SPURs
Sanchit Jain 160050043
Pritam 170070042
Utkarsh Gupta 160050032
Rupesh 160050042

ZERO SHOT LEARNING in SCENE GRAPH GENERATION

4th May 2019

INTRODUCTION

Scene graphs is a succinct representation of important information in an image. For any given image, the scene graph is a graph whose nodes represent objects (in the image) and the edges represent the relationships between the objects corresponding to the two nodes of the edge. For example, Man — sits — Horse, where 'Man' and 'Horse' are objects and 'sits' is a relationship.

There's been a significant progress made towards solving this problem in recent literature. Many methods and innovative ideas have been proposed to tackle it, but on a higher level there seems to be a common flow underneath all of these which is,

Step 1 - Detect Objects

Step 2 - Classify relationship between every pair of objects

In this project we attempt to further the state-of-the-art methods to identify and classify unseen relationships using zero-shot learning techniques. Our key idea is to work with word-embeddings instead of limiting to the space of train-data predicates.

RELATED WORK

In this project, we are majorly guided by the work 'Visual Translation Embedding Network for Visual Relation Detection' by Zhang et al. [1]. Key learning from this work is working in embedding space:

a. Learn the transformation matrices from feature space to embedding space

b. Maximize dot product between target relationship and the predicted relationship

1. Visual Translation Embedding Network for Visual Relation Detection

   Github: https://github.com/zawlin/cvpr17_vtranse

   https://github.com/yangxuntu/vrd

2. Scene Graph Generation by Iterative Message Passing

   Github: https://github.com/danfeiX/scene-graph-TF-release

Additional References

Zero-Shot Learning - The Good, the Bad and the Ugly

Semantic Autoencoder for Zero-Shot Learning

Brief summary of various works

1) Scene Graph generation by iterative message passing (https://arxiv.org/pdf/1702.08319.pdf)
   - End-to-End model and uses global context for prediction of scene graph relations and object labels. Global context is enforced via message passing among nodes.
   - Input is an image and a set of bounding boxes proposed by an RPN(region proposal network [the one used in Faster R-CNN paper] (https://arxiv.org/abs/1506.01497) and then for each object proposal we infer
     1) the object class label
     2) 4 bounding box offsets relative to the proposed box which refines the given box
     3) a relationship variable for each pair of the objects.
   - Let the set of object classes be C and the set of relationship classes be R (in practise they pick the 150 most used object classes in Visual Genome dataset and 50 most used relations in same dataset). The set of variables is defined as **x** = (x_{i}^{cls},x_{i}^{bbox},x_{i->j} | i \in [1..n] and i!= j and j \in [1..n]). First term is the class label for the object the second term is the bounding offsets and the last term is the relationship predicate between the ith and jth proposed objects.
   - We want to find **x** such that Pr(**x**|I,B_{I}) is maximized where where I,B_{I} is the image and the set of bounding boxes proposed by RPN. Now the rest is how to approximate it.
   - Based on the proposed relationships we have obtain a “scene graph” now at each object node place a gated recurrent unit and similarly at each edge. GRUs at object nodes share same parameters and GRUs at relationship

nodes share same parameters. Now using these iterative message passing occurs from neighbours(GRU takes in composed message, and its hidden state as input and updates its hidden state accordingly in one iteration) the final hidden states are used for prediction of object categories, bounding box offsets and relationship types.

- In the very first iteration GRUs take as input the visual features of the proposal box that were extracted by a ROI pooling layer from the image as input (for object nodes) for relationship nodes we use the visual features of the union box over the proposal boxes (which are corresponding subject and objects). In later iterations objects are the composed messages.
- Composed messages are weighted sum of individual messages(which are hidden layer) and weights are learned. (exact function is a bit different but is same in spirit)
- The graph can be seen as a bipartite graphs where one partite set is the set of object nodes and other set is the set of relationship nodes(don't view relationship as edges but as nodes) and hence iteration can be viewed in a simpler manner as message flowing from one partite set to another and vice versa.

