Analysis of Learning Cooperative Visual Dialog Agents: Project Milestone

Juanita Ordonez
ordonez2@stanford.edu
Custom Project
Mentor: Christopher Manning

Abstract

We implement the goal-driven cooperative multi-agent for the task of Visual Dialog. Here, agents play a game of "Guess What?", where one agent has the role of answerer (A-Bot) and the other has the task of questioner (Q-Bot). In this environment Q-Bot is trying to guess the image that A-Bot is looking at by asking A-Bot a series of questions about the image. Q-Bot is responsible asking good questions and guessing which image A-Bot is looking at. Finally, the model is fine-tuned using reiforcement learning where A-Bot and Q-Bot each learn a policy guided by a reward that measures whether the current guess of Q-Bot is closer to the correct image at every round.

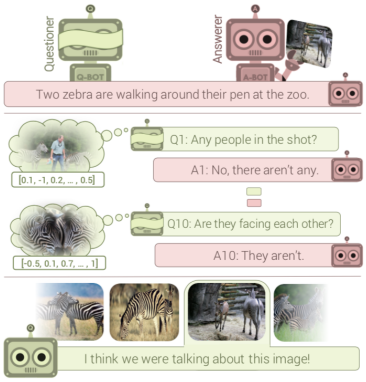
We evaluated the supervised pre-reinforcement-learning agents, trained on a subset of 20k examples from the VisDial dataset which were randomly selected. We achieve an MMR of 38.11, Recall @ 5 42.34 and Recall @ 10 50.52. We found that Q-Bot is significantly harder to train because it struggles to differentiate facts from each other. We also found that adding a scaling factor to the Q-Bot decoder loss helps with the repetitive nature of the questions but makes the model overfit more easily.

1 Introduction

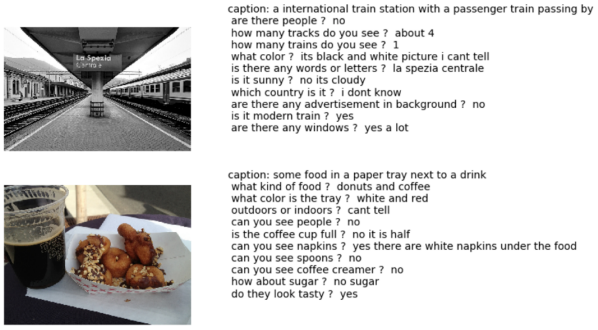
As technology is becoming more and more integrated into our society, human AI interaction is becoming more commonplace. To have an effective communication with an AI, other sources of data such as vision and speech will have to be integrated. This type of interaction will lead to a Human-AI cooperation which can aid us in many areas; for example, helping visually impaired users understand their surroundings or social media content[6], enabling analysts to sift through large quantities of surveillance data[4] and enabling users to interact naturally with intelligent assistants.

Having two agents converse about an image, also known as visual dialog, is a challenging task which involves to modalities of data: image and text. In this task, we require an agent to answer multi-round questions about an image. Specifically, given an image, a dialog history and a question about the image, the agent has to relate the question to the image, infer context from history and answers the question accurately [2]. Similar to Visual Question Answering (VQA), this task comes with the challenges because it requires vast set of AI capabilities to answer each question. Some examples of the capabilities needed are object detection, activity recognition, knowledge base reasoning and commonsense reasoning [6]. Unlike VQA, Visual Dialog has the additional challenge of keeping track of what has been said before in the conversation. Keeping track of the history significantly complicates the task. For example, consider that in the VisDial dataset which we use for this work, 98% of the dialogs and 38% of the questions contain a pronoun [1]. This means that the agents must keep track of the pronouns used and to which objects they refer throughout the conversation in order to both ask and answer questions in any meaningful way.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.



