Building Chatbot with Emotions

Honghao WEI
Stanford University
weihh16
weihh16@stanford.edu

Yiwei Zhao
Stanford University
ywzhao
ywzhao@stanford.edu

Junjie Ke
Stanford University
junjiek
junjiek@stanford.edu

Abstract

This project aims at generating dialogues not only appropriate at content level, but also containing specific emotions. To tag emotion categories, we apply sentimental analysis on the dataset and pick up dialogue with strong emotion. We first adopt vanilla sequence-to-sequence neural network and do modification for sentiment level correlation. Secondly we take advantage of deep reinforcement learning and introduce sentiment rewards during learning phase. To better incorporate emotion tag and shorten training time compared to deep reinforcement learning, we make a try for our third method by training an emotional neural network chatting machine. We introduce emotion embedding, internal memory and external memory to pass the information flow and emotion category reasonably to the final output.

1. Introduction

In this paper, we tackle the issue of dialogue generation with different emotions. On current stage, dialogue generation has been widely studied. However, creating response containing specific emotions remains difficult due to its complexity in nature. In our study, we train a recurrent neural network to predict the sentiment category with labeled data from LiveJournal. To generate responses with different sentiment, we adopt the modified Sequence-to-sequence model, deep reinforcement learning, and Emotional Chatting Machine model with static emotion embedding, dynamic internal memory and explicit external memory components to better convey the emotion contents during generation.

2. Problem statement

Problem Given users utterance and emotion, the chat bot is able to generate response correct on word contents and emotion level.

Dataset Text data: Approximately 88 million lines of subtitles of videos and movies in open subtitle database.

Audio data Switchboard and Fisher in Linguistic Data Consortium (LDC)

Evaluation We evaluate the performance for both word contents and emotion level correctness. For word coherence and grammatical correctness, we apply perplexity and BLEU as evaluation. We believe an emotional chatbot should not bore users with limited and repetitive vocabulary and tokens in generated responses. For this reason, we also consider about diversity of contents, in terms of type-token ratio for unigrams and bigrams in generated response, as evaluation for our model. The last but not the least, we compare the emotion tags for generated responses with the expected one and compute the accuracy to measure the performance on emotion level.

3. Dataset

We collected 88 million lines of subtitles in open subtitle database. The subtitles of movies and TV programs provide sufficient conversation contents for training. Nevertheless, subtitles are slightly different from daily conversation due to theatrical and dramatic contents. Sometimes, it may contain swear words. On current occasion, we believe theatrical replies are acceptable when generating replies to negative sentiments, such as anger. For instance, when users growl some swear words, replying 'it sucks' would be a common way to avoid enraging an angry man. Due to the large volume of inappropriate remarks in dataset, and the nature of different sentiment, we decide not to filter the swear words but keep them as one of features in certain emotion category.

When we preprocess the data, we filter long segment of words which is impractical in daily conversation. These long pieces of texts could be monologues in movies. In addition, we only select those conversation, in which the both the question and response convey sufficient sentiment. We use a pretrained model for word vectorization, but would also co-train word vector with sequence-to-sequence model which would confirm the features in vectorization results.

4. Related Works

Sentiment Analysis is one of the most active research areas in natural language processing and has been widely studied in data mining. There are numerous applications on review-related websites, sub-component technology, business and government intelligence across different domains [7]. While there are many practical approaches to analyze sentiment in text, there are two major problems that needs to be tackled in the domain of human-agent integration. First, sentiment detection methods used in human-agent interaction are rare and most of the existing models take advantage of the transcript contents directly and just use the same techniques in text mining. The second problem is how to generate reasonable response after detecting the user's sentiment. This requires sufficient complexity in interaction and is hard to give good performance without hardcoded rules. In our work, we would focus on the second problem to study how to generate responses not only makes sense on token distribution but also on sentiment level.

Intelligent chatbots are of great significance in building successful assistant applications. Given the users input, the chatbot need to analyze the dialog and generate appropriate responses. In recent years, response generation based on sequence-to-sequence (SEQ2SEQ) neural network model is of growing interest. As an example, multi-layer LSTM model [10] have been successfully used in translation tasks. Those SEQ2SEQ models maximize the probability of the response given the previous dialogue context and are able to integrate rich information to produce meaningful response.

