Conversational and Image Recognition Chatbot

Stanford CS224N Custom Project

Shengyang Su
billysu@stanford.edu

Abstract

This paper proposes a chatbot framework that adopts a model which consists of natural language processing and image recognition technology. Based on this chatbot framework, neural encoder-decoder model is utilized with Late Fusion encoder[1] and 2 different decoders(generative and discriminate). The chatbot is able to detect object in the image, tell about and recognize the image, later on the chatbot is also able to answer the questions about this image. It is a task where an AI agent can hold meaningful dialogue with humans in natural language about a visual content, specifically images. The basic workflow is that given an Image (I), current question (Q) and a history of Question and answers (H), the agent should be able to generate the answer of the current question.

1 Introduction

Ever since the birth of AI and computer vision, modeling conversations remains one of the field's challenges, especially to combine both natural language processing and image recognition. Chatbots are now widely used as part of platform as applications like Apple's Siri[2], Google's Google Assistant[3] or Microsoft's Cortana[4]. A conversational chatbot is an application that is able to communicate with humans using natural language. An image recognition deep learning based chatbot is an application to recognize the image which the user uploaded and answer the questions about the image. The main problem domain of my project is building a image recognition chatbot, which is capable of recognize the object in an image and generating the best response for any the user's query about the image. In order to achieve this goal, the chatbot needs to detect the object in the image and have the related dialog of the image after training, also have understanding of the sender's messages so that it can predict which sort of response will be relevant and it must be correct lexically and grammatically while generating the reply.

2 Related Work

Conversational Modeling Chatbot For text based chatbot, there are two main approaches for generating responses in chatbot. The traditional approach is to use hard-coded templates and rules to create chatbots. The more novel approach was made possible by the rise of deep learning. Neural network[4] models are trained on large amounts of data to learn the process of generating relevant and grammatically correct responses to input utterances. A common challenge with most conversational chatbot models is that they don't have any memory. Although architectures have been devised in order to take into account previous dialog card turns, they are still limited. For example they can't consider previously happened dialog cards. A user would expect from a chatbot that once he/she talks about something, the chatbot will mostly remember the dialog, as is normal in human-human conversations. The chatbot should be able to learn new things about the user in order to make the experience personalized.

Object Detection on COCO[5] COCO stands for Common Objects in Context. COCO was an initiative to collect natural images, the images that reflect everyday scene and provides contextual information. In everyday scene, multiple objects can be found in the same image and each should be labeled as a different object and segmented properly. COCO dataset provides the labeling and segmentation of the objects in the images. A machine learning practitioner can take advantage of the labeled and segmented images to create a better performing object detection model.

Object detection is the task of detecting instances of objects of a certain class within an image. The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN and Cascade R-CNN. The most popular benchmark is the MSCOCO dataset. Models are typically evaluated according to a Mean Average Precision metric.

The Encoder-Decoder Model When applying neural networks to natural language processing (NLP) tasks each word (symbol) has to be transformed into a numerical representation [6]. This is done through word embeddings, which represent each word as a fixed size vector of real numbers. Word embeddings are useful because instead of handling words as huge vectors of the size of the vocabulary, they can be represented in much lower dimensions. Word embeddings are trained on large amounts of natural language data and the goal is to build vector representations that capture the semantic similarityb between words. More specifically, because similar context usually is related to similar meaning, words with similar distributions should have similar vector representations. This concept is called the Distributional Hypothesis[7]. Each vector representing a word can be regarded as a set of parameters and these parameters can be jointly learned with the neural network's parameters, or they can be pre-learned.

Mask R-CNN[7] Mask R-CNN extends Faster R-CNN to solve instance segmentation tasks. It achieves this by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. In principle, Mask R-CNN is an intuitive extension of Faster R-CNN, but constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is evident in how RoIPool, the de facto core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, Mask R-CNN utilises a simple, quantization-free layer, called RoIAlign, that faithfully preserves exact spatial locations. Secondly, Mask R-CNN decouples mask and class prediction: it predicts a binary mask for each class independently, without competition among classes, and relies on the network's RoI classification branch to predict the category. In contrast, an FCN usually perform per-pixel multi-class categorization, which couples segmentation and classification.

