COMPSCI682 Course Project
Visual Question Answering with Neural Networks over Graphs.

Author-1
College of Information and Computer Science.
University of Massachusetts, Amherst.

Abstract

Visual question answering (VQA) is an essential method in extracting useful information from image and video data. This type of semantic analysis of visual data can be very pivotal in fields like human-machine interaction and robotics. Most methods on this task handle the language modelling and image modelling tasks separately which loses out on a lot of structural information. This report discusses a different approach, first suggested by Teney et al. [6], in which we build graphs over the image objects and question words. These graphs are then processed with neural networks over graphs in order to capture the un-ordered nature of information in the images and the questions and then finally, identify correlations between the objects and the words most useful to the VQA task.

1. Introduction

The task of visual question answering (VQA) entails developing a model that can answer several questions based on a given input image. This task is particularly interesting to explore because it lies on the intersection of computer vision, natural language processing and artificial intelligence. This is a task which is performed rather easily by an intelligent agent (i.e. humans), and being able to replicate it will reveal greater insights into how an intelligent agent may possibly go about deciphering information from visual input.

Earlier attempts at this problem use a combination of pretrained CNN and/or LSTM networks, to derive features from the image and the question and train with these features on the visual question answering task. However, these methods have several drawbacks, as shown by [1] and [7]. Firstly, note that image features from a CNN contain information about the image as a whole. Most information about the spatial relations between the objects in the scene is not directly available for use. Similarly for the case of LSTMs, the syntactic structural information of the question isn't readily available for use.

Finally, most earlier approaches focus mainly on real-world images. In such cases, more often than not, the true answer is obvious from just learning the co-occurances of words in the questions and answers while training. Therefore in those cases, its hard to say for certain that the model is learning to perform at the visual question answering task, or if its just learning a better model of the questions and answers of the dataset itself.

The model that I will mainly be focusing on, as a part of this project was first suggested by Teney et al. in [6]. The main idea behind this method, is to extract the unordered nature of information between the words of a question and objects in an image. In a nutshell, this model performs the following key tasks:

- Build a semantic graph over the input question using pre-trained word embeddings as node(word) features, and using a pre-trained parser to encode the edges of the graph as sematic relationships between the nodes(words).
- Build a graph over the image, where each of the nodes represents an object in the image, and the edges represent various spatial relationships between the objects.
- In each graph, the embedding of each node is updated based on its context (its neighbours in the graph).
- Finally, a graph neural network is trained with the graphs of the question and the image and input, and the answer to the question as output. The network learns a soft matching between the nodes of the question graph and the nodes of the image graph, which is used to answer the question. This soft matching can also be referred to as the attention mechanism of the model.

2. Background

In this section, I have presented several works relevant to building the approach in the next section of this paper.

2.1. VQA dataset

The VQA dataset, or the Visual Question Answering Dataset was first developed by Antol et al. [1] and later expanded upon by Zhang et al. in [7]. We will be working with the abstract scenes and the balanced binary abstract scenes from version 2 of this dataset. These have been discussed below.

The abstract scenes dataset is a set of user-generated clipart images from an interface which allows the user to pick different objects (3 to 5) and create a cartoon scene. This interface stores all the information during creation as scene information which will later be used to create graphs over the scene objects. The information stored includes information about the object type, object pose and its spatial information. This scene information allows us to focus exclusively on the task of visual question-answering, independent of the visual system. This is discussed in greater detail in the approach section. This dataset includes $20k$ training images, $10k$ validation images and $20k$ test images.

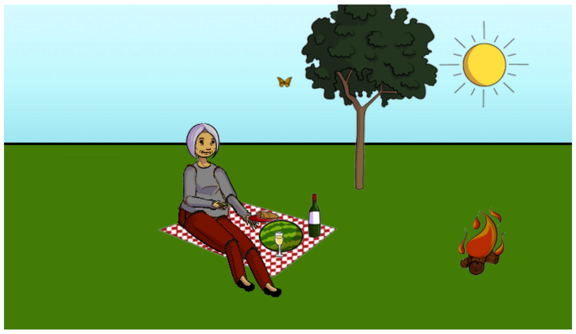

Figure 1. An example image from the abstract scenes dataset.

The questions in this dataset include, both, multiple choice questions and open-ended questions. However, we will mainly be focusing on multiple choice questions which have a set number of possible answer choices. This dataset includes $60k$ training questions, $30k$ validation questions and $60k$ test questions.

