Problems and Limitations of Current Chatbot Models

A conversation is a complex concept that can’t be modeled with a simple encoder-decoder architecture, that simply take in an input and transform it to an output. The problem is that they assume that only 1 correct output exists, which is true for neural machine translation (NMT) [1] or text summarization [2], but absolutely not true for conversations [3,4]. This architecture is better suited for question-answering where we encode a specific question and get a specific answer as output of decoding. The task of coming up with a response to a conversation history is not well defined. Because of this if we show an encoder-decoder model a lot of similar inputs with completely different outputs it will learn an amalgamation of the response, thus it will output an average of all the responses, which is usually generic, and dull. We can see this in many papers where authors show that the generated responses often times tend to be generic like “I don’t know” [4]. My presumption is that the model tends to learn these responses precisely because in the vector embedding space they are at the center of the coordinate system.

Another problem with chatbot models is related to evaluating these models. It has been shown that typical score that are good for NMT like Blue and perplexity don’t correlate at all with human judges when applied to chatbots [5,6]. The reason for this is exactly the same as before, namely that these metrics assume that there is only 1 correct response for each input utterance. While there have been proposals to metrics that correlate more with humans [5], I haven’t seen anyone to use them in the literature. In consequence of the lack of good metrics researchers rely on human judges to measure the performance of a chatbot model, which can be biased and have high variance. Thus, I think it’s extremely important to publish the resulting chatbot program of a research paper, to let the public try it out, because a handful of (sometimes cherry-picked) examples are not enough to see how well the model performs.

In conclusion, I think that there aren’t enough features involved in conversation modeling to make distinctions between response candidates.

Proposals Towards Solving the Issue

Since human-like conversation is basically AGI it is extremely hard to tackle it up front. We would have to make a very complex model and include a lot of knowledge about the word in meaningful ways. Until we can’t do this we have to make do with an encoder-decoder architecture. In this setting, I propose that additional features and priors should be taken into consideration when modeling a conversation. Since maximizing the MLE objective is essentially maximizing the probability of a reply given an utterance there should be other prior probabilities that the reply can be conditioned on. Ideally, we should feed in features besides word embedding into the decoder such that for each input only one output exists. A number of similar augmentations have been done before [7,8,9,22]. They all try to embed some additional information and feed it into the seq2seq model like personality, mood and topic categories. While they all show that generated responses from these models are somewhat more diverse than a basic seq2seq model I still think that the conversation model is ambiguous.

While at the birth of the seq2seq architecture, conversation models were either trained on utterance-reply pairs or by concatenating a couple of previous turns into a vector, nowadays there are many efficient models and techniques to take into account a long conversation history like attention [10,14], hierarchical encoder decoder architectures [11,13] and convolutions over whole sentences [12]. Thus, I consider the problem of conditioning a reply on the entire conversation mostly solved.

Speaker-addressee representations

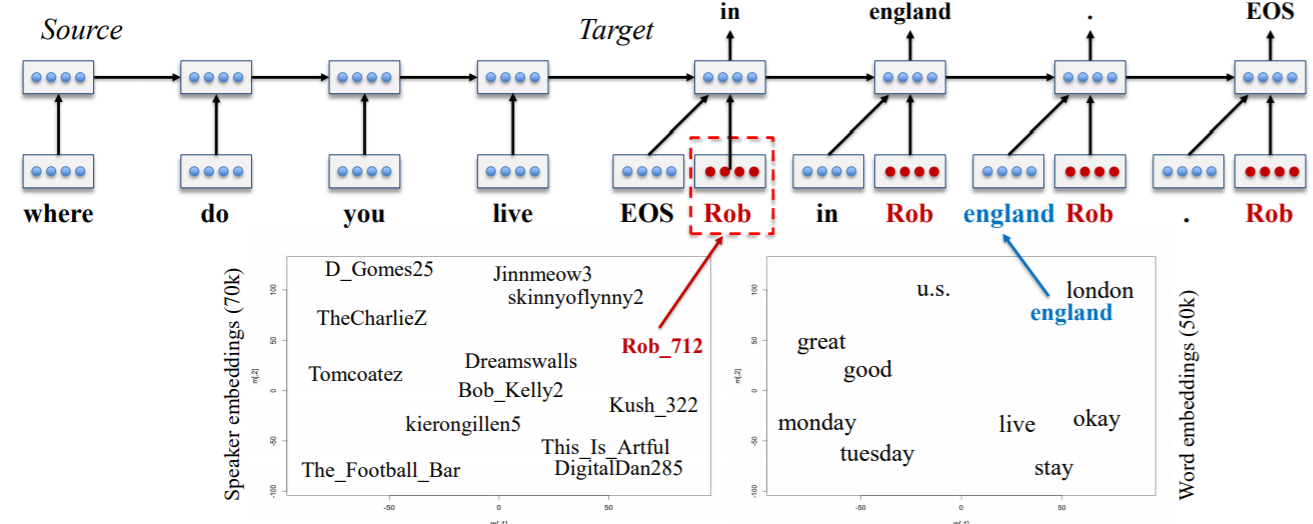
I propose a joint approach. Ideally, we would have a huge dataset composed of conversation utterances which are annotated with speaker and addressee tokens. The speaker and addressees could be represented by a vector or matrix similar to word embeddings which the model can learn. In addition, I think that the mood of the reply should also be fed as an input to the model. The mood can be extracted with an RNN trained on this task. By adding these priors to the network, it will make the dataset more unambiguous. If we accept the assumption that a response usually depends on the conversation history, on the person who is replying and his/her mood and on the person who he/she is replying to then we have a dataset which is unambiguous, and for every input utterance only 1 correct output utterance exists. Thus, after training the generated responses of the model should be specific to the speaker and addressee representations and on the speaker’s mood. All these priors assure that the generated replies will be diverse and conversation-like, not just an average of all possible responses.