Self-Attention Generative Adversarial Networks Traditional convolutional GANs generate high-resolution details as a function of only spatially local points in lower-resolution feature maps. In SAGAN, details can be generated using cues from all feature locations. Moreover, the discriminator can check that highly detailed features in distant portions of the image are consistent with each other. Furthermore, recent work has shown that generator conditioning affects GAN performance.

3 Approach

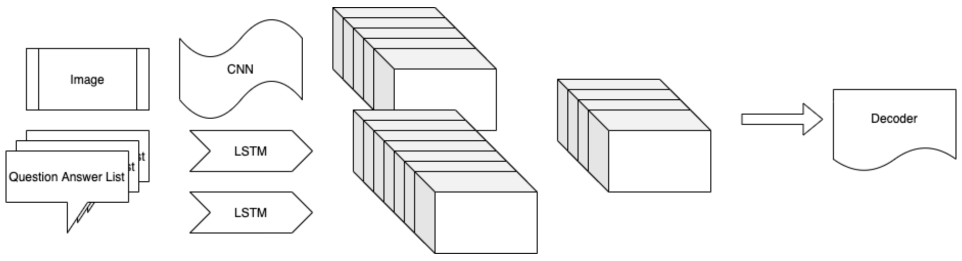


Figure 1: Late Fusion Encoder Model Framework

In this project, the model for the image reorganisation chatbot is based on a encoder-decoder framework, composed of two parts 1. an encoder that converts the input(images and question/answers) into a vector space, 2. a decoder that converts the embedded vector into an output(answers).

Encoders: One encoder is being used as a novel encoder methodology called late fusion that convert inputs into a joint representation. For this problem, we have handled the image feature extraction part. To process language a sequence to sequence model had to be used as questions and answers can be of varied lengths. A sequence to sequence model contains an Encoder and a Decoder both a kind of RNN called Long Short Term Memory (LSTM). They process each question and answer sequentially. Unlike traditional Encoder, our encoder has to include the features from image and the history of question and answers. There are several proposed methods and each have their pros and cons. Considering the implementation difficulty and accuracy of the results I decided to go for Late Fusion (LF) Encoder.

• Late Fusion (LF) Encoder: LF Encoder is discussed in this paper[1], the basic ideas is that we treat history(H) as a long string with the entire history concatenated. The current question(Q) and H are separately encoded using two different LSTMs. The Image(I), H and Q are are concatenated and linearly transformed to a desired size of joint representation. We encode the image with a VGG-16 CNN, question and concatenated history with separate LSTMs and concatenate the three representations. This is followed by a fully-connected layer and tanh non-linearity to a 512-d vector, which is used to decode the re-sponse. In this encoder, we treat each feature as a long string with the entire history concatenated. Question and Question-Answer pair are separately encoded with 2 different LSTMs, and individual representations of participating inputs are concatenated and linearly transformed to a desired size of joint representation.

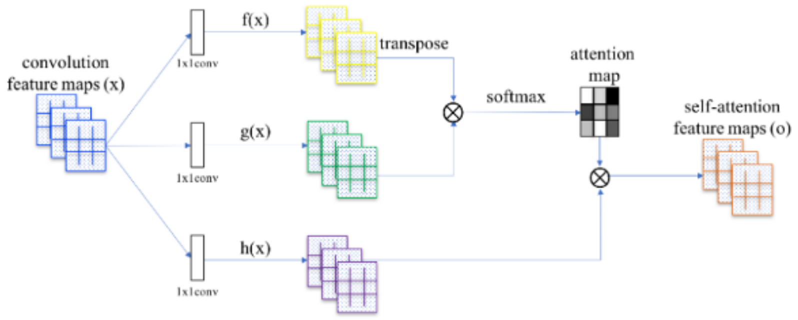


Figure 2: Self-Attention Module (Source: SAGAN Paper)

Feature Exctration: In this part, let's also discuss the model we are using for object detection and localization. Mask-RCNN [9] is an extension of Faster-RCNN. Apart from object detection and classification Mask-RCNN also produces instance segmentation masks. Mask-RCNN introduces a new branch for the instance segmentation which outputs the masks of detected objects. The instance segmentation branch is a fully connected network, which provides pixel to pixel segmentation on the basis of the Region of Interests. A fully connected network is a network in which every node is connected to every other node. As detecting and overlaying a mask over the object is much more complex than just drawing the bounding boxes around them, Mask-RCNN introduces a new layer called ROI-Align layer in place if ROI-pooling layer. Some Things that maskrcnn does really well: 1. everything's a hyper param; 2. logging - for lots of feedback; 3. single tqdm output for training. Figure 4 represents the architecture of Mask-RCNN.