(a) A-Bot and Q-Bot game setup [1]



(b) VisDial Examples

Previous work has unnaturally treated dialog as a supervised learning problem where the answers are not generated but chosen from a list of possible candidates [9, 1]. These works treat dialog as a static supervised learning problem, instead of interactive learning problem that it naturally is. In this work we aim to implement and reproduce the result from [1], where they develop the first goal-driven multi-agent conversational model. The game environment is inspired from the "Guess What" game where two agents, answer bot (A-Bot) and question bot (Q-Bot) need to work together and communicate in natural language to win the game. First, these agents were pretrained on dialog data in supervised learning manner. Then, they were fine-tuned using reinforcement learning technique, REINFORCE [5]. We run a comprehensive analysis on the model performance and took a closer look what potential improvements opportunities can be apply to this setup.

2 Related Work

Previous work based on the ReferIt [7] game where one agent asks a question about the image and the other agent respond by "yes", "no" or" N/A". Based on these responses, the first agent then makes a guess of what object was selected. This game was then transformed to a "Guess What" game where the setup is similar. Here, there are two agents, A-Bot which sees an image and answers questions about it and Q-bot whose goal is to ask questions whose answers allow it to guess the image seen by A-Bot. This latter setup is simpler than the first in the sense that no bounding box labeling is necessary. However, because the answers from A-Bot can now be open-ended, i.e. generated, the output space of this model is significantly larger. This setup forces the agents to first learn a medium of communication (language) and then learn how to use this language effectively in order to complete the task.

Another game which bears some resemblence to ours is the Lewis Signaling Game presented in [8]. Here, there is a sender and a receiver. The sender has to send information over to the receiver and the receiver must encode it into a state. This is similar to our "Guess What" formulation where the sender can be seen as A-Bot and the receiver is Q-Bot. However, in our case, the receiver is not passive for Q-Bot prompts more information from A-Bot after encoding the past knowledge into a state.

3 Approach

The problem is set up as a conversation with a sequence of questions and answers centered around an image. Each conversation is in turn described as a series of rounds. A single image can have up to 10 question and answer rounds. This task is similar to Visual Question and Answering, but with follow-up questions (rounds) about the same image.

The model consist of two agents, answer agent (A-bot) and question agent (Q-bot). Where both agents are given a description of the image but only A-bot has seen the image while Q-bot needs to guess the image. At each round Q-bot asks a question about the image, then A-bot answers the question, by the end each round Q-bot guesses the image A-bot is looking at. This approach first pretrained both A-bot and Q-bot on dialog data to generate answers and question respectively. Then,

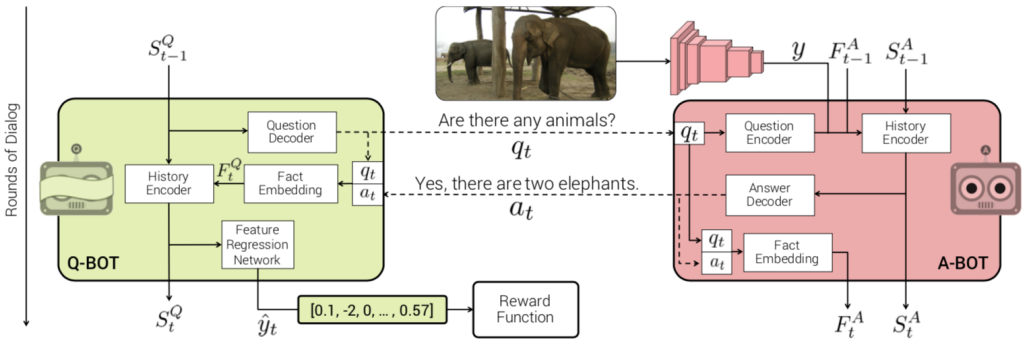


Figure 1: A-Bot and Q-Bot arhitecture [1]

using reinforcement learning, both agents are fine-tuned. The entire model and preprocessing was implemented from scratch. Using the Pytorch and NLTK [3] frameworks. For this work, we ran diagotics experiments to pin point problems and challenges with training this model.

