolutional neural net [8]

More neural networks training techniques can be found in Appendix A.

2.1.4 Convolutional neural networks (CNNs)

One obvious advantage of neural network is fighting against the curse of dimensionality. For example, a small image patch of 28 pixel $\times$ 28 pixel with binary pixel values will generate $2^{28\times 28} = 2^{784}$ different possible images. In the classic MNIST handwritten digit dataset [17], there are only sixty thousand training examples. Neural networks can learn the input-output relation with a continuous function that has the ability of generalize to unseen data.

However, one hidden layer neural networks still has its limitations. To achieve larger learning capacity, a large number of hidden neurons are required, but this also increases the total number of weights need to be learned. With only small number of examples, a neural network quickly overfits the training set, and perform poorly on the test set. Weight sharing is an important technique to preserve the generalization ability. The idea is to force some weights the same across the entire learning process, which is often done through summing all the error derivatives for those shared weights and update them with the sum. Convolutional neural networks [17], is exactly built on top of the the idea of weight sharing. LeNet [17], one of the first CNNs, achieved the highest MNIST handwritten digit classification rate at the time.

The flipped version of the shared weights can be regarded as small square shape image filters (Figure 2.5), and the computation to each hidden neurons is equivalent to a 2-D convolution on the original image. Filtered images are then passed to a Rectified Linear Unit (ReLU) ($f(z) = \max\{0, z\}$) to obtain nonlinear activation to certain features. Next, to reduce data dimensionality and to make the model robust to translational invariance, a max pooling layer effectively downsamples the intermediate image with the maximum value in each subregion. Lastly, a fully connected softmax layer makes class predictions with each unit representing the probability of input data belonging to certain classes. Such deep architecture has achieved the state of the art image classification and detection results in recent object classification and detection competitions [2, 10]. Figure 2.4 shows a typical CNN architecture.

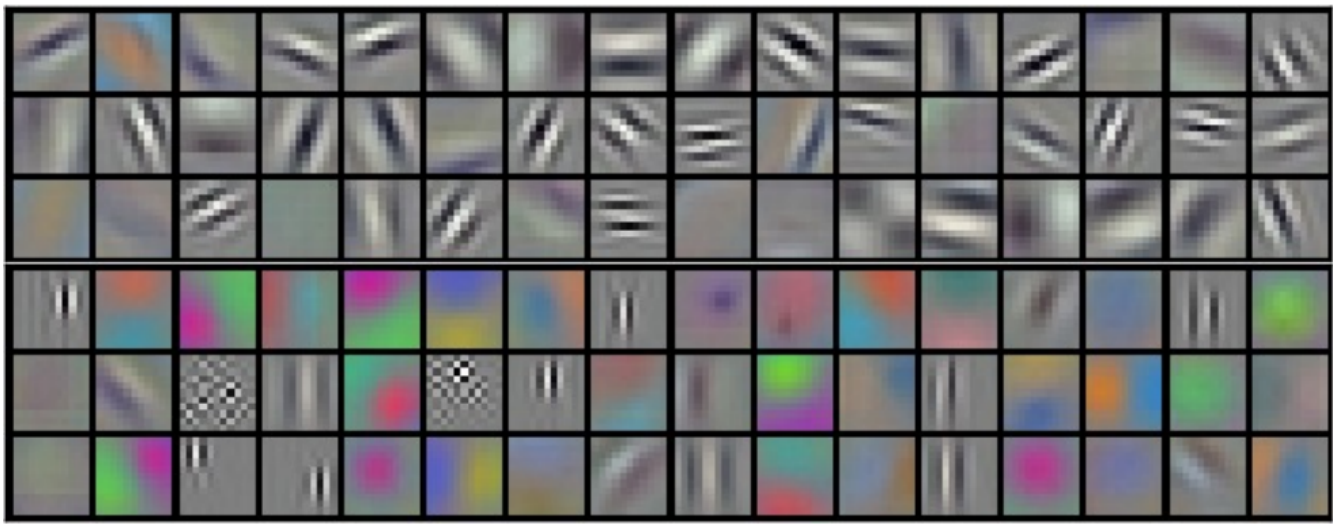


Figure 2.5: Automatically learned low-level image filters [2]

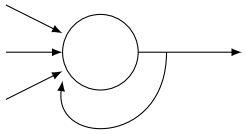


Figure 2.6: A recurrent neuron with a feedback loop

2.1.5 Recurrent neural networks (RNNs)

Recurrent neural network is a type of network that sends the output back to the input neurons, creating feedbacks in the network topology. It is designed to learn time series input because time dependency features can be effectively captured by the feedback loop. Recently, recurrent neural networks have been applied on many natural language tasks, such as language modelling, machine translation, etc. This thesis will focus on one popular type of recurrent neural network called long short-term memory (LSTM), and we will use it as our sentence model.

2.1.6 Long short-term memory (LSTM)

Long short-term memory (LSTM) [18] is a major variant of recurrent neural networks. The training of regular RNNs is notoriously difficult because of the vanishing and exploding gradients problems. LSTM is designed to address these training issues. One of the main feature of LSTM is the constant error propagation in its memory unit. This is brought by the linear activation of the memory content at every timestep. But this also results in a complex structure as in shown Fig 2.7.

LSTM has a central memory unit (c) with three different types of multiplicative gating that controls the flow of information in the memory. The value of each gating at every time step is

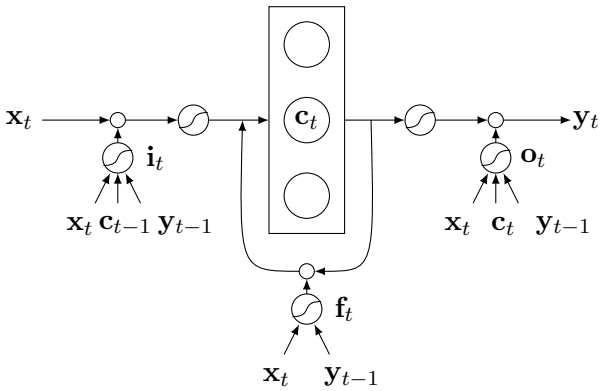


Figure 2.7: Long short-term memory

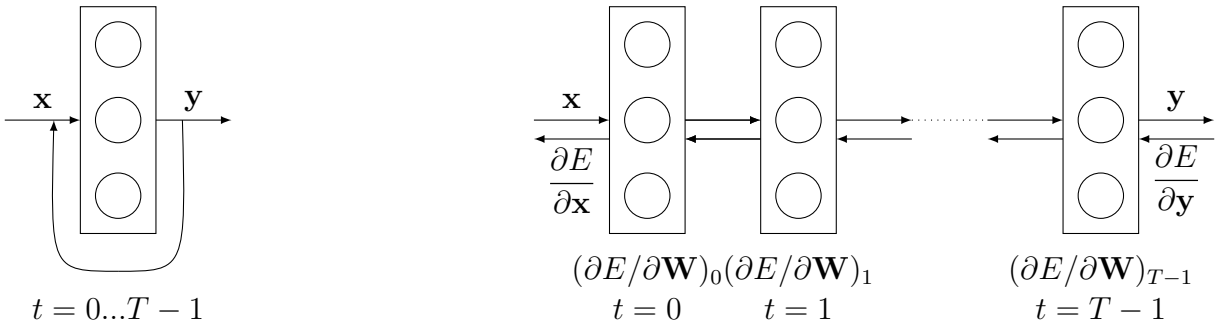
determined by a weighted sum of the current input ($\mathbf{x}_t$), the current memory content ($\mathbf{c}_t$), and the current output ($\mathbf{y}_t$). A sigmoid function maps the weighted sum to a gating value between zero and one. The memory of each timestep is then updated by the forget gate ($\mathbf{f}_t$) multiplying with the memory content from the previous time step, and the input gate ($\mathbf{i}_t$) multiplying with the current input through a hyperbolic tangent (tanh) activation function. Lastly, the output of the neural network can be obtained by the output gate ($\mathbf{o}_t$) multiplying with the current memory through another hyperbolic tangent activation. The equations of LSTM are summarized in Equation 2.11.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{iy}\mathbf{y}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fy}\mathbf{y}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{z}_t &= \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cy}\mathbf{y}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{z}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oy}\mathbf{y}_{t-1} + \mathbf{W}_{oc}C_t + \mathbf{b}_o) \\
\mathbf{y}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{2.11}$$

Here, σ denotes the sigmoid function, and $\odot$ denotes component-wise product. $\mathbf{x}_t$ is a D dimensional input vector at time t. $\mathbf{W}_{ix}, \mathbf{W}_{iy}, \mathbf{W}_{ic}$ are $M \times D$ dimension weight matrices, which maps the input dimension to the memory dimension. Other weight matrices are of dimension $M \times M$. $\mathbf{y}_t$ and $\mathbf{c}_t$ are output and memory content vector of dimension M at time t. And $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o$ are bias vectors of dimension M.