A basic SEQ2SEQ model consists of two recurrent neural networks (RNNs): an encoder that processes the input and a decoder that generates the output. Apart from feeding the decoder of the encoded state, attention mechanism was introduced in [1] to allow the decoder have more direct access to the input. Conceptually, this allows the decoder to focus on whats most important in the input at every decoding step.

The objective function for SEQ2SEQ models is maximum-likelihood estimations (MLE). Therefore, these models often ignore the input and produce highly generic responses such as "I dont know what you are talking about" [9, 8, 6] answer to be shortsighted and ignore their influence on future outcomes, which is not good for building a chat bot that can calm users down, and not to mention the performance of vanilla Seq2Seq model on dialogue generation with specific sentiment.

In order to solve the problem of generic responses and let the chat bot have successful long-term dialogues, we try to employ a reinforcement learning framework introduced by Li et al [6]. In this paper, the author does not directly train a model that minimize the loss for each pair of training sample, but simulates dialogues between two virtual agents. Then the model is trained using reward sequences that represent three useful conversational properties: informativity, coherence, and ease of answering.

Furthermore, in order to utilize sentiment information and produce different responses for the different emotion category, the chat bot should be able to learn and capture the emotion representation under different context. This emotion category representation can be directly modeled by using high-level abstraction of emotion expression. Hao et al [11] used embedding emotion categories, together with capturing the change of implicit internal emotion states and explicit emotion expressions with external emotion vocabulary to generate responses that are both contextually and emotionally appropriate.

5. Technical Approach

5.1. Speech and text transition

We use SpeechRecognition in python to perform speech recognition, which supports CMU Sphinx. Meanwhile, we use linux say command to turn text contents into audio voice.

5.2. Sentiment and Sentiment Tag

In this work, we use six sentiments in the Ekmans' atlas of emotions as emotion category[3]. The six emotions include anger, disgust, fear, joy, sadness and surprise. It is claimed that discrete psychophysiological differences found across these emotions. Currently, we use two different methods for emotion detection and tagging.

Text-based Classifier We train a recurrent neural network (RNN) models to predict Ekman's six basic emotions with texts. The input is processed text while the output is the distribution over different sentiment category. For training, we get tweets in LiveJournal with emotion tags. We then perform the multi-category classification on open subtitle data to tag different sentiments. Since the data source is different for training and tagging, there might be decline in classification accuracy on open subtitle datasets.

Adaptive Empirical Mode Decomposition when a person is under extreme sentiment, micro-muscle tremors (MMT) occur in the muscles that make up the vocal tract which are transmitted through the speech. When users are emotional, this MMT shifts in frequency. Extreme sentiment can thus be detected by analyzing the change in MMT frequency of an individuals voice. We use adaptive empirical mode decomposition methods to analyze users voice with content of transcripts to determine the polarity and category.

5.3. Sequence-to-Sequence Model (Seq2Seq)

For chatbot response generation, we use a multi-layer sequence-to-sequence encoder-decoder model. The most common approach is to use an encoding recurrent

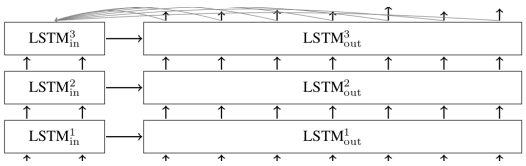

Figure 1: Architecture for Seq2Seq

neural network (RNN) reads the source sentence $x = (x_1, \cdots, x_T)$ and encodes it into a sequence of hidden states $h = (h_1, \cdots, h_T)$ where $h_t = f(x_t, h_{t-1})$. Then, another decoding RNN generates a corresponding response $y = (y_1, \cdots, y_{T'})$ based on the encoded sequence of hidden states h. When generating y_t the model calculates the probability using:

$$p(y_t|y_{<t}, x) \propto \exp\{q(y_{t-1}, z_t, c_t)\}$$

where

$$z_t = g(y_{t-1}, z_{t-1}, c_t)$$
$$c_t = r(z_{t-1}, h_1, \cdots, h_T)$$
$$y_{<t} = (y_1, \cdots, y_{t-1})$$

The model is trained to maximize the conditional log-probability of the appropriate response given a user input with respect to the model parameters θ:

$$\theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(y_t^n|y_{<t}^n, x^n)$$

(x_n, y_n) is the n-th training pair of sentences, and T_n is n-th target sentence length.