3.1 Model Architecture

Q-bot consist of four components. The first is what they call a “Fact Encoder" which keeps track of the questions asked and the answers received. This boils down to a 2 layer LSTM where the input is the concatenation of question and answer pairs (q_t, a_t). The hidden state of this LSTM is the encoded fact of the particular timestep in this round, F_t^Q. After the fact is encoded in Q-bot, it is encoded into the State/History encoder resulting S_t^Q. This results in a two-level hierarchical encoding for the dialog. After the question and answer is encoded into the Q-bot then using the previous state S_{t-1}^Q the next question is generated by sequentially sampling words. Q-bot needs to take a guess of what image is A-bot is seeing. To do this a linear layer is used to generate an image representation prediction $\hat{y}_t$ from the current encoded state $\hat{y}_t = f(S_t^Q)$. In this paper the image features were extracted from a VGG-16 network pretrained on ImageNet, at layer fc7. So the aim is that $\hat{y}_t$ is close representation of fc7 features of A-bot image y. Using the torchvision pretrained VGG-16 model, we extracted these features for the entire VisDial dataset.

A-bot has a similar structure to Q-bot with slight differences since it also has the image information. First the A-bot has a question encoder which receives a question q_t from Q-bot and encodes it via LSTM Q_t^A. Just like Q-bot, A-bot also has a fact encoder, which encodes (q_t, a_t) pairs via an LSTM to get F_t^A. The idea of the fact encoder is for the models to make a mental note of entities that have been referenced (questions and answers) before. A-bot also has a state/history encoder, but this time the input are the image VGG-16 representation y and the encode question Q_t^A and the previous fact encoding F_{t-1}^A. Therefore $((y, Q_1^A, F_0^A), \dots (y, Q_t^A, F_{t-1}^A)) \leftarrow S_t^A$. The reason for this is that A-bot needs to have a sense of history to understand the current question and look at the image to answer this question. The last component of A-bot is the answer decoder which is an LSTM that takes the state encoding S_t^A and generates an answer by sampling over a vocabulary.

3.2 Reinforcement Learning Setup

Q-bot and A-bot comprise two “constituent agents" that are required to cooperate to perform well at the task. The action space of both agents consists of all possible output sequences based on the number of words in the vocabulary. The action is discrete and can be infinitely be large since the sequence can be any length. Because both agents were pretrained in the VisDial dataset, both have the same vocabulary shared between them. However, Q-bot needs to guess which image A-bot is looking at, because of this Q-bot predicts $\hat{y}_t$ as well.

The state of each agent is assymetrical because Q-bot does not have access to the image given to A-bot. Because of this, the Q-bot state is described as the following $s_t^Q = [c, q_1, a_1, ..., q_{t-1}, a_{t-1}]$ where c is caption of the image and (q_t, a_t) is the question and answer pair. The state for A-bot is

the following $s_t^A = [I, c, q_1, a_1, ..., q_{t-1}, a_{t-1}]$. The policy for both A-bot and Q-bot operates under stochatic policies $\pi_Q(q_t, s_t^Q; \theta_Q)$ and $\pi_A(a_t|s_t^A; \theta_A)$. The parameters θ_A and θ_Q are learned from two seperated neural networks. Because the game is completly cooperative, both agents received the same reward. The reward is the following:

$$r_t(s_t^Q, (q_t, a_t, y_t)) = l(\hat{y}_{t-1}, y) - l(y_t, y)$$

What the reward is saying if the guesses (distance between the image representation and generated image representation) is getting better over time then the reward will be positive if otherwise, the agents receives a negative reward.

3.3 Training

Before having Q-bot and A-bot interact in the "Guess What" environment, the authors decided to pretrain the agents. This was done in order to deal with the difficulty of perceiving images correctly while discovering a human-interpretable language and communication strategy. The agents were both pre-trained on the VisDial train split dataset and later finetuned on this environment using the REINFORCE algorithm:

$$\nabla_{\theta_A} J = \mathbb{E}_{\pi_A, \pi_Q}[r_t(.) \nabla_{\theta_A} log \pi_A(a_t|s_t^A)] \tag{1}$$

$$\nabla_{\theta_Q} J = \mathbb{E}_{\pi_A, \pi_Q}[r_t(.) \nabla_{\theta_Q} log \pi_Q(a_t|s_t^Q)] \tag{2}$$

Both A-bot and Q-bot were pretrained on the train split for 15 epochs. After the 15th epoch reinforcement learning rounds are gradually introduced into the dialog. Where at the start there are 9 rounds of supervised learning then gradually which are gradually annealed to 0. Towards the end of training all rounds are done on reinforcement learning mode. This curriculum ensures that the agent team does not suddenly diverge off policy, just in case if one of the questions or/and answers could be generated incorrectly. The different modes of training are implemented by us, which includes, supervised learning for Q-bot (pretraining mode), supervised learning for A-bot (pretraining mode), and supervised learning for both A and Q bots, then slowing switching to reinforcement learning.