2.1.7 Training of RNNs

The training of RNNs is typically achieved by backpropagation through time (BPTT) [19]. The idea is to imagine the recurrent network as a chain of feedforward network by unfolding the recurrent connections through time. The error derivatives can then be evaluated as in normal



Recurrent neural network

Unfolded through time

Figure 2.8: Back-propagation through time

feedforward networks. In the end, the errors to weight connections are summed up through time.

$$\frac{\partial E}{\partial \mathbf{W}} = \sum_t \left(\frac{\partial E}{\partial \mathbf{W}} \right)_t \tag{2.12}$$

More RNN training techniques can be found in Appendix A.

2.2 Word embedding

In the previous sections, we have covered some neural network basics and popular configurations. In the following sections, we will introduce some recent applications in the field of natural language processing and question answering using word embeddings and recurrent neural networks.

A short definition for word embedding is a dictionary which assigns each vocabulary a high dimensional vector that captures semantic similarity. Typical English vocabulary size in NLP applications are between 10,000 and 100,000, and following the Zipf's law[20], most of the vocabularies are rare words. In a typical n-gram analysis, our computational and storage resource quickly run out because of the sparsity of word combinations. To fight against this curse of dimensionality, one would like to share knowledge between similar words without redundant training examples for each word. For example in a language modelling task, if we have seen the sentence "the cat is walking in the bedroom", then the other sentence "the dog was running in a room" should also be assigned with higher probability [21]. While traditional non-parametric n-gram models fail to capture the associativity of similar words, neural networks have the ability to approximate the probability distribution over a continuous hyperspace. But to enable this task, we need to convert every word into its vector representation. We can first randomly

initialize the word embedding matrix as the first layer of a neural network, and then train the network to predict the next word given a sequence of previous words. In the end, the first layer can be then used as a fine-tuned word embedding. The trained embedding vectors are distributed representation of words that preserve semantic similarity. Earlier word embeddings are mostly trained from neural language models, until recently researchers found that language modelling is not necessary to obtain a good word embedding.

2.2.1 Skip-gram embedding model

The skip-gram model [22] is a very successful word embedding model. In the paper, the authors used two methods to train word vectors: the continuous bag of words (CBOW) and the skip gram model. The former aims to predict the missing word given a bag of context words, and the latter aims to predict the context surrounding a given single word. The model is streamlined to one single linear mapping which significantly shortens the training time down to a few minutes on a standard computer, as opposed to a few weeks for neural language models. It is also astonishing to see algebraic operators can be directly applied on word vectors, and can preserve the intended meaning. For example, "king" + "woman" - "man" $\sim$ "queen" [22]. The software package released with the paper is called "word2vec." A lot of subsequent research directly uses this software to train general purpose or problem specific word embeddings.

2.3 Jointly learn image and word

Human beings are capable of associating objects of their different forms. For example, when people see a text "pink pig", they are likely to imagine a picture of a pink pig in their head. Likely, when people see blue sky in their eyes, they are likely to have some word-level mental activity. It is an intriguing question to ask if we can engineer a "visual-semantic joint embedding," an abstract vector space in which the word "sky" and a picture of sky has very small distance. It is very tempting to build such an embedding on top of the existing convolutional neural networks and word embeddings. In the work "Deep Visual-Semantic Embedding Model" (DeViSE) [23], researchers used model proposed earlier [24] but replaced the image feature extractors with the last hidden layer convolutional net [2] (just before the softmax layer). The image representation is then mapped to the word embedding space [22] with a linear transformation. The model is trained to correctly classify images, and the joint embedding outperforms the standard convolutional neural net by a large margin on a test set with a large number of unseen classes. The reason is that even though the exact images of the classes are unseen, the class names have already had their word similarity captured in the word embedding space, and

thus beat the simple fully connected softmax layer in standard convolutional neural networks. Not only does joint embedding improves the result for image object recognition tasks, it also encourages better cooperation of visual and semantic knowledge. The work by [4] proposed models that generate descriptions based on an image input using visual semantic embeddings, and the generated sentences have much better plausibility and accuracy compared to earlier results [25, 26]. In their work, they used LSTM introduced earlier as part of the encoder of training descriptions as well as the decoder of generated descriptions. These recent works have shown us very strong feasibility to jointly learn image and text in a joint embedding space.

2.4 Image-based question answering

The following sections will focus on the background of question-answering and recent research specifically related to image-based question answering. Traditional QA builds entity relationships within a large text collection, or a corpus, using natural language processing (NLP) techniques. Large-scale question answering has a long history, mostly initiated via the Text Retrieval Conference (TREC). Despite some eminent successes in building such systems, such as [27], building a knowledge graph requires a great amount of engineering and is heavily dependent on the knowledge base or the search engine, and is also hard to generalize to unseen entities. There have been ongoing efforts of using embedding-based model and recurrent neural networks on question-answering tasks. [28] showed that their memory-based recurrent neural network can perfectly answer questions that involve long time dependency in a paragraph.

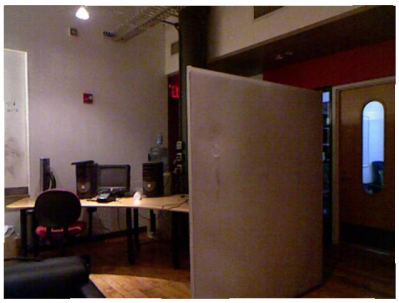
Image-based question answering has its own differences compare to text-based question answering: First, the question is centered on the image. Therefore, the model needs to acquire both visual and semantic knowledge. Second, unlike traditional question answering where the training data contains almost the exact answer with certain predicates revealing its relation with the question, here the training data provides no direct information to the answer, which is solely contained the unseen test image. However, the training data provides very strong clue of what catagories of answers are expected. For example, the model should be able to conclude from the training data that "chair" is a more probable answer than "window" for a question like "What is around the table?" These differences actually make the problem unique from traditional QA tasks.

2.4.1 DAQUAR dataset

In 2014, Malinowski et al. released a dataset with images and question-answer pairs. It is called DAtaset for QUestion Answering on Real-world images (DAQUAR) [9]. All images are from

Q7: what colour is the ornamental plant in fornt of the fan coil but not close to sofa ? -red

(a) Hard to locate the object



Q20: how many black objects are on the desk ? -three

(b) Counting ambiguity, could be three to five

Q1129: what is behind the tap ? -blinds

(c) Object ambiguity, could be a window or blinds

Q5: what is under the white cabinet on the right side of the oven and and on the left side of the fridge ? -sink

(d) Lengthy description of spatial relations

Q1423: what is the colour of the furniture ? -brown

(e) Colour ambiguity due to dim lighting

Q1467: what is on the night stand ? -paper

(f) Object is too small

Figure 2.9: Various types of difficult questions in the DAQUAR dataset

the NYU depth v2 dataset [29], and are taken from indoor scenes. Human segmentations, image depth values, and object labellings are available in the dataset. The QA pairs were created by human crowdsourcing. The original train-test split ratio is about 3800:3300. The QA data has two sets of configuration: the 37-class and the 894-class dataset, differed by the number of object classes appearing in the questions. There are mainly three types of questions in this dataset:

1. Object type
2. Object colour
3. Number of objects

Qualitatively speaking, a large proportion of the questions are very hard to answer. Figure 2.9 summarizes the sources of difficulties. Since DAQUAR is the only publicly available image-based QA dataset, it is one of our benchmarks to evaluate our models.

2.4.2 Previous attempt

Together with the release of the DAQUAR dataset, [5] presented an approach which combines semantic parsing and image segmentation. In the natural language part of their work, they used a semantic parser [30] to convert sentences into latent logical forms. The parser is specially designed for QA tasks, and can be trained well where the true dependency tree is missing from the training data. In the image part, they evaluated different latent logical forms of the question with multiple segmentation. They obtained the multiple segmentations of the image by sampling the uncertainty of the segmentation algorithm. Their model is based on a Bayesian formulation that every logical form and image segmentation has certain probability. To make the inference step of their algorithm scalable, they chose to sample from the nearest neighbours in the training set according to the similarity of the semantic parsing.

While their model seems to handle a number of spatial relations, their predefined set of possible predicates constrains the question form and the environment. Moreover, the quality of answers also depends on the accuracy of the automatic image segmentation algorithm. Lastly, inferring all possible predicates between every pair of objects does not scale to larger dataset. The embedding space models introduced earlier have significant differences with their approach, and we hope that our embedding space model will outperform their model in terms of answer accuracy and similarity.

Chapter 3

Methods and Results

3.1 Problem restatement

We consider every examples in a image QA dataset as a triplet of an image vector, a question of a sequence of word indices, and an answer of a sequence of word indices. We start off by assuming that the answers are all one-word answers, i.e. the length of the answer sequence is always one. The assumption is established based on more than 98% of the data entries in DAQUAR have only one-word answers. Our goal is to learn a function that takes the input of an image vector and a sequence of word indices, and emits the output of the answer word.

3.2 Our models

3.2.1 GUESS model

One very simple baseline is to predict the mode based on question type. For example, if the question contains "how many" then output "two." This baseline actually works unexpectedly well in DAQUAR dataset.