World-knowledge representations

There are still problems with this approach however. The speaker and addressee “embeddings” should have a lot of parameters because by intuition the personality of one depends on all of his/her prior experiences up to that point, which is a lot of information to encode into a simple vector or even a matrix. Just think about the fact that we already use ~500 dimensional vectors only to embed words. However, if we would make these personality representations really huge then the network couldn’t be efficiently trained. Even if we add up all conversation-based datasets they would be too small and sparse for the network to learn each personality representation’s parameters. While naïvely this approach should work and I definitely think it should be tried out, we can’t expect an encoder-decoder model to learn complex personalities based on the person’s world knowledge and past experiences just from conversations.

To solve this issue, I propose a second approach. First of all, we keep the priors which I previously mentioned, but we implement personality representations as a learnable sized matrix. The intuition behind keeping this representation smaller is that people generally don’t differ that much in basic knowledge and ways of conversing. Thus, we can encode this information in a 500x500 matrix for example, which is like saying that we can describe in what aspects this person differs from others in 500 words. Secondly, we implement a representation of general world knowledge by encoding a huge amount of text based on general knowledge. How this representation could be added to a seq2seq I haven’t figured out yet, but a naïve approach would be to simply feed it into a separate encoder, and hoping that the backpropagation from the generated response will make this encoder learn a meaningful representation of world knowledge through its parameters.

The actual ways in which such embeddings and encodings should be implemented into an encoder-decoder model can have many variations of course. We can concatenate everything into a giant vector and just feed it into the encoder, or separate some features like mood and feed those directly into the decoder jointly with the word generated in the previous timestep. An example of this is shown below, where personality vectors are fed into the decoder [7].



Temporal conditioning

Sometimes a response is not conditioned on neither of the features and priors mentioned previously. Take for example a simple question like “how are you?” and the reply is “I almost got hit by a car”. Obviously, this response has little to do with the personality of the speaker or any general knowledge about the world. Rather the response is conditioned on outside factors, which are temporary. Temporary because after a week the person will probably not reply with this answer, because “how are you” is a question that asks about the present. There are some ways in which we could deal with these external factors, like changing the person’s personality representation slightly for this specific response or encode the outside factor into some further representation, or even simpler just cut out the conversations where such unobserved factors have a role. Unfortunately, all methods require a dataset that is labeled with labels that show whether the response is based on some external factors, which can probably only be done manually and I currently do not know of such a dataset.

An important part of a conversation is timing and involvement of both parties. Current chatbot models are trained in such ways that they will only, but instantly emit a response utterance when the user writes something. To make them more human-like I propose to add an additional term to the loss function, which is based on the temporal delay between an utterance and a reply. Essentially the chatbot has to guess the reply and also how much time should it wait before replying. Chatlog datasets usually do come with such annotation so the implementation of this feature should be straight-forward. The time-delay can be represented as a vector of probabilities over some time frames. For example the model has 10% confidence that the reply should be delayed by 0 to 10 seconds, and 90% confidence that it should be delayed by 10 to 20 seconds. Such a representation is effective because we can use a simple softmax to get the probabilities of time frames and backpropagation can be done by comparing the predicted time-delay vector with the one-hot ground-truth timing vector. Not only would these cause a human-like delay (which is tunable) in the chatbot’s responses it would also allow a chatbot model to self-feed a conversation history. Regardless of whether the user has said something or not, the conversation history can be constantly fed into the model at a fixed rate (each couple of seconds for example), and a reply can be generated together with a temporal delay. This would make the chatbot further the conversation by itself without waiting for user interaction.