The balanced binary abstract scenes dataset is an extension of the VQA dataset that helps test robustness of the model to language priors. This dataset contains pairs of contrasting images which have opposite answers to the same yes/no type questions. High performance of the model on this dataset shows that the model is learning to correlate items from the scene and the question, rather than being biased by language model priors. This dataset includes $22k$ training questions and $11k$ validation questions; and $20k$ training images and $10k$ validation images.

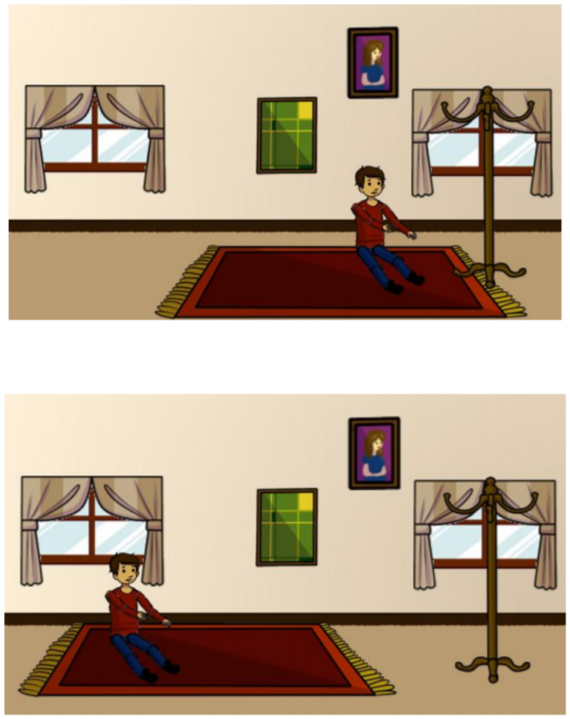

Figure 2. Balanced Binary Abstract Scenes Example. Pairs of contrasting images that have opposite answers to the question, ”Is the man close to the left window?”

2.2. GloVe Word Representation

GloVe stands for global vector, and it is an unsupervised learning algorithm to model words as vectors in $\mathbb{R}^N$, such that the word vectors in that space showcase interesting linear substructures that are consistent with the semantics of the words.

The GloVe vectors are derived over a given corpus of documents. The main idea here is that semantic relations between words can be derived from how often they co-occur in the same document, in a given corpus of data. The precise details of how word co-occurence probabilities can be used with matrix factorization methods to form vector representations can be found in [4].

2.3. Dependency Parser

A dependency parser takes as input, a sentence from a natural language, and outputs its grammatical structure in the form of a directed tree. The edges of the tree are labelled with the grammatical/syntactic relationship between the words/punctuations that the edge connects. The word at the head of the edge, modifies the word at the tail and the exact type of modification is given by the edge label (which is one of several possible dependencies).

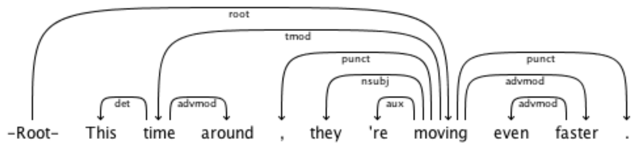

Figure 3. Example of a dependency parse.

The figure above shows the output of the stanford dependency parser on the given sentence. In this case, the arrow from the word 'time' to the word 'around' labelled 'advmod', means that 'around' modifies 'time', and the modification is adverbial in nature. For the exact functioning and construction of the parser, refer to [3].

2.4. Gated Recurrent Unit

The gated recurrent unit, or GRU is an RNN architecture which resolves the problem of vanishing gradients in vanilla RNNs. In the vanilla RNN architecture, the hidden state of the cell changes in very few time steps. Due to this, it is unable to perform accurate predictions which rely on retaining information from several time steps back.

The GRU overcomes this problem by learning two gates, which perform the function of selectively forgetting past information from the hidden state, and then selectively updating it with new information.