3.2.2 BOW model

Another baseline is to use the last hidden layer of the Oxford convolutional neural net (4096 dimension) [1] as an image feature extractor, and use wording embeddings from word2vec. To obtain the sentence-level embedding, we simply sum all the word vectors, i.e. bag-of-words (BOW). We concatenate the image vectors and the word vectors, and send the combined feature into a softmax layer.

3.2.3 LSTM sentence model

Due to the sequential nature of natural languages, researchers have been looking for ways to model sentences using recurrent neural networks. There has been increasing interests in using LSTM to model the embedding vector for a whole sentence. Recently, [3] uses LSTM as both encoders and decoders in machine translation, and achieved BLEU score [31] better than the traditional phrase-based models [32]. In our work we use LSTM as our sentence embedding model. At every timestep we input a word vector to the LSTM, and we use the output values of the LSTM at the last timestep as our embedding space representation of the question asked.

3.2.4 Image-word model

We started off the experiment by directly building on top of the LSTM sentence model. In this experiment, we designed a model called the "image-word model" because it treats the image as one word of the question. We borrowed the idea of treating the image as the first word of the question from caption generation by [33]. The difference with caption generation is that here we only output the answer at the last time step.

1. We used the last hidden layer of the Oxford convolutional neural net (4096 dimension) [1] as our visual embedding model.
2. We use the word2vec embedding model (skip gram) trained from Google News [22] as our frozen semantic embedding (300 dimension).
3. We then treated the image as if it is the first word of the sentence. In order to map the image embedding to our semantic embedding space, we used a linear transformation layer of dimension 4096×300 (similar to [23]).
4. The last time step of the LSTM output is passed to a softmax layer. The final output is a 68-dimension vector representing the probability of the answer being each possible answer classes.

We used the 37 class DAQUAR dataset, so the question and answers will only focus on 37 object classes. We also filtered dataset down to only one-word answers. This simplification trims the possible set of answers to 63 words (including unknown class "UNK"). To evaluate the model, we used the plain answer accuracy as well as the Wu-Palmer similarity (WUPS) measure [34, 5]. The WUPS calculates the similarity between two words based on their longest common subsequence in the taxonomy tree. The similarity function takes in a threshold parameter. If the similarity between two words is less than the threshold then zero score will be given to the candidate answer. It reduces to plain accuracy when the threshold equals to 1.0. Following [5], we measure all the models in terms of plain accuracy, WUPS at 0.9 threshold, and WUPS at 0.0

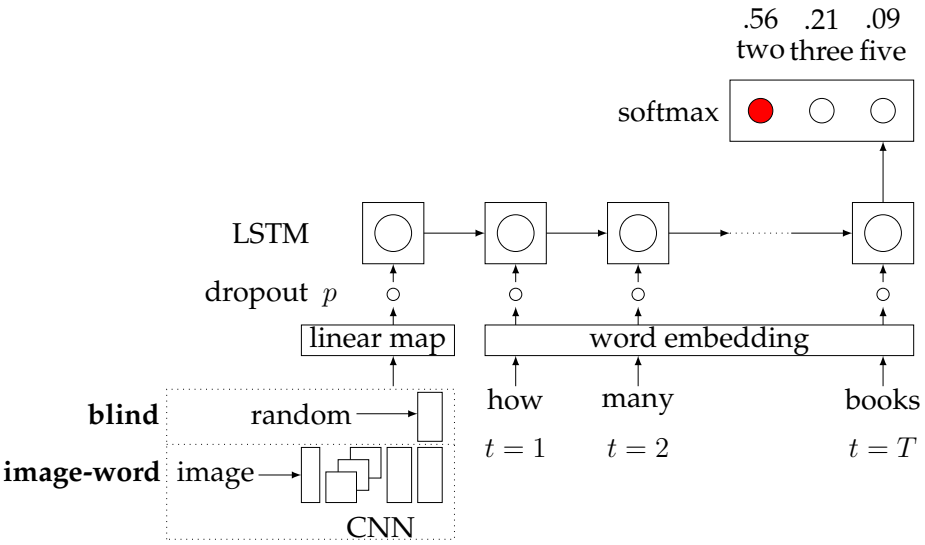


Figure 3.1: Image-word model and blind model

threshold.

To evaluate the results, we also designed a baseline model to avoid over-interpretation. In this model, the images are replaced by randomly generated vectors instead of meaningful graphics. We call this baseline the "blind model."

3.2.5 Bidirectional image-word model

The image-word model, which takes in words sequentially in a linear fashion, may not be powerful enough to handle complex sentence structures in the question. At the last timestep it needs to output the answer immediately after reading the last word of the question. In some cases where the hint is revealed at the last timestep, the single direction model may fail to change direction because the previous words have made the model very confident of an answer already. The bidirectional image-word model aims to fix this weakness. It contains two LSTMs. The first one takes the image as the first word and then processes the words sequentially just like the previous model. Moreover, the second LSTM processes the words in a reverse order, together with the hidden state of the first LSTM at that timestep. At the last timestep, the second LSTM takes in the image again. So intuitively, the first timestep gives an "impression" of the image, and after knowing the sentence, it checks the image again to confirm the answer. The second LSTM "sees" the whole sentence at every timestep, so this model is thought to be better because it considers the entire question for a longer period of time.

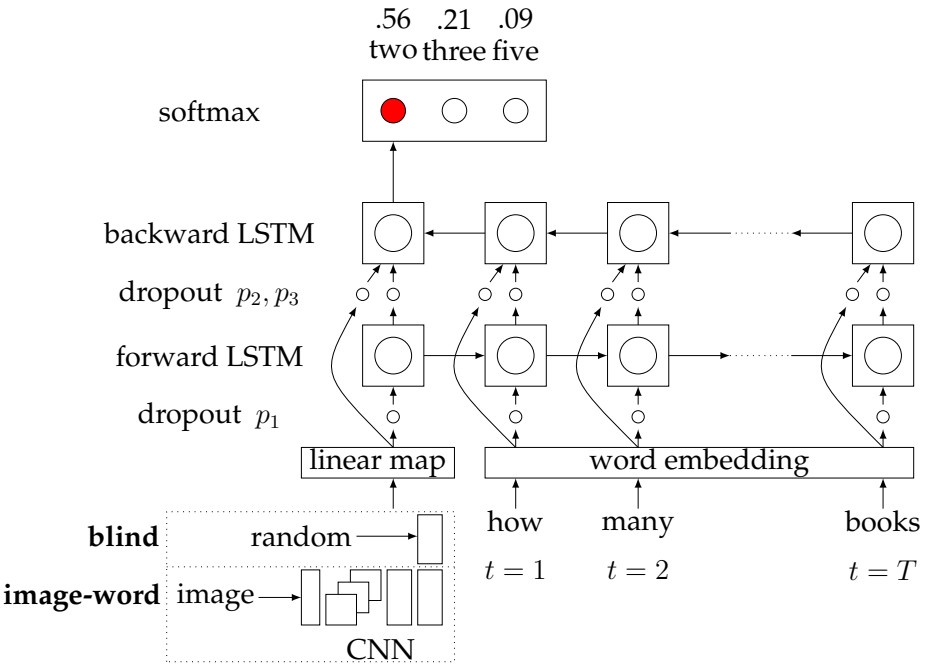


Figure 3.2: Bidirectional image-word model and blind model

3.2.6 Image-word ranking model

The softmax layer can be replaced by other types of loss. The ranking loss is very popular in recent question answering literature because it is very fast to train. This type of model outputs a vector that is the nearest neighbour of the answer word in the semantic embedding. The error function uses the pair-wise ranking loss [24], defined below:

$$\sum_{\mathbf{y}} \sum_{i \neq j} \max\{0, \alpha - s(\mathbf{y}, \mathbf{a}_j) + s(\mathbf{y}, \mathbf{a}_i)\} \tag{3.1}$$

$\mathbf{y}$ and $\mathbf{a}$ are both vectors in the answer embedding space. $\mathbf{y}$ is the output of the model, and $\mathbf{a}_j$ is the correct answer for the question, and $\mathbf{a}_i$ is one of any possible answers. $s(\cdot, \cdot)$ denotes the similarity measure of two vectors. It is usually implemented as the cosine similarity in high dimensional spaces:

$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \tag{3.2}$$

This error function will penalize the model if the right answer is not winning over other wrong answers by a certain margin α. It is a widely used in ranking image descriptions [4]. The ranking loss effectively replaces the softmax layer, and the hidden dimension of the LSTM need to match with the word embedding dimension.

Q193: what is the largest object ?
Image-word: table (0.576)
Blind: bed (0.440)
Ground truth: table

(a) The question gives no extra clue of the class of the object. The visual model recognizes the correct object through the shape.

Q212: what is the object left of the room divider ?
Image-word: toilet (0.3973), sink (0.1367), towel (0.1323)
Blind: refridgerator (0.5318)
Ground truth: door

(b) The visual model recognizes the correct scene.