We use the attention mechanism that was introduced in [2], which allows the decoder to have more direct access to the input while decoding. At each time step, the decoder would compute the context vector c_t as a weighted sum of the hidden states with the coefficients $(\alpha_1, \cdots \alpha_T)$ defined as:

$$\alpha_t = \frac{\exp\{a(h_t, z_{t-1})\}}{\sum_k \exp\{a(h_k, z_{t-1})\}}$$

where a is a single-layer feedforward neural network. z_t is a new hidden state of the decoder which depends on the previous hidden state z_{t-1}, word generated in previous time step y_{t-1} and the context vector c_t.

Fig 1 shows our network framework. We first encode all the input words using pre-trained GLOVE embeddings. Then we use three layers of LSTM cells with attention mechanism in the decoder. The model will jointly learns the attention alignments and response generation.

Due to the large size of the output vocabulary, we use sampled softmax loss as described in [4]. At each update, a small subset V' of the vocabulary will be used to approximate the expected gradient.

To speed up the training process, we also makes use of bucketing. We pre-define a set of buckets that takes care of sentences of different lengths. The sentences are padded to the nearest length. In this way, we won't need to pad all the sequence to the longest sentence length. This allows us to efficiently handle sentences of different lengths.

To modify Seq2Seq model for sentiment level correctness, we introduce additional cross-entropy in the initial optimization phase. To avoid repetition, please refer to the section for loss function.

5.4. deep reinforcement learning methods

For the next step, we are planing to use the method descried by Li et al.[6]. We plan to apply the reinforcement learning framework to the training and use two chat bots chat with other. Just like what have used in this paper, we plan to use the same definition of state and action. The state would be $[p_i, q_i]$, which is previous two dialogue turns. The action a would be the dialogue utterance to generate. The policy $p_{RL}(p_i + 1|p_i, q_i)$ would be a network with the same structure of the SEQ2SEQ model we descried above, which is a LSTM encoder-decoder with attention mechanism. Then we plan to use the following rewards to train the network:

The first loss is the Ease of Answering loss, which is helped to erase the dull response. It is defined as $r_1 = -\frac{1}{N_S}\sum_{s\in S}\frac{1}{N_s}\log p_{seq2seq}(s|a)$, where the $p_{seq2seq}(s|a)$ is the LSTM we trained end to end directly on the data pairs. The S is set of dull response and N_S is the size of S and N_s is tokens number of the responses.

The second loss is the Information Flow, which is helped to encourage the chat bot contribute new information to the dialogue. It is defined as $r_2 = \log\cos(hp, hp)$, where h_{p_i} and $h_{p_{i+1}}$ are representations obtained from the encoder for p_i and p_{i+1}

The third loss is Sentiment Coherence to make the generated sentence coherent not only grammatically but also emotionally . We use a pre-trained RNN model to predict the polarity and category each sentence conveys. We want the emotion generated responses contains similar to the current given emotion settings and the emotion contained in another bot's question. If the question is significantly joyful, a sad remark in response is considered inappropriate.

5.5. Emotional Chatting Machine

To let the chatbot learn to express emotion, we implement an emotion chatting machine (ECM) [11]. The basic structure of ECM is very similar with an sequence to sequence model. The architecture consists of Recurrent neural network of GRU cells with attention mechanism. But it contains three different mechanisms are used for generating response with specific emotion.

1.**Emotion Category Embedding:** We represent different emotion category as an embedded vector. This vector is feed to the decoder and will be learned during training. Note this emotion embed remains static and would not change in the flow.

2.**Internal Memory:** In addition to traditional Seq2Seq network, we integrate a dynamic Internal Memory to capture the emotion dynamics during decoding. An internal state for each emotion category exists before the decoding process starts. From start to end, the emotion state decays to zero which indicates the emotion is completely expressed.

3.**External Memory:** External memory serves to model emotion expressions explicitly. Different generation probabilities are assigned to emotion words and generic words. There is no intersection of vocabulary for emotion and generic words, and we balance the word distribution with learned score.