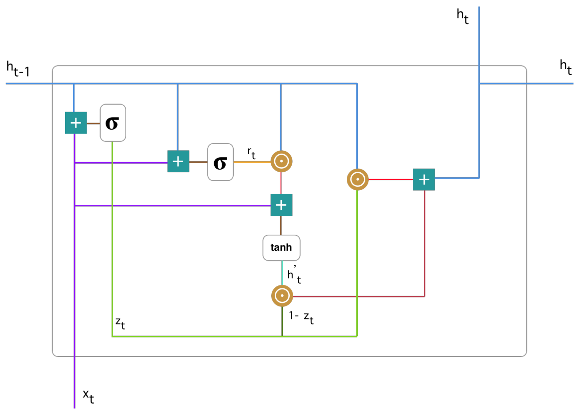

Figure 4. A single cell of GRU.

The first gate is called the update gate. It can be thought of as dictating to the model, how much of the information from the past needs to be transferred forward based on the current input and past hidden state. It has the following formula;

$$z_t = \sigma(W^z x_t + U^z h_{t-1}) \quad (1)$$

The second gate is called the reset gate, which helps the model selectively forget past information. It has the following formula;

$$r_t = \sigma(W^r x_t + U^r h_{t-1}) \quad (2)$$

Now that we have equations for both the gates, we will see how they are used to make the recurrent update. We will first use the reset gate to selectively forget past information (h_{t-1}) and add new information from current input.

$$h'_t = tanh(W x_t + r_t \circ U h_{t-1}) \quad (3)$$

Here $a \circ b$ represents element-wise product.

Finally, we use the update gate to control what fraction of the previous information/state is passed along. This gives us the final value of the current state.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t \quad (4)$$

For more details about the GRU, refer to [2].

3. Approach

This section describes all of the steps that goes into building the graph over the questions and the images, and finally shows how these graphs are processed to produce the answers.

3.1. Scene Graph

The graph over the input image should perform two key functions. Each node of the graph should encode the features of each object in the image, and the edges of the graph should encode the spatial relationships among those objects. As discussed in 2.1, the abstract images subset of the VQA datset already contains details about all the objects in the image, as well as their spatial information in the image. This information can be used to encode the abstract image as a graph.

3.1.1 Object Information as Nodes

For a given scene S with N^S-many objects, each object is encoded using the following information from the dataset:

- Each object is categorized as either a human, animal, small object or large object. This is encoded as a one-hot vector and appended to the object node vector.
- Beyond object category, there is also the information of object type such as the animal object category may contain further information whether the animal is a cat, dog etc. This information is also encoded as a one-hot vector and appended to the object vector.
- For animate/deformable objects the data also holds the information about the expression (encoded as one-hot again, out of all possible expressions) and the pose of the object. The pose is quantified as the position coordinates of the arms, legs and head, relative to the torso (therefore, a 10-dimensional vector).

Appending all this information together as a single vector, produces the object node for a given object in the scene which ends up being 159-dimensional. Lets call this object node vector $x_i^S \in \mathbb{R}^{159}$. But in order to be able to process all the nodes and edges of both, the scene and the question graph together, we need to transform this 159-dimensional vector to an H-dimensional vector (let H be 300 for now). Therefore;

$$x_i'^S = W_1 x_i^S + b_1 \tag{5}$$

3.1.2 Spatial Relation between Objects as Edges

The edges e_{ij}^S encode the spatial relationship between the objects i and j in the image S. It has the following components:

- The signed difference between their x and y coordinates.
- The inverse of the absolute difference between their x and y coordinates.
- Relative depth from the image plane, encoded as $+1$, if object i is closer to the image plane and -1 if object j is closer.

Finally, e_{ij}^S also has to undergo an affine transformation and projected to an H-dimensional space. Therefore;

$$e_{ij}'^S = W_2 e_{ij}^S + b_2 \tag{6}$$

3.2. Question Graph

When building graphs over questions, we need to keep in mind our final goal of visual question answering. Therefore the graphs we build over the questions need to capture both, the semantics of the individual words (as graph nodes) in the question, as well as the syntactic relationships between the words (as graph edges).

3.2.1 Word Embeddings as Nodes

First lets focus on the task of encoding the word meaning as nodes. Let a given question have N^Q-many words (including punctuation). Now, for each word in the question, let x_i^Q denote the index of the word in the vocabulary V, corresponding to the corpus over which the GloVe word embeddings are learnt.