Another addition that is tied to temporal conditioning is real-time model updates. Basically, at each turn the user reply can be backpropagated through the network, which would make the model update its weights and speaker / addressee representations as the conversation goes on, thus remembering the dialog. This could either replace or complement the need for taking into account long conversation histories with hierarchical or attentional models. It would be especially useful for market deployment of chatbots, where users expect the chatbot to remember what they said 1000 conversation turns ago, which can’t be achieved through only a hierarchical model.

Model Proposals

I propose several models which take the general form of an encoder-decoder architecture. I mix RNN and CNN hierarchical layers in a number of ways in order to preserve good information flow through the network. I also mix up and make variations of where we can feed into the model the previously mentioned additional features. In these models, I will only note basic RNN and CNN layers, but these can be augmented with the following well explored techniques in the seq2seq and chatbot literature:

* Attention at decoding and self-attention for both RNN [14,15,16,17], and CNN layers [12]
* Bidirectionality in RNN layers [10,17]
* Fast-forward connections between RNN layers [17,18] and residual connections in CNN layers [12,19]
* 1-dimensional [12,20] and depth-wise separable convolutions [19]
* Mixture-of-Experts layers [21]

Also, for the sake of simplicity I don’t explicitly note, but bucketing and padding is used wherever it’s required to handle dynamic size inputs [23].

(This section is not finished yet, I will add more architectures later.)

**Fully conv-deconv architecture**

Do this times the number

of previous utterances (n)

CNN encoder

depth

(d)

Speaker, addressee,

utterance feature maps

Reply generated

from softmax

word

embedding

size

CNN decoder

d \* n

number of words

in utterance

Other Ideas for Chatbots

Goal-based bots

Train 2 similar chatbot models with the MLE objective, then let them talk to each other. Give one the goal of getting a specific response from the other bot. If in some number of turns the response is uttered then either one or both bots get a positive feedback. This is basically a reinforcement learning environment, where chatbot models can be further trained without data, just by talking to each other. A somewhat similar setting has been proposed before [28], mainly differing in the fact that the 2 bots invented their own language instead of being pretrained and they had to guess the contents of an image.

Different objective functions

We could use different objective functions that better capture the nature of conversations. Maximum mutual information [3] has been proposed before as well as hand crafted reward functions [24]. These functions have been successfully applied to chatbots to generate somewhat more diverse responses.

Neural architecture search

With all the previously mentioned neural architectures we can craft a lot of different models, but we don’t know which one will perform the best. Neural architecture search [25] has been successfully applied before to create a completely new CNN model. We could inject into this model all the different blocks, modules and techniques that it can use to come up with the best possible architecture.

New and untried architectures

Other architectures that are different from the traditional encoder-decoder model [15,19] have been successfully applied to neural machine translation tasks to achieve state-of-the-art results, but as of yet lack implementations in the conversational agent domain. Since all of the current chatbot architectures emerged thanks to advances in NMT models it’s important to try these out as well. In addition, generative adversarial networks [26] have been applied before to the chatbot domain [27], but this was only one attempt and I think that further ways of implementations should be explored.

[1] Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

[2] Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond

[3] A Diversity-Promoting Objective Function for Neural Conversation Models

[4] A Neural Conversational Model

[5] RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems

[6] How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation

[7] A Persona-Based Neural Conversation Model

[8] Topic Aware Neural Response Generation

[9] Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory

[10] Neural Machine Translation by Jointly Learning to Align and Translate

[11] Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models

[12] Convolutional Sequence to Sequence Learning

[13] Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation

[14] Hierarchical Recurrent Attention Network for Response Generation

[15] Attention Is All You Need

[16] Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models

[17] Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

[18] Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation

[19] Depthwise Separable Convolutions for Neural Machine Translation

[20] Neural Machine Translation in Linear Time

[21] Outrageously Large Neural Networks: The Sparsely-Gates Mixture-of-Experts Layer

[22] Domain Aware Neural Dialog System

[23] <https://www.tensorflow.org/tutorials/seq2seq#bucketing_and_padding>

[24] Deep Reinforcement Learning for Dialogue Generation

[25] Neural Architecture search with Reinforcement Learning

[26] Generative Adversarial Nets

[27] Adversarial Learning for Neural Dialogue Generation

[28] Emergence of Language with Multi-agent Games: Learning to Communicate with Sequences of Symbols