Q1615: how many pictures are there on the wall ?
Image-word: three (0.4518)
Blind: three (0.6047)
Ground truth: seven

(c) No evidence shows that image-word model learns how to count.

Figure 3.3: Direct comparison between the image-word model and the blind model.

3.2.7 DAQUAR results

Table 3.1 summarizes the performance of our proposed models and our baselines. First, our models win by a large margin compared to the results from the publisher of the DAQUAR dataset. Second, we argue that most gain in accuracy results from a good sentence embedding, because the blind model has almost the same accuracy compared to the image-word model. Third, the bidirectional model further improves the performance by a small amount. Lastly, the ranking loss model does not outperform the softmax model. Although the training is much faster even with a smaller learning rate, it achieves lower accuracy because it tends to overfit the training set too quickly.

In Figure 3.3, we further discovered the some weak visual ability by a direct comparison of test examples. We observed that the image-word model seems to perform better on questions centered on dominant objects or dominant colours, but does not seem to learn how to count.

3.3 COCO-QA dataset

From DAQUAR results, we see that although our models have improved the answer accuracy by a lot compared to the previous attempt, the blind version of the model can do almost equally well, suggesting that the image features from the CNN are not very useful. Since 1500 images are a very small dataset, maybe it is worthwhile for us to build a larger dataset so that the neural networks can be trained more robustly.

Table 3.1: DAQUAR results

	Accuracy	WUPS 0.9	WUPS 0.0
2-IMGWD[a]	**0.3276**	**0.3298**	0.7272
IMGWD[b]	0.3188	0.3211	**0.7279**
IMGWD-RK[c]	0.2787	0.2782	0.7074
BLIND[d]	0.3051	0.3069	0.7229
RANDWD[e]	0.3036	0.3056	0.7206
BOW[f]	0.2299	0.2340	0.6907
GUESS[g]	0.1785	0.1823	0.6874
MultiWorld [5]	0.1273	0.1810	0.5147
HUMAN	0.6027	0.6104	0.7896

[a]Bidirectional image-word model
[b]Image-word model
[c]Image-word ranking model
[d]Single-direction blind model
[e]Single-direction model with random word embedding
[f]Bag-of-words model
[g]Guess "two", "white", and "table", depending on question type

Manually annotating pictures with QA pairs requires large amount of time, capital, and tools. We propose to use currently massively available image description dataset and convert descriptions into QA forms. This method is cheap, fast, and scalable; however the drawback is that the quality of automatically generated questions is unforeseeable. There lacks a measure to test how sensical the questions are overall.

We used recently released Microsoft Common Objects in COntext (MS-COCO) [6] dataset, which contains 100K images and 500K sentence descriptions, and converted the descriptions into QA pairs. We only considered three types of questions: object, number, and colour, and only single-word answers.

3.3.1 Question conversion algorithms

Here we present an algorithm that converts sentences into different types of questions.

Object-type questions

First, we consider asking an object using "what" or "who". This involves replacing the actual object with a "what" in the sentence, and then transforming the sentence structure so that the "what" appears in the front of the sentence. The algorithm below assumes that the input is a sentence in the form of a syntactic tree, and the output is a question in the form of a syntactic

tree.

The entire algorithm has the following stages:

1. Split long sentences into simple sentences (see Algorithm 1).
   For question generation task, sentences need not to be very descriptive, and shorter the original sentences are, more readable the questions will be. Moreover, the wh-words that are found in sub-sentences or clauses will less likely be able to perform wh-movement [35]. Here we only consider a simple case that is when two sentences are joined together with a conjunctive word "and". We split the orginial sentences into two independent sentences. For example, "There is a cat and the cat is running." will be split as "There is a cat." and "The cat is running."

2. Change indefinite determiners to definite determiners (see Algorithm 2).
   Asking questions on a specific instance of the subject requires changing the determiner into definite form "the". For example, "**A** boy is playing baseball." will have "the" instead of "a" in its question form: "What is **the** boy playing?".

3. Traverse the sentence and identify potential answers (see Algorithm 3).
   We traverse the tree structure and identify answers that belong to certain word categories. Then we replace the answer with a question word "what" or "who" depending on the object type.

4. Perform wh-movement with some constraints (see Algorithm 4).
   For English language, questions tend to start with interrogative words such as "what" and "who". The algorithm needs to move the verb as well as the "wh-" constituent to the front of the sentence. However, not all sentences allow such transformation. In this work we consider the following constraints:

   (a) A-over-A principle
   The A-over-A principle was first discovered by Chomsky [36]. For example, "I am talking to John and **Bill**" cannot be transformed into "*Who am I talking to John and" because "Bill" is an noun phrase (NP) constituent that is under another NP "John and Bill". Therefore, the child NP cannot be removed from the parent NP in the wh-movement.

   (b) Clauses
   Except for a few cases, interrogative words in clauses usually cannot be moved to the front of the sentences. For example, "I am riding a motorcycle that **Bill** wanted." cannot be transformed into "*Who am I riding a motorcycle that wanted.". In the future, there is a need to separate the original sentences into two: "I am riding a motorcycle." and "Bill wanted the motorcycle.". For now, wh-word in the clauses will

terminate the wh-movement process and our algorithm will only output sentences like "I am riding a motorcycle that **who** wanted?".

The entry point of the overall algorithm is shown in Algorithm 5. Figure 3.4 illustrates these procedures with tree diagrams. We used WordNet [37] and NLTK software package [38] to lemmatize verbs and to get noun categories. We used Stanford parser [39] to obtain syntatic structure of the original sentence.

Algorithm 1 Split compound sentences

input: Root of the syntactic tree of the original sentence
output: List of roots of syntactic trees of split sentences

```
1: procedure SplitCC(root)
2:     node ← root.children[0]                    ▷ Search directly from "S" node
3:     if node is "S" and it has more than 3 children then
4:         if All children are "S" or "CC" and "CC"s are always in between two "S"s then
5:             return each "S" child
6:         end if
7:     end if
8: end procedure
```

Algorithm 2 Replace indefinite determiners to definite determiners

input: Root of the syntactic tree with wh-word in the original place
output: Root of the syntactive tree with wh-word in the front

```
1: procedure SwitchDefDet(root)
2:     node ← DFS(root, "NP")          ▷ Depth-first search for the subject in the sentence
3:     node ← DFS(node, "DT")          ▷ Depth-first search for the determiner of the subject
4:     if node.text = "a" or node.text = "an" then
5:         node.text ← "the"
6:     end if
7:     return root
8: end procedure
```

Number-type questions

To generate "how many" types of questions, we follow a similar procedure as the previous algorithms, except a different way to identify potential answers. This time, we need to extract numbers from original sentences. Splitting compound sentences, changing determiners, and wh-movement parts remain the same. Algorithm 6, 7 shows an detailed algorithm.

Algorithm 3 Identify object-type answers

input: Root of the syntactic tree of the original sentence
output: List of roots of the syntactic trees with answers replaced by "what"

procedure TRAVERSEWHAT(root, results)
  for child ∈ node.children **do**
    TRAVERSEWHAT(child, results)
  end for
  if node is"NP" and node's children contains a noun is of category
    animal, artifact, body, food, object, plant, possession, shape, person **then**
    answer ← noun
    whatNode ← Node(class="WP", text="who/what", children=[ ])
    Replace answer with Node(class="WHNP", text="", children=[whatNode])
    Insert the modified root into roots list
  end if
end procedure

Algorithm 4 Wh-movement

input: Root of the syntactic tree with wh-word in the original place
output: Root of the syntactive tree with wh-word in the front

procedure WH-MOVEMENT(root)
  Check if the sentence contains a verb-phrase. If no, terminate.
  Check if "WHNP" is under "SBAR" or "NP". If yes, terminate.
  verbFront ← **null**
  if the verb is any tensed form of "be" or "have done", or is a modifier e.g. "will" **then**
    verbFront ← verb
    Remove verb from its parent
  else
    verbFront ← Node(class=verb.class, text="does/do/did", children=[])
    verb ← LEMMATIZE(verb)
  end if
  Remove WHNP from its parent
  S_old ← root.children[0]
  S_old.children.insert(verbFront, 0) ▷ Insert verbFront as the first child of S_old
  S_new ← Node(class="S", text="", children=[WHNP, S_old])
  root.children ← [S_new]
  return root
end procedure

Algorithm 5 Generate object-type questions

input: Root of the syntactic tree of the original sentence
output: List of roots of the syntactic trees of generated questions

```
1: procedure ASKWHAT(Root)
2:     SWITCHDEFDET(root)
3:     for r ∈ SPLITCC(root) do
4:         roots ← [ ], TRAVERSEWHAT(r, roots)
5:         for r2 ∈ roots do
6:             yield return WH-MOVEMENT(r2)
7:         end for
8:     end for
9: end procedure
```

Algorithm 6 Identify number-type answers

input: Root of the syntactic tree of the original sentence
output: List of roots of the syntactic trees with answers replaced by "how many"

```
1: procedure TRAVERSEHOWMANY(root, results)
2:     for child ∈ node.children do
3:         TRAVERSEHOWMANY(child, results)
4:     end for
5:     answer ← null
6:     if node is "NP" and node's children contains number then
7:         howNode ← Node(class="WRB", text="how", children=[ ])
8:         manyNode ← Node(class="JJ", text="many", children=[ ])
9:         parent ← Node(class="WHNP", text="", children=[howNode, manyNode])
10:        Replace the number node with parent
11:        Insert the modified root into results list
12:    end if
13: end procedure
```

Colour-type questions

Compared to the previous two question types, colour-type questions are much easier to generate. It only requires locating the colour adjective and the noun which the adjective attaches to. Then it simply forms a sentence "What is the colour of the object" with the "object" replaced by the actual noun. The tree traversal is also similar to previously presented algorithms so the implementation details are omitted.