Therefore for each question Q;

$$x_i^Q \in \{1, 2, \dots, V\}, \forall i \in \{1, 2, \dots, N^Q\} \tag{7}$$

Now, we use a pre-trained matrix W_3, as an embedding matrix, to represent each word as a vector in a semantically consistent space (say, of dimension H).Therefore W_3 has a dimension of $V \times H$, where row $W_3[i]$ corresponds to the H-dimensional semantic embedding of the i-th word of the vocabulary. Here we use the 300-dimensional GloVe vector representation (therefore $H = 300$) trained on six billion words from the Wikipedia and Gigaword corpus. However, we are free to use pretrained weights over other corpuses and also use lower dimensional vector spaces ($H =$ 200,100 or 50), as long as the dimensionality of the vector space remains consistent across the embeddings of the node and edge features of both, the scene graph and the question graph. Therefore we have:

$$x_i'^Q = W_3[x_i^Q] \text{ where } x_i'^Q \in \mathbb{R}^{\mathbb{H}}, H = 300 \tag{8}$$

3.2.2 Syntactic Dependencies as Edges

Now lets focus on encoding the relations between the words that occur in the question, as edges between the word nodes. For this, we make use of a pre-trained dependency parser (in our case, we use the dependency parser from the spaCy module), which marks the syntactic relationships between the words of the question as directed edges.

The edge between the words i and j of question Q is encoded as integer e_{ij}^Q, which is nothing but the index of the dependency output by the parser, among all possible dependencies the parser can output. Therefore if the parser can output P-many dependencies, for each pair (i, j) of Q for which the parser outputs a dependency;

$$e_{ij}^Q \in \{1, 2, \dots, P\} \tag{9}$$

Now, just as we embedded the word vectors in an H-dimensional ($H = 300$) vector space, we will also embed the edge labels in an H-dimensional space, using a $P \times H$ dimensional embedding matrix W_4. But, unlike the word embedding which were pretrained GloVe vectors, W_4 is learnt during training. Therefore we have,

$$e_{ij}'^Q = W_4[e_{ij}^Q] \tag{10}$$

Now, all of $e_{ij}'^Q$ and $x_i'^Q$ are vectors in $\mathbb{R}^H$.

3.3. GRU over Graph Nodes

Now that we have e'_{ij} and x'_i for both, questions and scenes in the same H-dimensional space, we can start extracting information from them using a GRU architecture over each of the graph nodes. The key idea here is to update the representation of each of the nodes, based on the context from each of its neighbours. To do this, we associate a GRU with each node of both, the question and the scene graphs and process them for 3 to 5 ($T^Q = T^S = 5$) iterations.

In the equations below, $a \circ b$ denotes the element-wise product, and $[a : b]$ represents concatenation.

$$h_i^0 = 0 \tag{11}$$

$$n_i = avg_j(e'_{ij} \circ x'_j) \tag{12}$$

$$h_i^t = GRU(h_i^{t-1}, [x'_i : n_i]), \text{for } t \in \{1, \ldots, T\} \tag{13}$$

Here h_i^t is also H-dimensional. The final value, h_i^T is treated as the new updated state encoding. Therefore;

$$x''_i = h_i^T, \text{ for both questions and scenes} \tag{14}$$

3.4. Attention mechanism

The idea behind an attention mechanism is for the model to learn which parts of the input question relate most to, or focus most on which parts of the input scene. Therefore for each pair of nodes, one from the question graph and one from the scene graph, we need to associate a weight (scalar) which quantifies how much that specific word focuses on that specific object in the scene. Therefore we have,

$$a_{ij} = \sigma(W_5(\frac{x_i'^Q}{\|x_i'^Q\|} \circ \frac{x_j'^S}{\|x_j'^S\|}) + b_5) \tag{15}$$

Note that since a_{ij} need to be scalars, W_5 has shape $1 \times H$ and b_5 is a scalar. Also note that we have used the context free node vectors (and not $x_i''^S$ or $x_i'^Q$) to find the attention weights.

We now use these attention weights to scale every combination of question node and image node vectors, post GRU processing. Therefore;

$$y_{ij} = a_{ij}[x_i''^Q : x_j''^S] \tag{16}$$

Now note that, the number of nodes in Q and S can be variable. And so, we need to perform pooling operations over question nodes as well as scene nodes (that is, sum over i and j) in order to overcome this variability. However, we will interleave the pooling operations with dense linear layers. Therefore we get the following;

$$y'_i = ReLU(W_6 \Sigma_{j=1}^{N^S} y_{ij} + b_6) \tag{17}$$

$$\hat{y} = \sigma(W_7 \Sigma_{i=1}^{N^Q} y'_i + b_7) \tag{18}$$

Here W_6 is $2H \times 2H$ dimensional, and W_7 is $C \times 2H$ where C is the number of answer choices which depends on the type of question answering task (multiple choice answering, or yes/no answering).