4 Experiments

The first set of experiments were sanity checks, to make sure that the implementation of the model is correct and converging. These sets of experiments were done on a small subset of the training set, we were looking to see if the model overfitted in a tiny dataset. As soon we establish the performance of the model, we then started the pre-training process for both A-bot and Q-bot on the larger train set. We pre-trained the models for 15 epochs and evaluated them on the dataset.

4.1 VisDial Dataset and Sanity Dataset

The dataset used in this paper is [1] VisDial dataset, which was collected by pairing two subjects in Amazon Mechanical Turk to chat about the an image. This dataset contains 68k images from COCO dataset and a dialog of 10 rounds for each image. In total, there are 680,000 question and answers pairs. During the time of this work, VisDial version 0.5 was in use, however, version 1.0 is now available and this is what we're planning to use for this project.

VisDial version 1.0 training set contains over 120,000 images and dialogs, where each dialog contains 10 answers and questions sequences. Where we extracted VGG-16 fc7 features on all the images on this dataset using the torchvision pretrained VGG-16 on ImageNet. We have removed punctuation, made everything lowercase and tokenized all the questions, captions and answers in the entire dataset using NLTK framework. A vocabulary was created using train split captions, answers and questions, where the unknown, start and end tokens are also included, which at the end results in a vocabulary size of approximately 20,000 words. Using this vocabulary during runtime our code maps the words that appear in captions, questions and answers into indices to their respected word to index.

We created three datasets to run different experiments which are the sanity dataset, 20k example dataset and 100 example dataset. To construct the sanity dataset, we randomly selected 50 images and

dialogs from the train split. The sanity dataset was used for development and debugging purposes. We then created a 20k exmaples dataset, where we randomly selected 10k training examples, 9.5k validation examples and 2.5k training examples. For further analysis we created the 100 examples dataset which we selected randomly as well.

4.2 Evaluation and Metrics

The authors of the original paper [1] and the VisDial dataset [2] used the Mean Reciprocal Rank, Recall @ 10 and Recall @ 5 metrics. The Mean Reciprocal Rank takes into consideration only on the single highest-ranked relevant item. For this case an acceptable answer for the question given. VisDial for train and validation set provides 100 candidates answers for each question. The model should be able to rank the candidates answers by putting the correct answer towards the top. VisDial does not provide these candidates for the test set, however, anyone is welcomed to submit their code to their evaluator service for evaluation on the test set. For the scope of this work, our test set consists of examples that we have held out from the training set of the VisDial dataset.

We used the log-probabilities for each of the candidates from the A-Bot decoder to rank them based on their score. After this, we apply the MMR and Recall@k metrics. We implemented these metrics and the scoring of each candidate.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{3}$$

4.3 Experiment Details

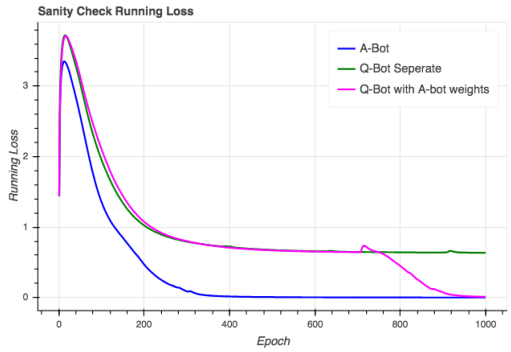
As soon as it was established that model was working, we started pretraining our A-Bot and Q-Bot on the 20k dataset. Which took 2 days train each, after the training was completed we evaluated our model (using MRR, Recall@5 and Recall@10) and generated its dialog. We noticed a strange behavior in the output of these pre-trained models. Q-Bot's questions were very repetitive. When we looked at the states of Q-Bot we saw that they were almost all identical thus the questions outputted were also very similar. This is the reason we decided to create a very small 100 example dataset in which we could debug this problem. Due to time constraints, we're currently running QA-Bots on the reinforcement learning finetuning experiments.