3.3.2 Reducing rare answers

Here we impose a hard constraint such that all answers must at least appear L times in the dataset. In the 6.6K dataset, $L = 5$, and in the full dataset, $L = 20$. The reason of imposing this

Algorithm 7 Generate number-type questions

input: Root of the syntactic tree of the original sentence
output: List of roots of the syntactic trees of generated questions

```
1: procedure AskHowMany(Root)
2:     results ← [ ]
3:     SwitchDefDet(root)
4:     for r ∈ SplitCC(root) do
5:         roots ← [ ], TraverseHowMany(r, roots)
6:         for r2 ∈ roots do
7:             yield return Wh-movement(r2)
8:         end for
9:     end for
10: end procedure
```

constraint is that otherwise we would end up with many answers that only appear once in the training set. This will significantly increase the difficulty of the task.

3.3.3 Reducing common answers

Besides creating more data, the other main motivation here is to reduce the performance of a model that simply guesses for the three most common answers (i.e. mode guessing) for each type of question (e.g. "man", "two", "white"). Therefore, additional constraint needs to be imposed to reduce the common answers. The probability of enrolling an question-answer pair (q, a) is:

$$p(q,a) = \begin{cases} 1 & \text{if count}(a) \leq U \\ \exp\left(-\dfrac{\text{count}(a) - U}{2U}\right) & \text{otherwise} \end{cases} \tag{3.3}$$

where $U = 100$ in both the 6.6K dataset and the full dataset. This will penalize answers for appearing more than U times while still preserve the rank of the most common answers.

3.3.4 Question statistics

With the automatic question generation technique described above, we gathered a dataset from the MS-COCO dataset. We name this dataset to be COCO-QA dataset. Table 3.2 presents some important statistics of COCO-QA, with comparison to DAQUAR. The full dataset is considered to be harder than DAQUAR, because it has more possible answer classes and lower guess baseline accuracy. However, COCO-QA contains 60 times more images and 20 times more question-answer pairs, so it is more suitable to deep learning approaches.

Table 3.2: General statistics of COCO-QA and DAQUAR

Dataset	# Images	# Questions	# Answers[a]	GUESS accuracy[b]
DAQUAR-37[c]	795+654[d]	3825+3284	63	0.1885
DAQUAR-894[e]	795+654	6149+5076	412	0.1180
COCO-QA 6.6K	3.6K+3K	14K+11.6K	298	0.1157
COCO-QA Full	80K+20K	177K+83K	794	0.0347

[a]Number of answers present in the training set plus the unknown "UNK" class
[b]Guess the three most common answers depending on question types
[c]Trimmed to include only single-word answer
[d]"+" denotes train-test split
[e]Trimmed to include only single-word answer

Table 3.3: COCO-QA and DAQUAR question type break-down (numbers inside the brackets denote the proportion with regard to the entire dataset)

Dataset		Object	Number	Colour	Total
DAQUAR-37	Train	2814 (0.7357)	903 (0.2361)	108 (0.0282)	3825 (1.0000)
	Test	2406 (0.7326)	779 (0.2372)	99 (0.0301)	3284 (1.0000)
DAQUAR-894	Train	5131 (0.8344)	903 (0.1469)	115 (0.0187)	6149 (1.0000)
	Test	4194(0.8262)	779(0.1535)	103(0.0203)	5076(1.0000)
COCO-QA 6.6K	Train	10759 (0.7641)	1134 (0.0805)	2187 (0.1553)	14080 (1.0000)
	Test	8850 (0.7645)	939 (0.0811)	1878 (0.1544)	11576 (1.0000)
COCO-QA Full	Train	155888 (0.8824)	7213 (0.0408)	13561 (0.0768)	176662 (1.0000)
	Test	73483 (0.8875)	3181 (0.0384)	6135 (0.0741)	82809 (1.0000)

3.3.5 Question quality

Assessing the exact quality of the generated question is not feasible. We either need human workers to manually go through the entire dataset, or we need to have a program that knows a perfect grammar. There are mainly two sources of errors: first is caused by incorrect sentence structures produced by the Stanford parser and second is caused by our algorithms. By going through the first 100 samples of the generated questions ourself, we found that 81% of the questions are grammatical, and 63% of the questions are meaningful. This precision can be further tuned up by employing more strict question generation rules (i.e. lowering the recall). We will discuss a few future work items in Chapter 5. But the two raw statistics above show that we have a functional question generation algorithm that can be used for our QA learning task.

3.3.6 Learning results

We applied the same models, the single directional and the bi-directional image-word model, on the COCO-QA dataset. Table 3.4 summarizes the results. We further computed the accuracy in each question category in Table 3.5.

Table 3.4: COCO-QA results

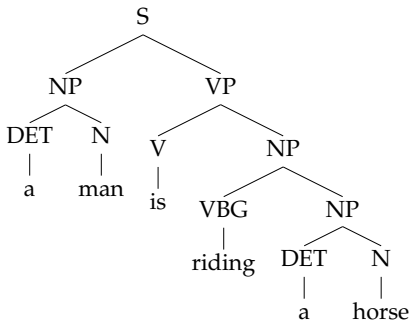
	6.6K			Full		
	Acc.	**WUPS 0.9**	**WUPS 0.0**	**Acc.**	**WUPS 0.9**	**WUPS 0.0**
2-IMGWD	**0.3358**	**0.3454**	**0.7534**	**0.3208**	**0.3304**	**0.7393**
IMGWD	0.3260	0.3357	0.7513	0.3153	0.3245	0.7359
BOW	0.1910	0.2018	0.6968	0.2365	0.2466	0.7058
RANDWD	0.1366	0.1545	0.6829	0.1971	0.2058	0.6862
BLIND	0.1321	0.1396	0.6676	0.2517	0.2611	0.7127
GUESS	0.1157	0.1608	0.6893	0.0347	0.0609	0.6376

Table 3.5: Full COCO-QA accuracy per category break-down

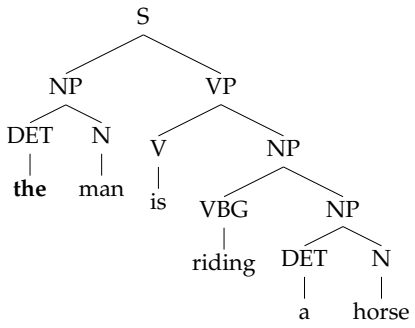
	Object	**Number**	**Colour**	**Total**
2-IMGWD	**0.3139**	**0.4233**	0.3600	**0.3208**
IMGWD	0.3078	0.4189	**0.3604**	0.3153
BOW	0.2273	0.3398	0.3027	0.2365
BLIND	0.2389	0.3713	0.3539	0.2517
RANDWD	0.1828	0.3836	0.2885	0.1971
GUESS	0.0138	0.3024	0.1446	0.0347

First, the bi-directional image-word model is still the best model, which is consistent with our DAQUAR results. Second, the model has reasonable visual ability answering object-type questions, as the bi-directional model wins 7% compared to the blind model. Since the CNN was originally trained for the ImageNet challenge [40], and the distribution of object types in COCO is closer to ImageNet, the image feature from the CNN is more useful here than in DAQUAR. Third, the model "guesses" the colour of an object almost entirely from a good word embedding and sentence modelling, since there is only 0.6% gain in colour-type questions by adding image features. Fourth, the model has very limited counting ability since the guess baseline (always guessing "two") achieves 30.24%, whereas the bidirectional model achives 42.33%. In short conclusion, in our new COCO-QA dataset, we show that our models perform much better than simply guessing the modes. In particular, our model is doing reasonable on object-type questions even with a large number of answers classes present, but it clearly needs further improvement in counting and recognizing colours.

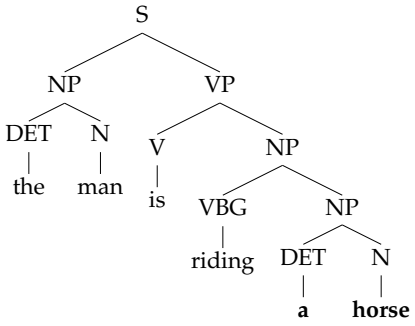
(a) Step 1: Parse the sytactic structure.