Decoder: there are two decoders that are being used in the framework. There can be different types of decoder as well namely Generative and Discriminative. According to the results reported from the paper the Discriminative Decoder's performance was better. The discriminative decoder outputs a probability of the answer options. The discriminative decoder is better on metric but generative decoder is more realistic. While doing error analysis, I found that the model used very little information from the image, as the magnitude of the gradient during backpropagation on the image was very low. This would primarily have been because, we were not fine tuning the VGG16 for the task. Fine Tuning would have made the complete model understand what parts are important in the image. I then integrated a Self-Attention module inspired from the SAGAN[??] paper. On training the complete Encoder-Decoder Network with Self-Attention stabilized the training a lot and decreased the loss further and improved the performance on the used metric on the validation set. The gradients were now travelling all the way back to the image. Hence, proving that the information from the image was also utilized.

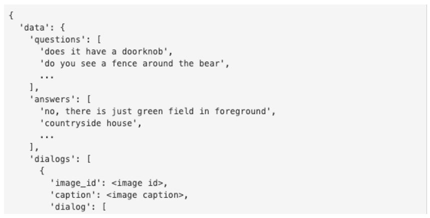
• Generative (LSTM) decoder: where the encoded vector is set as the initial state of the Long Short-Term Memory (LSTM) RNN language model. During training, we maximize the log-likelihood of the ground truth answer sequence given its corresponding encoded representation (trained end-to-end). To evaluate, we use the model's log likelihood scores and rank candidate answers. Note that this decoder does not need to score options during training. As a result, such models do not exploit the biases in option creation and typically underperform models that do, but it is debatable whether exploiting such biases is really indicative of progress. Moreover, generative decoders are more practical in that they can actually be deployed in realistic applications.

• Discriminative (softmax) decoder: computes dot product similarity between input encoding and an LSTM encoding of each of the answer options. These dot products are fed into a softmax to compute the posterior probability over options. During training, we maximize the log-likelihood of the correct option. During evaluation, options are simply ranked based on their posterior probabilities.

4 Experiments

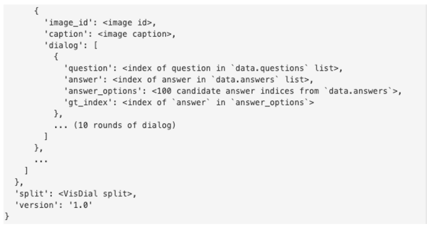
4.1 Data

My goal is to train an ML model on the COCO Dataset. Then be able to generate my own labeled training data to train on. So far, I have been using the maskrcnn-benchmark model by Facebook and training on COCO Dataset 2014. Getting the data The COCO dataset can be download here: https://cocodataset.org/download I am only training on the 2014 dataset. Here is an example for the COCO data format JSON file which just contains one image as seen the top-level "images" element, 3 unique categories/classes in total seen in top-level "categories" element and 2 annotated bounding boxes for the image seen in top-level "annotations" element.



```
{
  'data': {
    'questions': [
      'does it have a doorknob',
      'do you see a fence around the bear',
      ...
    ],
    'answers': [
      'no, there is just green field in foreground',
      'countryside house',
      ...
    ],
    'dialogs': [
      {
        'image_id': <image id>,
        'caption': <image caption>,
        'dialog': [
```

Figure 3: Sample Data Part 1



```
      {
        'image_id': <image id>,
        'caption': <image caption>,
        'dialog': [
          {
            'question': <index of question in `data.questions` list>,
            'answer': <index of answer in `data.answers` list>,
            'answer_options': <100 candidate answer indices from `data.answers`>,
            'gt_index': <index of `answer` in `answer_options`>
          },
          ... (10 rounds of dialog)
        ]
      },
      ...
    ]
  },
  'split': <VisDial split>,
  'version': '1.0'
}
```

Figure 4: Sample Data Part 2

Data splitting: With the data encoded, we can now split it into training and testing sets. The training set will be used to train the model while the testing set will be used for evaluating its performance on unseen data. This can be done using a stratified approach, whereby of the patterns in the tags are well represented in the testing set. I split the 83k into 80k for training, 3k for validation, and use the 40k as test.