We used a batch size of 16, Adam optimizer with the learning rate of 1e-3 which we decay the learning rate every epoch by .999. The reason for this is because we noticed that after a while the loss of the model trained on the 20k dataset started to get stuck and eventually stopped decreasing. When we saw this, we took the latest checkpoint and restarted the training with the learning rate decay which helped very well. We also clipped our gradients between -5 and 5 as stated in the work to help avoid gradient explosion. Training A-Bot was straightforward, but Q-Bot was a challenging task as it is both learning to ask questions and predict the image. After much tinkering, we learned that scaling up the Q-Bot decoder loss function by 300 helped to combat the repetitive nature of the questions.

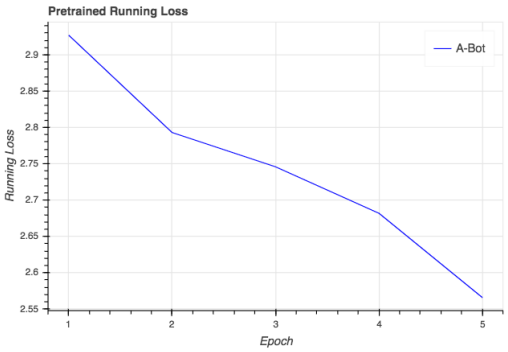
4.4 Results

We evaluated models that were trained on the sanity, 100 and 20k Dataset, using the MRR, Recall@5, Recall@10 metric. We notice that once a dialog generates incorrect answers or questions, then the errors quickly propagate and the conversation becomes uneffective. For example, suppose that Q-Bot generates a question incorrectly, then A-Bot will also generate an incorrect answer which is then encoded as a fact in Q-Bot. Thus the errors compound themselves as soon as the first mistake in the dialog occurs.

We saw that adding the scaling factor to the Q-Bot decoder losses helped with repetitive behavior but a the same time it hurt validation and test perofrmance. This is most likely becase now that we're scaling up the loss, the model overfits and we need less training time or perhaps the scaling factor was to big. In any case, we've broken the results for both of these cases, scaling and non-scaling.



(a) Sanity dataset A-Bot and Q-Bot seperate and Q-Bot with A-bot pretrained weights running loss for 1000 epochs



(b) Pretrained A-Bot running loss for 5 epochs

Table 1: Overall Performance

Model	Dataset	MRR	Recall@5	Recall@10	Epochs
SL QAbots	Sanity Dataset	44.87	46.00	59.80	1000
SL QAbots	100 Dataset	22.98	29.00	41.00	100
SL QAbots with scaling factor	100 Dataset	19.24	24.00	41.99	100
SL QAbots	20k Dataset	38.11	42.34	50.52	15
SL + RL [1]	VisDial version 0.5	43.8	53.08	60.22	15

5 Analysis

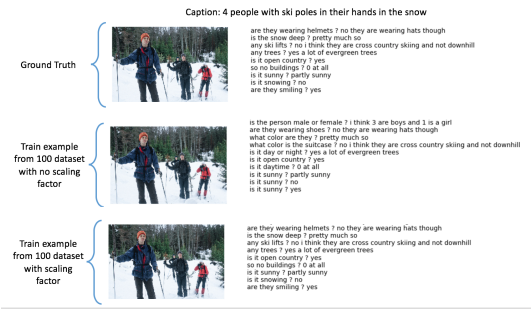
We see the repetitive behavior across all pre-trained models. Adding the scaling factor helps the questions be less repetitive but it seems that Q-Bot now has a hard time deciding which question is informative. The reason for this is that while the scaling factor helps because it overfits to the train set it learns that each round is different question but it doesn't generanized what questions should it should asking while A-Bot is doing its best to answer these incorrect questions. Training Q-Bot is challenging because it has the same structure as A-Bot but is expected to do more. Q-Bot is the driving force of the conversation while A-Bot is just needs to answer correctly. The way dialog history are embedded in Q-Bot should be different, because of this the current architecture struggles to capture the difference between facts and incorrect data.