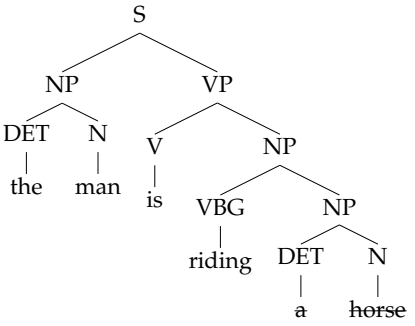
(b) Step 2: Change determiner.



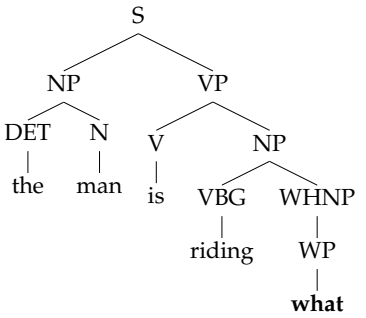
(c) Step 3: Find the answer of the question.



(d) Step 4: Replace the answer with "what".



(e) Step 4: Replace the answer with "what".



(f) Step 5: Perform WH-fronting.

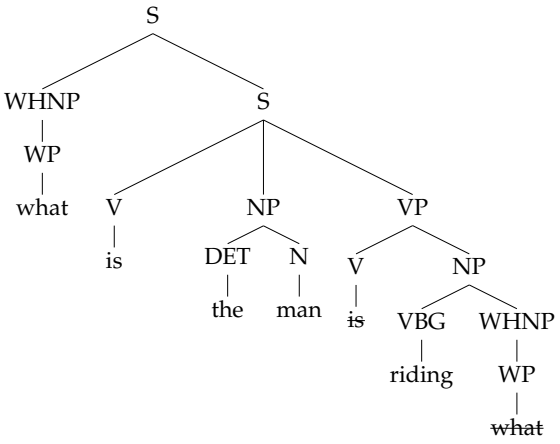


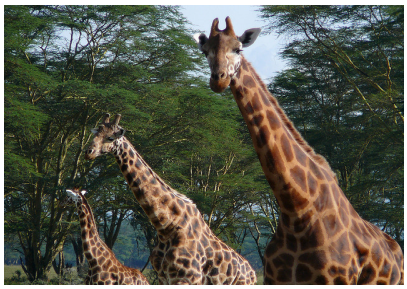
Figure 3.4: Example: "A man is riding a **horse**" => "**What** is **the** man riding?"

Q683: how many military jet fighter is flying in formation alongside a 1 military propeller pilot ?
2-IMGWD: four (0.402)
Ground truth: one

Q7746: how many airplanes are flying up in the air in formation ?
2-IMGWD: four (0.334)
Ground truth: four

Q8473: how many jets in the sky in a flying formation ?
2-IMGWD: four (0.329)
Ground truth: eight

Q8325: how many giraffes are partially hidden in brush and sticking their necks out ?
2-IMGWD: three (0.302)
Ground truth: two



how many giraffes in grassy field next to trees ?
2-IMGWD: three (0.361)
Ground truth: three

Q13103: how many giraffes standing together and eating near some trees ?
2-IMGWD: three (0.317)
Ground truth: five

Figure 3.5: Counting ability of the image-word model: the outputs are strongly dependent on the types of object asked.

Q44: an open food container box with how many unknown food items ?
2-IMGWD: two (0.258)
Ground truth: four

Q47: the beautiful dessert waiting to be shared by how many people ?
2-IMGWD: two (0.450)
Ground truth: two

Q1016: how many motor cycle racers posing on their parked bikes ?
2-IMGWD: two (0.460)
Ground truth: three

Figure 3.6: Counting ability of the image-word model: the model outputs “two” when it is not certain about the objects.

Q10371: what is the colour of the bananas ?
2-IMGWD: green (0.488)
Ground truth: green

Q16954: what is the colour of the bananas ?
2-IMGWD: green (0.488)
Ground truth: yellow

Q5594: what is the color of the bananas ?
2-IMGWD: green (0.488)
Ground truth: yellow

Q18455: what is the colour of the cat ?
2-IMGWD: black (0.227)
Ground truth: yellow

Q19096: what is the colour of the cat ?
2-IMGWD: black (0.227)
Ground truth: black

Q19096: what is the colour of the cat ?
2-IMGWD: black (0.227)
Ground truth: grey

Figure 3.7: Colour recognition ability of the image-word model: the outputs are strongly dependent on the types of objects described in the question. Even worse, since the questions are the same, the output probability are the same regardless of the images. The output class is consistent with the blind model.

Q5141: what parked on the side of the road ?
2-IMGWD: motorcycle (0.226)
BLIND: truck (0.110)
Groud truth: motorcycle

Q5017: what is sitting on top of someones bed ?
2-IMGWD: computer (0.303)
BLIND: cat (0.229)
Groud truth: bicycle

Q2472: what is sitting on the handle bar of a bicycle ?
2-IMGWD: cat (0.460)
BLIND: cat (0.256)
Groud truth: bird

Figure 3.8: Object recognition ability of image-word model: when there is less clue from the text, the image-word model seems to ouput better answers. But it still fails on many situations, so the gain is still mainly from language understanding.

Chapter 4

Discussion

In the previous chapter we presented two models, the single directional image-word model and the bi-directional model with some variations. We tested these models on two datasets, DAQUAR [5] and COCO-QA which is automatically generated from MS-COCO image description dataset. We showed that the model has demonstrated non-trivial ability as it wins over a number of baselines, particularly the previous attempt on DAQUAR dataset [5]. By replacing the actual image with random numbers, we further evaluated the usefulness of the image features in these models. In this section, we will break down our discussion into four parts: model selection, dataset selection, semantic knowledge, and visual knowledge.

4.1 Model selection

4.1.1 LSTM sentence model

The LSTM is shown to be a very effective sentence model. It not only adds up the word semantics as does in the bag-of-word (BOW) model, but also dynamically weights every word when input to the memory cell without any handcrafted heuristics (e.g. TF-IDF [41]). The image-word model always outputs the correct raw class of answer in the top 10 answers: "what" corresponds to an object; "who" corresponds to a person, "how many" corresponds to a number, and "what is the colour" corresponds to a colour. Note that "how many", "what", and "who" not always appear in the beginning of the sentence as some image descriptions cannot perform wh-fronting algorithm. Although this might seem to be a simple task, the BOW model in fact did not perform very well on it. Compared to BOW, the LSTM distinguishes different orderings of the words. For example, BOW cannot distinguish questions like "what is on the red chair beside the table ?" and "what is on the red table beside the chair ?". Based on the re-

sults presented in the last chapter, we argue that LSTM is an excellent model for sentence-level embedding.

4.1.2 Word embedding

In all of our experiments, we either take a word2vec embedding trained from Google News [22] or use a frozen randomly initialized word embedding. We did not experiment weight-update on the randomly initialized word embedding because given the amount of data in DAQUAR, it is very easy to overfit in the training set. However, with more data as in the case of COCO-QA, it may be worthwhile to train a task-specific word embedding. Based on the experimental results, we observed that word2vec embedding performs better than random initialized embedding, especially on the COCO-QA dataset, where there are more variety of object classes. This shows the effectiveness of pre-trained word embedding.

4.1.3 Single direction or bi-direction

In all the experiments, the bi-directional model wins over the single directional model in terms of both plain accurarcy and WUPS. However, the gain is usually under 1%. If we take into consideration that the bi-directional model takes two times the training time, then the benefit of bi-directional model is not very significant.

4.1.4 Softmax cross entropy or ranking loss

As discussed in Section 3.2.7, although the ranking loss makes the training faster, the results does not outperform cross entropy because the model overfits the training set too quickly. It is also questionable whether it makes sense to use generic word embedding trained from Google News as the target vector. Both softmax cross entropy and ranking loss are two effective loss function for training such objectives, and will be tested as one of our hyperparameters in the future work.

4.2 Dataset selection

DAQUAR dataset is the first dataset on the image question-answering tasks. It is human generated, so all the questions are understandable and well written. We noted in Section 2.4.1 that some questions are too hard to answer, as human can only achieve 60% accuracy on this dataset. Moreover, with only 1500 images and 7000 questions, this dataset is too small for deep learning

approaches. Lastly, the fact that a simple GUESS baseline can perform very well suggests that we need to have more even answer distribution so that the mode takes less proportion of the entire distribution.

We then created a dataset called COCO-QA that aims to directly address to the three issues mention above. All questions are generated from image description, so the questions will mostly focus on salient objects in the image instead on small details. It contains 20 times more questions and 60 times more images, which is more suitable to deep learning approaches. With a specially designed "mode damper" that reduces the most common answers, the GUESS baseline can only get 3.47% accuracy on the full dataset. On the other hand, it is hard to evaluate the quality and the difficulty of the questions. More future improvement will be discussed in the next chapter.

The results from Table 3.4 actually show that our model scales to a large number of object types and answer classes. Both single directional and bi-directional models achieve over 30% accuracy on the full COCO-QA dataset, which has 794 possible answer classes and over 13K vocabularies.