Data Preprocessing: The data handling of this type of dataset is quite complex and cumbersome. As for each image there are ten rounds of question and answers. To meet the deadlines I set for myself, I extracted VGG16-relu7 features as a input to language processing models. The dataset was handled in such a way that for each coco image id we get its image features and all the questions to iterate on. This strategy greatly helped in handling such large and complex dataset. Also need to spell-correct VisDial data, convert digits to words, and remove contractions, before tokenizing using the Python NLTK [10]. We then construct a dictionary of words that appear at least five times in the train set.

4.2 Experimental details

The leanring models are implemented with python package Torch [11]. All the Model hyperparameters are customized by early stop based on Mean Reciprocal Rank (MRR) metric. The dimension of the LSTMs are 2-layered with 512-dim hidden states. The model trained to learn 300-dim embeddings for words and images. All of word embeddings are used across question that are asked, history, and decoder LSTMs. For better tuning the performance of the model, Adam optimizer with slicely changed learning rate. At each iterations, gradients are set between to [5, 5] to avoid explosion.To better visualize the process and result, I utilized Django to develop a user friendly web interface. Below is a simple backend and AI interaction in terms of architecture.

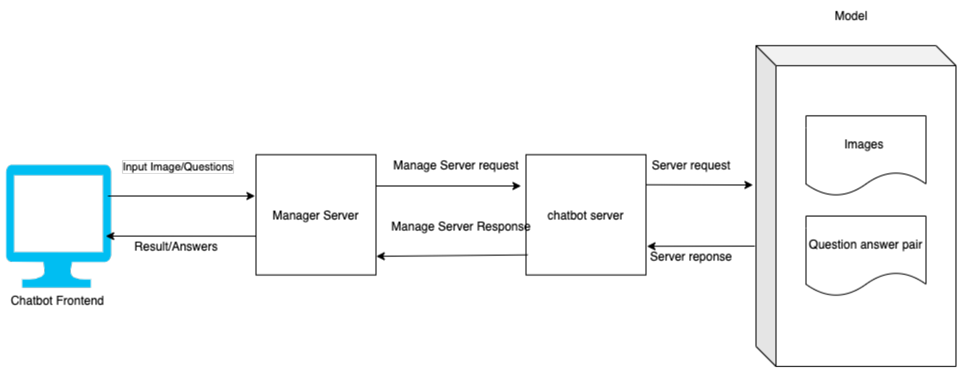


Figure 5: System Archetecture

4.3 Evaluation method

Evaluation and Performance: One fundamental challenge in dialog systems is evaluation. It is an open problem in NLP to evaluate the answers. Metrics like BLEU, ROGUE are there but they are known to correlate poorly with human judgement. To challenge this issue, I used the metric evaluating individual responses. At test time, the model is given an Image(I), History (H) and a set of 100 candidate answers. The model ranks the answers and it is then evaluated on Mean Reciprocal Rank of the human response (higher the better) and Recall@k i.e. existence of the human response in top-k ranked responses (lower the better). For evaluation, conducted by different decoder c comparison measured by mean reciprocal rank (MRR), recall@k for k = {1, 5, 10} and mean rank. Note that higher is better for MRR and recall@k, while lower is better for mean rank. As we can see, most discriminative models significantly outperform generative models. I also found out that model with complcated history has better result than the one without well explained history based on different datasets.

