# Graph representation of scenes and questions

Chaonan Song

May 15, 2018

## 1 Related work

The input data for each training or test instance is a problem and is a parametric description of the scene content. The question is processed with the Stanford dependency parser [3], which outputs the following.

- A set of $N^Q$ words that constitute the nodes of the question graph. Each word is represented by its index in the input vocabulary, a token$x_i^Q \in \mathbb{Z}(i \in 1 \dots N^Q)$.
- A set of pairwise relations between words, which constitute the edges of our graph. An edge between words i and j is represented by $e_{ij}^Q \in \mathbb{Z}$, an index among the possible types of dependencies.

The dataset provides the following information about the image

- A set of NS objects that constitute the nodes of the scene graph. Each node is represented by a vector $x_i^S \in \mathbb{R}^C$ of visual features ($i \in 1 \dots N^S$). Please refer to the supplementary material for implementation details.
- A set of pairwise relations between all objects. They form the edges of a fully-connected graph of the scene. The edge between objects i and j is represented by a vector $e_{ij}^S \in \mathbb{R}^D$that encodes relative spatial relationships (see supp. mat.).

Their experiments were performed on the clip art scene data set, where scene descriptions were provided as a list of objects. This method also applies to real images where the object list is replaced by the candidate object.

The features of all nodes and edges are projected to a vector space RH of common dimension (typically H=300). The question nodes and edges use vector embeddings implemented as look-up tables, and the scene nodes and edges use affine projections:

$$x_i^{'Q} = W_1[x_i^Q] \qquad e_{ij}^{'Q} = W_2[e_{ij}^Q] \tag{1}$$

$$x_i^{'s} = W_3 x_i^{'s} + b_3 \qquad e_{ij}^{'s} = W_4 e_i^{'s} + b_4 \tag{2}$$

with W1 the word embedding (usually pretrained, see supplementary material), $W_2$ the embedding of dependencies, $W_3 \in \mathbb{R}^h \times c$ and $W_4 \in \mathbb{R}^{hd}$ weight matrices, and $b_3 \in \mathbb{R}^c$ and $b4 \in \mathbb{R}^d$ biases.

## 2 Processing graphs with neural networks

They describe a deep neural network that is suitable for processing problems and scene graphs to infer the answer. See Fig. 1 The two graphics that represent the problem and the scene are handled independently. They have discarded the indices S and Q of this paragraph, because both figures apply the same procedure. Each node $x_i$ is associated with a gated recurrent unit (GRU [6]) and processed over a fixed number T of iterations (typically T=4):

$$h_i^0 = 0 \tag{3}$$

$$n_i = pool_j(e_{ij}^{'} o x_j^{'}) \tag{4}$$

$$h_i^t = GRU(h_i^{t-1}, [x_i^{'}; n_i]) \qquad t \in [1, T]. \tag{5}$$

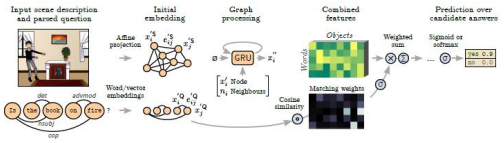


Figure 1: Architecture of the proposed neural network. The input is provided as a description of the scene (a list of objects with their visual characteristics) and a parsed question (words with their syntactic relations). The scene-graph contains a node with a feature vector for each object, and edge features that represent their spatial relationships. The question-graph reflects the parse tree of the question, with a word embedding for each node, and a vector embedding of types of syntactic dependencies for edges. A recurrent unit (GRU) is associated with each node of both graphs. Over multiple iterations, the GRU updates a representation of each node that integrates context from its neighbours within the graph. Features of all objects and all words are combined (concatenated) pairwise, and they are weighted with a form of attention. That effectively matches elements between the question and the scene. The weighted sum of features is passed through a final classifier that predicts scores over a fixed set of candidate answers.

The final state of the GRU is used as the new representation of the nodes: $x_i = h_i^T$. Pool operations convert features from a variable number of nodes into a fixed-size representation. Any commutative operation can be used (*e.g.* sum, maximum). In their implementation, they discovered the best performance through the average function and considered the average of the number of variable connections. Their formula is similar to the gated graph network [1] but slightly different because the information dissemination in their model is limited to the first order. Note that their graphs are typically densely connected. In fact, they estimate the relevance of pairwise combinations of each possible word and object. More precisely, we compute scalar matching weights between node sets $x_i^{'Q}$ and $x_i^{'s}$ (*e.g.* [2]) Therefore, $\forall i \in 1 \dots N^Q, j \in 1 \dots N^s$ :

$$[H]a_{ij} = \sigma(W_5(\frac{x_i^{'Q}}{\|x_i^{'Q}\|} \circ \frac{x_j^{'S}}{\|x_j^{'S}\|}) + b_5) \qquad (6)$$

where $W_5 \in \mathbb{R}^{1h} and b_5 \in \mathbb{R}$are learned weights and biases,and $\sigma$ the logistic function that introduces a non-linearity and bounds the weights to (0, 1). They apply the scalar weights $a_{ij}$ to the corresponding pairing combinations of problem and scene features, thereby focusing and placing greater emphasis on matching pairs (Eq. 7 ). We sum the weighted features over the scene elements (Eq. 8) then over the question elements (Eq. 9), interleaving the sums with affine projections and non-linearities to obtain a final prediction:

$$y_{ij} = a_{ij} \cdot [x_i^{''Q}; x_j^{''S}] \qquad (7)$$

$$y_i^{'} = f(W_6 \sum_{j}^{N^S} y_{ij} + b_6) \qquad (8)$$

$$y^{''} = f'(w_7 \sum_{i}^{N^Q} y_i^{'} + b_7) \qquad (9)$$

# References


[1] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *Computer Science*, 2015.

[2] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. 2016.

[3] Marie Catherine De Marneffe and Christopher 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, 2008.