Currently, despite the large number of questions and images, the types of questions are quite limited: they are object, number, and colour. However, the fact that our model does not perform very well on counting and colour questions warns us whether it is meaningful to extend to even broader ranges of questions too quickly at this stage. Although counting objects and colour recognition can be addressed using problem-specific computer vision techniques, to work towards a generic QA framework, it is worthwhile to investigate generic machine learning models that perform well on those specific tasks first before we expand to a larger variety of questions.

4.3 Visual and semantic knowledge

Based on the results shown in Table 3.1 and Table 3.4, the major gain of our model is achieved through a better language model. In Figure 3.5 and Figure 3.7, same objects described in the questions are always assigned with same colours and numbers. Only in rare occasions such as Figure 3.8 where the model does not have sufficient hints from the questions, the image features help the model get the correct answer. Even worse, exactly same questions will output very similar answer probabilities. We conclude that in these cases, our model is not using the image signals at all, which is possible when the LSTM "forgets" the image features of the first timestep over the time through the "forget gate", and for bi-directional models, the image features at the last timestep are rejected by the "input gate". It is also possible that because we used the last hidden layer of the CNN, same object classes already have very similar and problem-specific

feature vector representations, so the image features are too similar to be used for asking other types of questions. The similarity in image feature vectors may explain the differences in terms of plain accuracy on counting and colour recognition between the blind model and the image model in Table 3.5, because the image models can simply predict the mode conditioned on the words appeared in the question without taking in too much noise from the image features. It will be future work for us to explore other alternatives to represent images.

Chapter 5

Future Work

5.1 Exploration of current models

Although we have presented a functional model on the image QA task, we acknowledge that due to the time constraints of this thesis, there remain a number of alternatives on the current models that are unexplored. We list a few modifications can be experimented in the future:

1. Modification on CNN
   In all of our experiments, we only tried the last hidden layer of the CNN trained on object recognition as our frozen image features. We did not let back-propagation occur in the CNN because we believed that DAQUAR dataset is too small to fine-tune the CNN. However, with 20 times more data as in the case of COCO-QA dataset, fine-tuning the CNN becomes possible. It is also worth experimenting whether the last hidden layer is really the most useful image features for our task. In a recent work [42], the authors show how the images features transition from general purpose to problem specific in a CNN. As discussed in the last chapter, the image features for the same class of object may be too similar to discriminate against different numbers and colours. Following the same methodology in [42], we can vary our model by using frozen lower layers of the CNN, and fine-tuned upper layers of the CNN.

2. Image features on every timestep
   One of the problems that we are encountering is that the image features are "forgotten" as the LSTM is fed with too much word semantic information. One way to mitigate this is through biasing the word vectors at every timestep, so that the entire sentence vector will be biased by the image input.

3. Different image projection layers for the first and last timestep
   In the bi-directional image-word model, we used same linear map to transform the im-

age features into the joint visual-semantic embedding space. Alternatively, we can use different projection layers. The intuition is that the model will extract different information from the model when it does not know about the question and when it has seen the question.

5.2 Visual attention model

As people are answering a question on some picture, they often skim through the entire picture and focus on the point of interest that is closely related to the question. For example, a question like "what is on the table?" expects them to first locate the object "table" and then look above the object. Knowing where to look at is very important in this task. And as shown in the DAQUAR dataset, there are many questions that do not focus on the main object in the image. THe experimental results suggest that there is an acute need for the model to learn to capture visual details. Therefore we believe that building an alignment between words and the attention of the image will help the model achieve better performance.

Fortunately we can borrow the recent breakthroughs in attention modelling [43] to help us on this task. The input of the attention model is the convolutional layer features in the CNN [1], rather than the fully connected layer. In this way, it will preserve the local information to a larger degree. The attention function gives a probability distribution (visual attention) over the entire image at every time step. The visual attention, the image and the question word are input to the recurrent neural network, and the output of the network forms a feedback loop to the attention function through a multilayer feedforward network. The attention model has the state-of-the-art scores on image description generation, and we believe that incorporation of this model will likely boost our answer accuracy.

5.3 Better question generation algorithms

We have presented our proposed algoritms that generate three types of questions from sentences. As discuessed before, there are mainly two sources of errors. First is caused by incorrect sentence stryctyres produced by the Stanford parser and second is caused by our algorithms. We list the following future work items to improve the quality of our question generation algorithms.

1. Sentence fragments
   Many descriptions in MS-COCO are not complete sentences. For example "A bathroom with toilet" or "A man riding a bike". Future work is needed to fill in missing verbs

in those incomplete sentences. More importantly, the Stanford parser erroreously tags nouns as verbs in many sentence fragments. For example, the word "sink" is tagged as verb in a sentence fragment such "A bathroom sink (n.)", and the sentence fragment will be converted to a question "what sink (v.) ?".

2. Split clauses
   Just like the way we separate composite sentences that have conjunction word "and", we can also separate clauses from the original sentence. As explained previously, shorter sentence will likely to be converted the good questions.

3. Identify phrasemes as one answer entity
   Sometimes multiple words represent a single entity, also called phrasemes. For example, "fire hydrant" should be regarded as one single word in the answer. Otherwise, the answer will be "hydrant" alone.

4. Better quality metric
   As mentioned before, it is very hard to evaluate the quality of auto-genereated questions. In fact, automatic question generation could be an interesting research topic on its own. It is possible to design some quality metric similar to BLEU [31], which compares the similarity between auto-generated questions and human generated questions.

It is also interesting to investigate how does the quality of the questions affect the performance of the models. For humans, asking a nonsensical question would certainly not lead to the "correct" answers but for machines, since they are trained on this dataset that contains certain proportion of nonsensical questions, maybe they can "learn" a way to output the "correct" answer.

5.4 Longer answers and free-form answers

Up until now we have assumed that there are only one-word answers, but ideally we would like to consider longer answers as well. Inspired by [3], we could possibly extend the length of our answers by using an RNN decoder such as LSTM. We can borrow the idea in [4], to either have a template to generate a word in certain word category at a time, or generate a list of free-form answer candidates and select the best candidate. It should be noted that free-form answers also need a better quantitative metric to measure answer quality, in term of its accuracy and language soundness. Lastly, this would need a larger varieties of question types for the model to learn variations in language generation. This will be our long-term goal in the image QA research.

Chapter 6

Conclusion

The goal of this research is to let a computer answer a question regarding an image. This type of problem has not been much explored yet. There is a lack of very large datasets as well as powerful models.

On the modelling side of our work, we proposed to use the last hidden layer of CNN trained on object recognition as an image feature extractor and LSTM as a sentence embedding model. We mapped the image features into the joint embedding space with a linear transformation. We found that our model outperforms the previous attempt by 20% on the same dataset with only one-word answers. We concluded that our proposed models of using joint visual-semantic embedding space achieved the best results so far. However, the gain is mostly from good natural language understanding. And the model is not doing very well on counting and colour recognition questions.

As the existing dataset might be too small and biased for training large scale neural networks, we decided to take an automatic approach to convert an existing image description dataset into QA forms. This results in a new dataset that is 20 times larger than the previous one. Compared to the previous dataset DAQUAR, our new dataset is much harder for a baseline such as "guessing the modes" to achieve good results; however ours is easier for human because it mostly consists of questions that focus on salient objects. Moreover, given the size of our dataset, it is more suitable to deep learning approaches. However, it still suffers from ungrammatical and unmeaningful questions, and is expected to be improved in the future.

In conclusion, image-based question is a hard problem. We hope that our models and datasets presented in this thesis can shed light on future research on this topic.

Appendix A

Neural networks training techniques

A.1 Momentum

Momentum is a common optimization technique applied on training neural nets. It takes a portion of the last weight update to propagate forward, which significantly speeds up the training by many times. Usually the momentum value is taken between 0.5 and 0.99.

$$\Delta \mathbf{w}_{t+1} = -\gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t + \alpha \Delta \mathbf{w}_t \tag{A.1}$$

In the equation above, α is the momentum term.

A.2 Weight regularization

Overfitting occurs when the model performs too well on the training set but generalizes poorly on the test set. Regularization is often applied to prevent overfitting. Common weight regularization methods include square weight loss, absolute weight loss, etc. In the case of square weight loss, we penalize the norm of the weight vector, to prevent each weight element from growing too large. In the backpropagation of the neural networks, the penalty can also be computed in the derivative.

$$\Delta \mathbf{w}_{t+1} = -\gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t + \alpha \Delta \mathbf{w}_t - \lambda \mathbf{w} \tag{A.2}$$

λ here is defined to be the weight regularization constant.