-

1) Scene graph generation from Objects,Phrases and region captions
    - Phrase here is the relationship predicate connecting two objects.
    - Given an image a graph is built to align the object,phrase, and the caption regions within the image. The a feature refining structure is used to pass message between the layers(Caption layer, phrase layer and the object layer) this thus takes into account information from other semantic levels before making final prediction. The network that they build they call as Multi-level Scene description network. (Based on the convolutional layer of VGG-16(https://arxiv.org/pdf/1409.1556.pdf)
    - First step is to generate ROIs for objects phrases and the region captions. For object RPN from faster RCNN paper used, for phrase all pairs of the objects are grouped, and caption regino proposals are directly generated by another RPN which is trained with ground truth region bounding boxes.
    - After this they do feature specialization , which is just they first feed the ROI to ROI pooling and then use different FC layers to get features.
    - Next step is the dynamic graph construction. View the graph as 3 tripartite graph where the respective sets are of object nodes, phrase nodes and the caption nodes. Connection between phrase and object pairs are built during the first step only, Edges between caption and object nodes are not considered, an edge between caption and phrase node is added only if if the caption proposal covers enough fraction of the phrase proposal.
    - Now comes feature refining, Like the iterative message passing approach message (or features are passed) and there are weights to be learnt and the features are updated, and them final prediction is made using those features.(I didnt find the exact details conceptually worth mentioning)
    - The phrases and the object class are predicted using the refined features and scene graph is constructed from that, using the refined features of the caption layer, region caption generation is also carried out.

2) Visual relationship detection with Deep structural Ranking
(https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16491/16300)
    - Proposed method separately models objects and predicate and fuses their results to predicate the relationship, Since the predicates may generalize across different types of object pair, they “fuse” visual appearance features, spatial location features, and semantic embedding features to get a joint feature.
    - For an input image first extract the above three features for each relationship instance triplet (subject,relationship,predicate) are obtained (described below). Then using that joint feature a compatibility of the relationship predicate wrt image is defined which is just a dot product with the joint feature and the other vector is learned.
    - For visual feature extraction same setup as the last paper used, ROI pooling followed by FC (same CNN used) on bounding boxes and union of bounding boxes (for the relationship).

- For spatial location, they use the 2 bounding boxes and define a 4 dimensional feature vector which uses the difference of the coordinates ratios of heights and lengths to make the feature. They also do one more thing, they take the spatial masks of the subject and the object(subject-predicate-object triplet) and downsample it, concatenate it and the feed it into a CNN which converts ir into a lower dimensional vector, they try both these approaches.
- Semantic features integrate the category of the subject and the object, the proposed semantic embedding layer maps the object category into a feature embedding vector, concatenate the embedding of the subject category. Now this is passed through a FC and then fused with visual and spatial features. (fusing is concatenate the 3 feature vectors and pass them through FC)
- The salient feature is that they propose to minimize a structural ranking criteria that addresses the incompleteness of annotations.

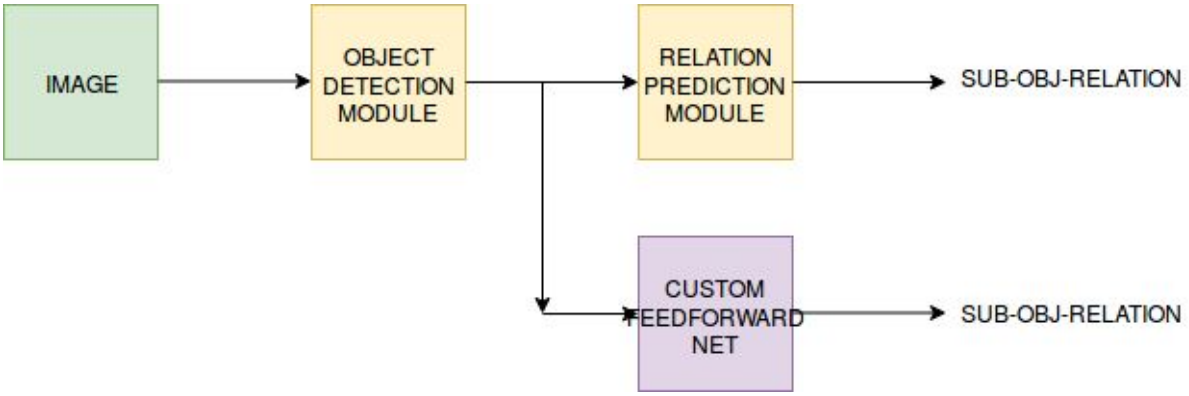
3) Deep Variation-structured Reinforcement Learning for Visual Relationship and attribute detection (https://arxiv.org/pdf/1703.03054.pdf)