Performance on v1.0 validation split (trained on v1.0 train + val):

Model	R@1	R@5	R@10	MeanR	MRR	NDCG
lf-disc-faster-rcnn-x101	0.4617	0.7780	0.8730	4.7545	0.6041	0.5162
lf-gen-faster-rcnn-x101	0.3620	0.5640	0.6340	19.4458	0.4657	0.5421

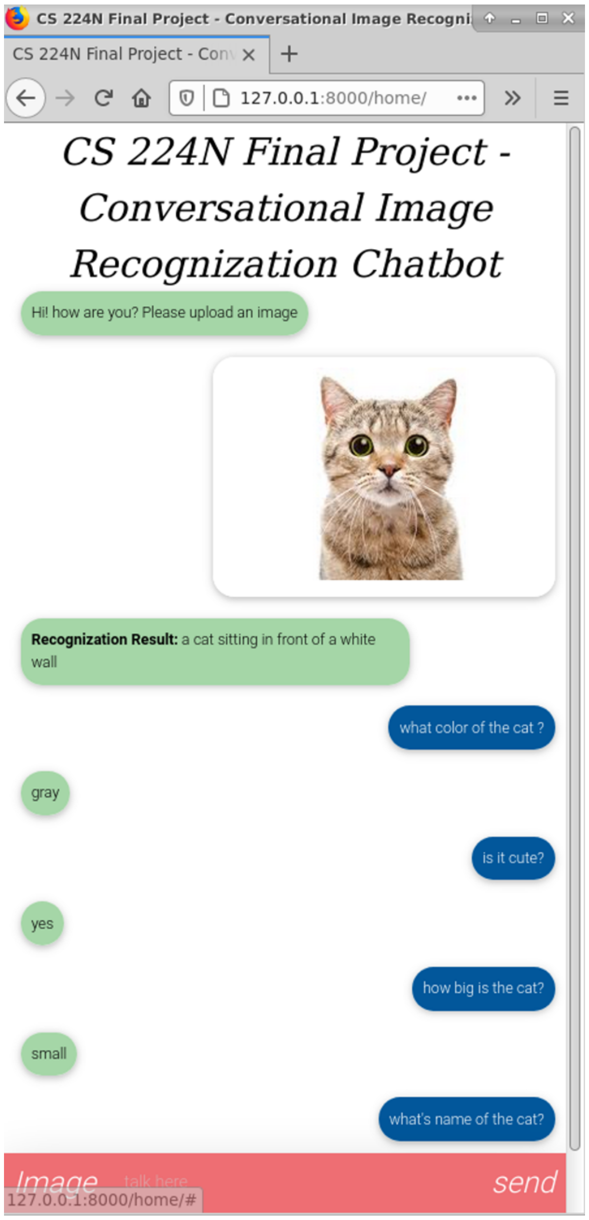
Performance on v1.0 test split (trained on v1.0 train + val):

Model	R@1	R@5	R@10	MeanR	MRR	NDCG
lf-disc-faster-rcnn-x101-v1.0	0.4700	0.7703	0.8775	4.954	0.6065	0.6092

Performance on v0.9 validation split (trained on v0.9 train):

Model	R@1	R@5	R@10	MeanR	MRR
lf-disc-faster-rcnn-x101-v0.9	55.05	0.8398	91.58	3.69	67.94

Reulst Analysis: Reulst Analysis: Based on testing regarding to NLP part, when the question asked to the model ismore complicated, the result or response become not accurate, this uncertainty may caused by the image context history itself, or due to the lack training of data, to improve this we need to make sureeach trained model get enough data to train



CS 224N Final Project - Conversational Image Recogni

CS 224N Final Project - Conv

127.0.0.1:8000/home/

CS 224N Final Project - Conversational Image Recognization Chatbot

Hi! how are you? Please upload an image

Recognization Result: a cat sitting in front of a white wall

what color of the cat ?

gray

is it cute?