4. Experiments

I was unable to complete the implementation of this model in due time for the submission. Here I've listed the major hurdles I faced while attempting an implementation. This is followed by showing the main results from Teney et al. [6] just for the sake of completeness.

4.1. Problems Faced

- The VQAv2 dataset has the scene information for every cartoon image in the form of a .json file which contains all the information produced by the interface that was used to create the images in the dataset. This data is quite complicated to navigate and it was particularly hard and time-consuming to extract the information needed to form the 159-dimensional object representation and the edge features. The API available for handling this data also didn't have any functions that could be particularly useful towards this task. This problem of extracting information from the .json file was also persistent in the questions data, and it took a lot of time to be able to extract the correct set of questions and multiple choice answers for a given image.

- As far as handling the questions is concerned, using the pre-trained GloVe embeddings to form word node representations was completed. The 300-dimensional embedding trained on the Wikipedia and Gigaword corpus was used. However, I faced a lot of setbacks in trying to build an adjacency matrix over the words of the question. This mainly happened in trying to map all possible dependencies of the parser to integers. I couldn't find an efficient way to do this in spaCy or the stanford dependency parser.

Given the above two major problems in handling the data, and the fact that I was working alone; I didn't have enough time to train the GRU by the time I was done with building the graphs over the questions and the scenes.

4.2. Original Results

- In the case of the abstract scene dataset, where the model had to choose an answer from a given set of choices, the implementation in [6] of the graph VQA model achieved an overall accuracy of 74.37%.

- In case of the balanced binary scenes dataset, the authors in [6] tested several variations of their model in order to see the impact of various design choices on the performance of the model on the task. During testing for this task, the network is shown the pair of complimentary images and it has to answer yes or no for each image in the pair. Their results are shown below;

(1)	Question: no parsing (graph with previous/next edges)	37.9
(2)	Question: word embedding not pretrained	33.8
(3)	Scene: no edge features (e'^{S}_{ij}=1)	36.8
(4)	Graph processing: disabled for question (x''^{Q}_i=x'^{S}_i)	37.1
(5)	Graph processing: disabled for scene (x''^{S}_i=x'^{Q}_i)	37.0
(6)	Graph processing: disabled for question/scene	35.7
(7)	Graph processing: only 1 iteration for question (T^{Q}=1)	39.0
(8)	Graph processing: only 1 iteration for scene (T^{S}=1)	37.9
(9)	Graph processing: only 1 iteration for question/scene	39.1
(10)	Uniform matching weights (a_{ij}=1)	24.4

Figure 5. Performance of the model [6] on the balanced dataset.

4.3. Visualizing Results

Here, I've shown the visualization of the attention matrix $a[ij]$ for a given scene and question from the abstract scene dataset, on the multiple choice task.

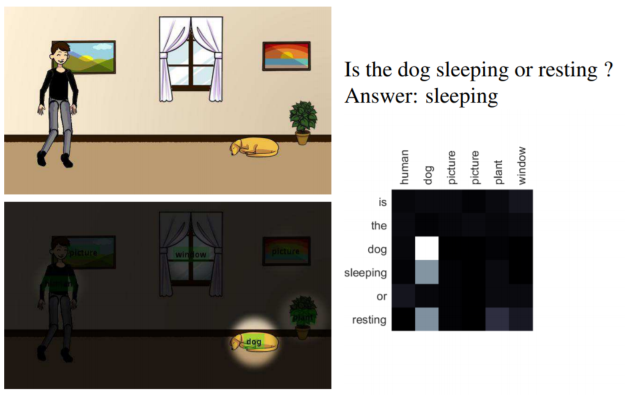

Figure 6. Visualization of the learnt attention matrix for a particular scene and question. The image on the bottom shows the parts of the scene that the model pays attention to, given the question.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] M.-C. De Marneffe and C. D. Manning. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[4] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] D. Teney, P. Anderson, X. He, and A. van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *CoRR*, abs/1708.02711, 2017.

[6] D. Teney, L. Liu, and A. van den Hengel. Graph-structured representations for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2017.

[7] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and yang: Balancing and answering binary visual questions. *CoRR*, abs/1511.05099, 2015.