- Sequentially discovers object relationships and attributes in the whole image. First a directed semantic action graph is built using language priors.
- In the graph nodes are nouns, attributes, and predicates connected by edges that represent semantic connection. Directed edges consist of attribute phrases or predicate phrases.
- Directed action graph is built from annotated data set i.e, Visual Genome dataset by selecting objects that appear more than some threshold number of times and using the attributes and relationship predicates associated with them.
- Current deep RL models require several costly episodes of trial and error to converge even with small action space and our action space is quite large, to circumvent this they while traversing over the graph construct small adaptive action sets A,P,C for each step based on current and historical actions. A contains candidate attributes to describe an object, B has candidate predicate for relating a pair of objects and C contains new objects instances to mine in the next step. (hence we don't learn in the entire action space)
- Same CNN (Faster RCNN and VGG-16) used for object detection, RL starts with subject having most confident classification score and predicts all possible relationships and attributes wrt to current object and then moves to next object.
- Given a subject instance s and object instance s', A = set of attributes that have an edge with s and haven't been mined before, P = set of predicates such that (s,p,s') edge is present in action graph, C is more involved as described below.
- Based on bounding boxes of the subjects and other object proposals few neighbours are shortlisted and the set of possible attributes for them trimmed by removing attribute that have confidence score atmost 0.1 less than the most confident category, further remove from this set object categories the

objects that have been mined before. If this set is empty we proceed in a BFS manner

- In each step algo selects actions from the action sets A,P,C.
- State Space is made up of feature vectors of s,s',whole image,history phrase embedding vector.
- Reward function encourages faster exploration by rewarding higher than usual if next predicted object category overlaps with a ground truth object and is new, other rewards are +ve if the attribute or the relationship are predicted correctly otherwise they are -ve.
- Deep Q-network framework (https://arxiv.org/pdf/1312.5602.pdf)used to estimate three Q-valued sets, parametrized by network weights w_a,w_p,w_c corresponding to the three action sets. And we use Q-values to probabilistically choose our actions from the action sets). I will read this paper to properly undertand this, I am not sure how the update rules work in depth, superficially they look ok to me).

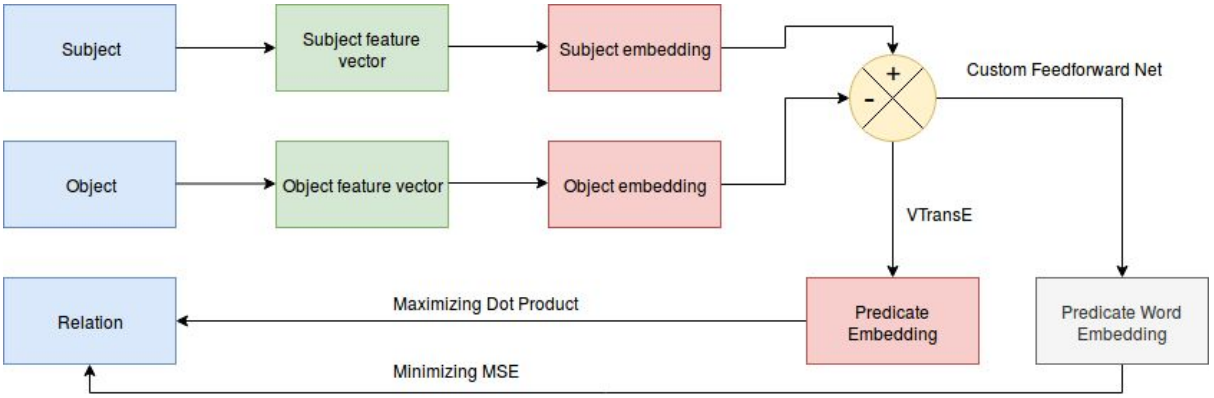
OUR APPROACH



The dataset used (is VRD dataset) has images corresponding to 100 object categories and 70 predicate categories. The VTransE [1] model can predict a relationship which is only among one of these 70 predicate categories. We planned to venture a step further by building a model that can predict unseen relations, i.e. predicates that are not present in training dataset.

How we went about realizing the above goal -
Firstly, we observed carefully what exactly VtransE model does and what part of it can we incorporate in our model. It is observed that this model learns a transform from object feature space to an embedding space, where it tries to minimize the dot-product between the difference of subject vector and object vector and the embedding of predicate.
Through our custom feed-forward network we learn a transformation from this embedding space to the word-embeddings, and then search for the nearest word in

the vocabulary thus allowing a possibility for identifying predicates outside the training data.