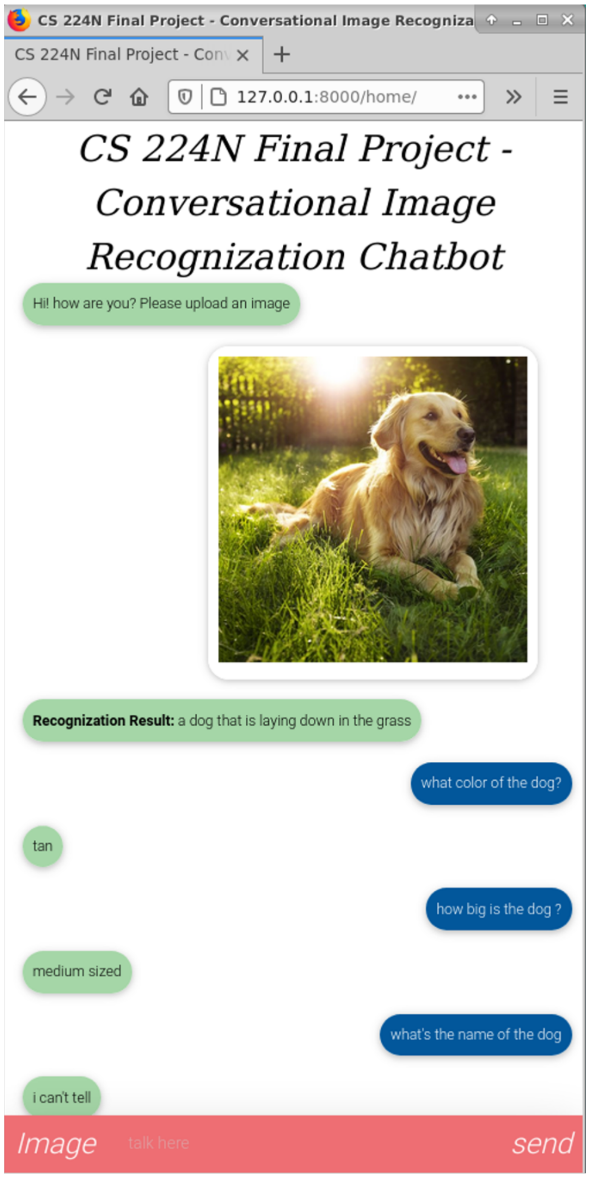
yes

how big is the cat?

small

what's name of the cat?

Figure 6: Result 1



CS 224N Final Project - Conversational Image Recogniza

CS 224N Final Project - Conv

127.0.0.1:8000/home/

CS 224N Final Project - Conversational Image Recognization Chatbot

Hi! how are you? Please upload an image

Recognization Result: a dog that is laying down in the grass

what color of the dog?

tan

how big is the dog ?

medium sized

what's the name of the dog

i can't tell

Figure 7: Result 2

5 Conclusion

The purpose of this project is to utilize natural language processing and computer vision models for efficient identify and answer following question to any images and following up questions, this could apply further in organizations, schools, hospitals and military areas. We are utilizing COCO dataset to train the data, images are fused together to get a combined more informative output for detection of a doubtful presence. We are using Encoder-decoder CNN architecture for fusion of images and Resnet[15] architecture for object detection and localization. Localization of object or persons is done using Mask-RCNN model which not only localizes the object, but also provides a mask for localized object. I believe this area has huge impact in natural language processing and visual recognition in industry or academic.

References

[1] arXiv:1611.08669

[2] Apple (2017). Siri. https://www.apple.com/ios/siri/. Accessed: 2017-10-04.

[3] Google (2017). Google assistant. https://assistant.google.com/. Accessed:2017-10-04.

[4] Microsoft (2017a). Cortana. https://www.microsoft.com/en-us/windows/cortana. Accessed: 2017-10-04.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. DollÃ¡r, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In ECCV, 2014. 2, 3

[6] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb):1137–1155.for neural dialogue generation. arXiv preprint arXiv:1701.06547.

[7] Harris, Z. S. (1954). Distributional structure. Word, 10(2-3):146–162

[8] maskrcnn-benchmark https://github.com/facebookresearch/maskrcnn-benchmark

[9] arXiv:1805.08318

[10] NLTK. http://www.nltk.org/.

[11] Torch. http://torch.ch/

[12] Django. https://www.djangoproject.com/

[13] Futurism, "The History of Chatbots," [Online]. Available: https://futurism.com/images/the-historyof-chatbots-infographic/.

[14] Chatbots.org, "Smarterchild," [Online]. Available: https://www.chatbots.org/chatterbot/smarterchild/.

[15] https://arxiv.org/pdf/2009.07190.pdf