A.3 Dropout

Another way to prevent overfitting in deep neural networks with a large number of weights is through dropout [44]. During training, dropout disables a random set of hidden units with certain dropout probability, so the network cannot rely on co-adaptation of hidden activations to learn the desired output. During training time,

$$f(X) = D \odot X, D \sim \text{Bernoulli}(p) \tag{A.3}$$

D is a Bernoulli random vector of zeros and ones, with dropout probability $(1-p)$. $\odot$ denotes component-wise product. During testing, the output is rescaled by the dropout probability to match with training statistics.

$$f(X) = pX \tag{A.4}$$

A.4 Gradient control in training RNNs

A.4.1 Gradient clipping

Exploding gradient is a common issue in the training of RNNs. Instead of having a uniform learning rate for any gradient update, the gradient clipping method [45] imposes a hard constraint on the norm of the gradient update. We rescale the gradient if the the norm of the gradient is larger than the constraint.

$$\partial E/\partial W_{\text{new}} = \begin{cases} \partial E/\partial W, & \text{if } \|\partial E/\partial W\| \leq C, \\ \partial E \partial W / \|\partial E/\partial W\|, & \text{otherwise.} \end{cases} \tag{A.5}$$

A.4.2 Weight clipping

Weight clipping is equivalent to adding a hard constraint to the weight vector. It rescales the weight vector whenever its norm surpasses the contraint. This effectively prevents the weights from exploding during the training.

$$\min E(w) \quad \text{subject to} \quad g(w) = ||w||^2 - U \leq 0 \tag{A.6}$$

A.5 Dropout in RNNs

Unlike in feedforward networks, dropout in RNNs cannot be simply applied on any hidden units [46]. In fact, dropout is only shown to be effective if applied on non-recurrent connections, i.e. the input neurons. In case of multiple stacked LSTMs, a dropout rate of 20% in the first LSTM input, and 50% in the following inter-stage inputs are shown to work the best [46].

Appendix B

Model training details

B.1 Image-word model

We list here the hyperparameters of the model by layers.

1. Image embedding layer
   Using Oxford net [1] last hidden layer 4096 dimension features
   Learning rate: 0.0
2. Word embedding layer
   Using word2vec skip gram embedding vectors trained from Google News [22]
   If the word is not found then initialize uniform from [-0.21, 0.21]
   Learning rate: 0.0
3. Image linear transformation layer
   Input dimension: 4096
   Output dimension: 300
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 100.0
4. Input dropout layer
   Dropout rate: 20%
5. LSTM layer
   Input dimension: 300
   Memory dimension: 150

Initialization of $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o$: 1.0
Initialization of $\mathbf{b}_c$: 0.0
Initialization of other weights: uniform [-0.05, 0.05]
Learning rate: 0.8
Momentum: 0.9
Gradient clipping: 0.1
Weight clipping: 100.0
Weight regularization: square loss, λ = 5e-5

6. Softmax layer
Input dimension: 150
Initialization: uniform [-0.05, 0.05]
Learning rate: 0.01
Momentum: 0.9
Gradient clipping: 0.1
Weight clipping: 10.0 in DAQUAR, 15.0 in COCO-QA 6.6K, 50.0 in COCO-QA Full
Weight regularization: square loss, λ = 5e-5

B.2 Blind model

All hyperparameters are kept the same with the image-word model, except that we replaced the 4096 dimension frozen image feature input with 4096 dimension frozen random vectors, uniformly initialized from range [-0.05, 0.05].

B.3 Image-word ranking model

Layers before softmax are kept the same with the image-word model, except that the LSTM hidden dimension is now 300 to match with the vector length.
Then the cosine similarity layer computes the cosine similarity of the output of the LSTM and the answer vector bank which is directly retrieved from word2vec.
The answer that has the largest similarity with the output vector is selected to be the predicted answer.

B.4 Bidirectional model

We list here the hyperparameters of the model by layers.

1. Image embedding layer
   Using Oxford net [1] last hidden layer 4096 dimension features
   Learning rate: 0.0
2. Word embedding layer
   Using word2vec skip gram embedding vectors trained from Google News [22]
   If the word is not found then initialize uniform from [-0.21, 0.21]
   Learning rate: 0.0
3. Image linear transformation layer
   Input dimension: 4096
   Output dimension: 300
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 100.0
4. Forward LSTM input dropout layer
   Dropout rate: 20%
5. Forward LSTM layer
   Input dimension: 300
   Memory dimension: 150
   Initialization of $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o$: 1.0
   Initialization of $\mathbf{b}_c$: 0.0
   Initialization of other weights: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 100.0
   Weight regularization: square loss, λ = 5e-5
6. Backward LSTM input dropout layer
   Dropout rate: 50%
7. Forward-to-backward dropout layer
   Dropout rate: 50%
8. Backward LSTM layer
   Same as forward, except input dimension is now 450 because it takes both the word vector and the output of the forward LSTM.

9. Softmax layer
   Input dimension: 150
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.01
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 10.0 in DAQUAR, 15.0 in COCO-QA 6.6K, 50.0 in COCO-QA Full
   Weight regularization: square loss, λ = 5e-5

Bibliography

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012., 2012, pp. 1106–1114.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," CoRR, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[4] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," CoRR, vol. abs/1411.2539, 2014. [Online]. Available: http://arxiv.org/abs/1411.2539

[5] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in Neural Information Processing Systems (NIPS'14), 2014. [Online]. Available: http://arxiv.org/abs/1410.0210

[6] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, 2014, pp. 740–755.

[7] K. M. Fauske, "Example: Neural network," 2006. [Online]. Available: http://www.texample.net/tikz/examples/neural-network/

[8] DeepLearning.net, "Lenet." [Online]. Available: http://deeplearning.net/tutorial/lenet.html

[9] M. Malinowski and M. Fritz, "Towards a visual turing challenge," CoRR, vol. abs/1410.8027, 2014. [Online]. Available: http://arxiv.org/abs/1410.8027

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accu-

rate object detection and semantic segmentation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, 2014, pp. 580–587.

[11] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," IEEE Transactions on Audio, Speech & Language Processing, vol. 20, no. 1, pp. 30–42, 2012.

[12] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013, 2013, pp. 8604–8608.

[13] A. Mnih and G. E. Hinton, "Three new graphical models for statistical language modelling," in Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007, 2007, pp. 641–648.

[14] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010, 2010, pp. 1045–1048.

[15] M. Hassoun, Fundamentals of Artificial Neural Networks. A Bradford Book, 2003.

[16] K.-L. Du and M. N. S. Swamy, Neural Networks and Statistical Learning. Springer, 2014.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," Complex Systems, 1995.

[20] G. K. Zipf, Human Behavior and the Principle of Least Effort. Addison-Wesley, 1949.

[21] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," Journal of Machine Learning Research, vol. 3, pp. 1137–1155, 2003. [Online]. Available: http://www.jmlr.org/papers/v3/bengio03a.html

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," CoRR, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[23] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "De-

vise: A deep visual-semantic embedding model," in Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013., 2013, pp. 2121–2129.

[24] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint word-image embeddings," Machine Learning, vol. 81, no. 1, pp. 21–35, 2010.

[25] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Baby talk: Understanding and generating simple image descriptions," in The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011, 2011, pp. 1601–1608.

[26] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, X. Han, A. Mensch, A. C. Berg, T. L. Berg, and H. D. III, "Midge: Generating image descriptions from computer vision detections," in EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012, 2012, pp. 747–756.

[27] B. L. Lewis, "In the game: The interface between watson and jeopardy!" IBM Journal of Research and Development, vol. 56, no. 3, p. 17, 2012. [Online]. Available: http://dx.doi.org/10.1147/JRD.2012.2188932

[28] J. Weston, S. Chopra, and A. Bordes, "Memory networks," CoRR, vol. abs/1410.3916, 2014. [Online]. Available: http://arxiv.org/abs/1410.3916

[29] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in ECCV, 2012.

[30] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," Computational Linguistics, vol. 39, no. 2, pp. 389–446, 2013.

[31] K. Papineni, S. Roukos, T. Ward, and W. jing Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in Meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[32] T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," CoRR, vol. abs/1410.8206, 2014. [Online]. Available: http://arxiv.org/abs/1410.8206

[33] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," CoRR, vol. abs/1411.4555, 2014. [Online]. Available: http://arxiv.org/abs/1411.4555

[34] Z. Wu and M. Palmer, "Verb semantics and lexical selection," in In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994, pp. 133–138.

[35] R. Borsley, Syntactic Theory: A Unified Approach, ser. Hodder Arnold Publication.

Arnold, 1999.

[36] N. Chomsky, Conditions on Transformations. New York: Academic Press, 1973.

[37] C. Fellbaum, Ed., WordNet An Electronic Lexical Database. Cambridge, MA ; London: The MIT Press, May 1998.

[38] S. Bird, "NLTK: the natural language toolkit," in ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006, 2006.

[39] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in In proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 2003, pp. 423–430.

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," arXiv:1409.0575, 2014.

[41] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," in Information Processing and Management, 1988, pp. 513–523.

[42] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 3320–3328. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks

[43] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," ArXiv e-prints, Feb. 2015.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[45] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in Proceedings of the 30th International Conference on Machine Learning, ICML 2013, 2013, pp. 1310–1318.

[46] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," CoRR, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329