The authors claim that adding reinforment learning helped tune down the repetitivenes of the model. We think the reason for this is that the words sampled are not greedy as in purely supervised learning mode of training, instead, the policy of the model learns to exploit and explore not just exploit. We started the fine-tuning on the model pre-trained on the 20k dataset. However, we noticed that current policy checkpoints are unstable while one point it perform ok while another it perform as random.

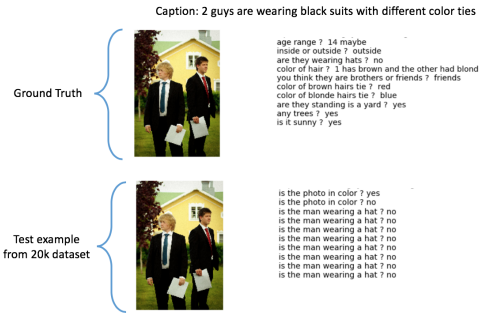
Another behavior we notice is that sometimes the reinforcement learning method undoes what the model learned to do during pretraining, generating questions and answers. This way of learning a policy without a model of enviroment makes the policy to be strongly depended on enviroment interactions to learn. Hence we conclude that the reason the our current reinforcement learning model is unstable is because it needs more interactions before it becomes stable.

Table 2: 100 Dataset Performance breakdown

Model	Train MRR	Test MRR	Validation MRR
SL QAbots with scale factor	53.47	19.24	21.71
SL QAbots	44.74	22.977	20.02



(c) Example taking from 100 dataset, this show the repititve behavior without scaling factor with the scaling factor it overfits



(d) Example taking from 20k dataset test set

6 Conclusion

In this work, we implemented the goal-driven multi agents for Visual Dialog. Pre-trained the model using VisDial dataset split into three small datasets; sanity, 20k and 100 dataset. We evaluated each of these dataset and learned that while training A-Bot is straightforward and performed as expected, Q-Bot is harder to train because it has more responsibilities. We also learned adding a scaling factor to the Q-Bot decoder loss helps with the repetitiveness of the question but makes the model overfit faster. We suspect that the achitecture of Q-Bot should be different to capture its differing responsibilities.

We also experimented with reinforcement learning fine-tuning and learned that model is currently unstable. We hypothesize that this is because the model is learning a policy directly from interaction with the environment and needs a high amount of samples to converge.

Our next steps are to experiment with different Q-Bot architectures by adding self-attention on the history encoder. We also are going to experiment with image ranking reward function. The current reward function only gives reward by answering the question "Am I getting closer?", but we would like to add an extra component that asks "Have I figured it out or am I close enough?". Instead of expecting Q-Bot to create a perfect representation of the image, which humans can't do, we should give Q-Bot a list of images to choose from by image ranking and if the top image is the one A-Bot is looking at then it should get a reward. Finally, we wonder if limiting the number of rounds that the bots are allowed to play would have any effect on the repetitive nature of Q-Bot. Intuitively, if we force Q-Bot to guess the image faster and faster then it must get better at asking the right questions.

References

[1] Jose M.F Moura Stefan Lee Abhishek Das, Satwik Kottur and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[2] Khushi Gupta Avi Singh Dshraj Yadav Jose M.F Moura Devi Parikh Abhishek Das, Satwik Kottur and Dhruv Batra. Visual dialog. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[3] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[4] Akshay Kumar Gupta. Survey of visual question answering: Datasets and techniques. *CoRR*, abs/1705.03865, 2017.

[5] Williams Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning 8.3-4*, 1992.

[6] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *CoRR*, abs/1610.01465, 2016.

[7] Sahar Kazemzadeh. Referitgame: Referring to objects in photographs of natural scenes. *Proceedings of the 2014 conference on emprical methods in natural language processing*, 2014.

[8] David Lewis. Convention a philosophical study. *John Wiley and Sons*, 2008.

[9] Harm De Vries. Guesswhat? visual object discovery through multimodal dialoue. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.