Training on two types of accuracies -

a) Accuracy 1: It checks whether the predicate that we predict matches with the original predicate
b) Accuracy 2: It checks whether true predicate is among the top 5 of the words predicted

Approaches tried

1. First approach is to simply enlarge the vocabulary with the new words before training (so the word does not appear anywhere in the results if the datasets but it is present in the vocabulary so that the word can be predicted). This approach was counter productive, as the loss function that the classifier used also penalizes wrong prediction so the model learns to not predict the extra words which we don't want.
2. Second approach was to modify the loss function of the model so that wrong predictions are penalized according to some similarity from some model like word2vec, but while doing so the baseline results did not came out to be good so we abandoned this approach
3. Our third approach that we use is avoids retraining the given implementation unlike the above approaches and hence is quite fast. In this approach we rely on the features extracted by the Region Proposal network used in the implementation for the object and the subject but we predict the predicate by passing those features through a feed forward network and matching th output with the embeddings of the predicates that we have. The nearest match is picked out as the answer. We try two embeddings one is obtained through the word2vec embeddings of the googlenews dataset and on is through Glove : GLobal Vectors for word representation.

EXPERIMENTS

1. Dataset

a. Visual Relation Detection
b. Visual Genome

VRD dataset description -

It has 4000 train instances and 1000 test instances, images belonging to 100 object categories and 70 predicate categories. Format is as below -

Image_name.jpg [...

{

1. Predicate : CATEGORY_ID
2. Object :
    a. Category : CATEGORY_ID
    b. Bbox : [YMIN, YMAX, XMIN, XMAX]
3. Subject :
    a. Category : CATEGORY_ID
    b. Bbox : [YMIN, YMAX, XMIN, XMAX]

}

…]

2. Code Description

Code references:

a. https://github.com/yangxuntu/vrd <Their Model> ,, Initial Code
b. https://github.com/rs9899/AML/ <Our additional network and changes>

Language: PYTHON

Environment: CUDA and python

Lines of code: ~300 in our implementation , ~100 lines modified in theirs

3. Platform

The customized Feedforward NN is built in Keras (with Tensorflow backend).
The code was tested and run on a GPU enabled laptop.

For VRD-50 dataset -

a. Training time of VtransE model: ~ 6 hrs
b. Testing time of VtransE model: ~ 1 hrs
c. Training time of custom Feedforward network: 15 min
d. Testing time of custom Feedforward network : 2 min

Multiple runs were performed to get a good feedforward network trying various architecture.

4. Results and Comparison

Accuracy 1

1. Using Custom Feedforward Net <Our side module>
    a. Subject||Object, Google News word-embedding : 0.41496

b. Subject - Object, Google News word-embedding : 0.407
c. Subject - Object, GLOVE word-embedding : 0.4401

2. Using VtransE Relation Prediction Module <Their code>: 0.4597

Accuracy 2 (top 5 metric)

3. Using Custom Feedforward Net
    a. Subject||Object, GLOVE word-embedding : 0.5644

EFFORTS

First 3 days at time, each group member picked separate fields on Machine learning and started searching for latest work and their possible extension that could be a potential project.

Next 2-3 days were spent were each team member to find the related work on the same as well as possible GITHUB implementation. Also, we discussed approaches among each other and decided which was easier to do. Our main aim was to pick one that has each of its submodule used in solving the Scene Graph problem known so that we don't have any implementation in our project that we can't explain. Also the available code we tried to find in Python so that we can easily reuse the code and add Zero-shot module.

Later it took 2 days to make the given implementation and required installation to make it running in the GPU of our laptop. Also in the meantime we planned how to add the zero-shot part and finalised on changing the loss function.

Finally last 3 days were all changing the code, making new tries and possibilities and also deciding what all can be done in the limited time and resources we have and finally completing the project.

The most challenging part was that after first run of their implementation it took around 6 hours and our initial plan was to change the loss function in their code but hypertuning after that with such a big data set was almost infeasible with our computational resource. So we overcame this by treating their code, once trained as a blackbox and output the embedding of subject and object to predict the predicate by our side model.

We contributed equally to the project taking turn and splitting the load to get it completed and presentable in time.