The overall architecture of the network can be shown in Fig 2, the above part of the architecture is a regular end to end sequence model, where the GRU is been used to be the basic cell and based on that we build our decoder using Recurrent Neural Network. We build our encoder using bi-directional LSTM and concatenate the cell state and hidden state to be the input of decoder. The attention mechanism is also integrated to our system but it is a little different from the regular design since we combine emotion in it, which will be explained in more details. Besides the regular encoder-decoder network, three special network structures, i.e. Emotion Category Embedding, Internal Memory and External Memory are integrated with the decoder to generate emotional responses.

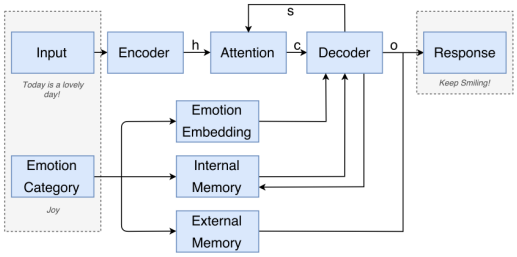

Figure 2: Overall Architecture for ECM

5.6. Emotion Category Embedding

Since emotion category is an very high-level abstraction of an expression of the emotion, like the words embedding, the most intuitive approach to model it is to take the emotion category as additional input. So if the e represent the emotion category, then we embed it to a vector v_e as an input of our system. v_e is a trainable value and then learn the vectors of emotion category through training. With the embedding of emotions, the cell of decoder becomes:

$$s_t = GRU(s_{t1}, [c_t; e(y_{t1}); v_e])$$

5.7. Internal Memory

The emotion category embedding method is a static method, since for each decoder process it will not change. However, in real life, the level of sentence in different part of sentence would change. It is hard to imagine users express emotion in a sentence from beginning to end. The relevant pattern would be yelling or screaming, which is rather unusual in daily chatting settings. Moreover, inspired by the psychological findings, emotional responses are relatively short lived. People tend to express their emotion completely at the end of sentence. To address these two problems, an internal memory module is introduced to decoding process to capture the emotion dynamics during decoding: we supposed that there is an internal emotion state before the decoding process starts and then at each time step it will decade by a factor. How much it decade will be controlled by a gate which is a output of GRU cell. The emotion left at the end of the decoding will cost a loss since it shows that the decoder fails to generate a response using up the emotion. The process can be shown in Fig 3.

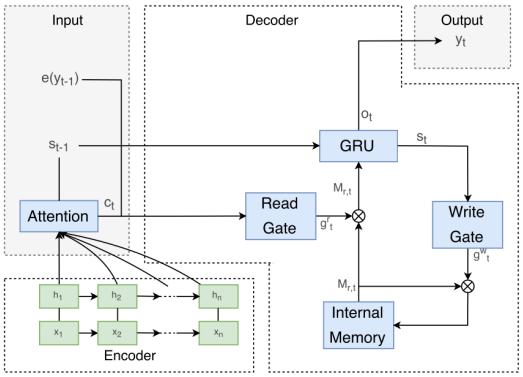

Figure 3: Architecture for Internal Memory

For each time step, the state of GRU of last time step will be feed to the attention and then scan the input to get the info from the encoder. Then concatenate with the sample of the output of last time and Emotion Category Embedding, we can use a sigmoid to know how much we should use(read from) the emotion and also how much we should write the emotion back:

$$g_t^r = sigmoid(W_g^r[e(y_{t1}); c_t])$$
$$g_t^w = sigmoid(W_g^w s_t)$$

So after each time step, the internal memory we used and left will become:

$$M^I_{r,t} = g^r_t \odot M^I_{e,t}$$
$$M^I_{r,t+1} = g^w_t \odot M^I_{e,t}$$

And then the state and output of GRU is updated by:

$$s_t = GRU(s_{t1}, [c_t; e(y_{t1}); M^I_{r,t}])$$

5.8. External Memory

When using the internal memory module, the emotional expression is implicit, making it not directly observable. In order to make the decoder pick up words with strong emotion which carry strong emotions compared to non-emotion words, here we use an external memory model to let the network choose at what probabilities should they choose emotion words or generic words. The overall design of the External Memory can be shown as:

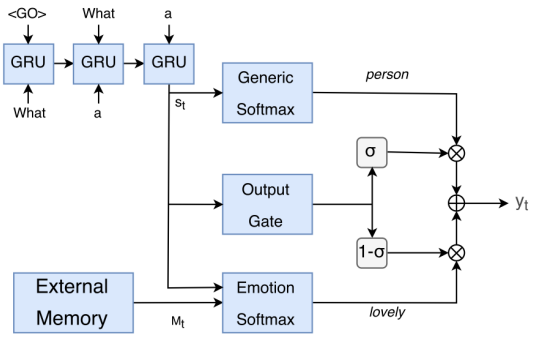

Figure 4: Architecture for External Memory

When the cell of decoder output, we use a vector dot followed with a sigmoid function to generate a scale value shows how we care about emotional words for this position of sentence. The probability we pick up the emotional word is:

$$g^o_t = sigmoid(v^T_u s_t)$$

The probability for each words are:

$$P(y_t) = \begin{bmatrix} (1 - g^o_t) softmax(W^o_g s_t) \\ g^o_t softmax(W^o_e s_t) \end{bmatrix}$$

where the upper part of the vector is non-emotional words and the lower part is the emotional words. We sort the vocabulary and divide them into two category in advance for acceleration.

5.9. Loss Function

The original loss function for Seq2Seq model is the cross entropy of token distribution. In notation of loss function, o_t is the predicted token distribution and p_t is the golden distribution. In this work, we introduce two more regularization parts than word level correctness. The first term is to supervise whether we choose an emotional word or generic one so that we could generate responses emotionally corrected. q_t is a scalar value either to be 0 or 1, indicating at current timestamps, should it be an emotional word or not. Similarly, g^o_t is the probability of choosing a generic or emotional word at current timestamps. The second term is l2 norm of internal memory. We expect it decline to zero at the end of the sentence so that bot expresses its feelings completely. Note that we introduce additional weight for regularization since their raw value could be very different from each other.

$$L(\theta) = -\frac{1}{m}\sum_{t=1}^{m} p_t \log o_t - a\frac{1}{m}\sum_{t=1}^{m} q_t \log g^o_t + b||IM||_2$$

For modified Seq2Seq model, we have different regularization terms in loss function. The only term in regularization refers to the cross-entropy on emotional level. q_t is a static label indicate which emotion category it falls in while s_t represents the probability of getting into each sentiment class ad current timestamps. We also introduce a regularization weight for the same reason.

$$L(\theta) = -\frac{1}{m}\sum_{t=1}^{m} p_t \log o_t - a\frac{1}{m}\sum_{t=1}^{m} q_t \log s_t$$

5.10. Techniques for Uncertainty

It is very likely that our model would encounter unexpected situations or on some occasion, the highest correlated replies might be of low confidence. In this way, the generated utterance might not be suitable for reply. It is devastating for chatbot, since once we fail to intrigue users for a single round of dialog, users is very likely to easily lose interests in extending the conversation with a bot. In this way, it is important to generate general but appropriate replies to certain kind of sentiments. Here, we take two approaches for compensation.

utterance-utterance collaborative filtering In this way, we analyze the contents of users' utterance and generate similar sentence in the same time. We would select the top correlated sentences and search them directly in our database for possible answers. We would also use open API for Question and Answer platforms and web discussion website such as Quora and Reddit for higher quality searching. In this way, we do not consider emotion as inputs.

Decision Tree for general query We would deploy a decision tree for different sentiment and contents in utterance. In this way, in the worst case, we could reply users with generally related contents. Besides utterance, relevant images, music, videos and articles are possible ways of reply as well. In this way, we do consider emotion as inputs.

Human utterance	Reply generated
Damn it	i was hoping you weren t going to ask
No problem	i love you
You bastard	i m always fucking smart
I ' m going to break up	he said you were a great girl

Table 1: Conversation of human-agent interaction generated by Seq2Seq model

6. Experiments results

In this section, we would discuss the experiment results for Seq2Seq model, static Emotional Chatting Machine and Dynamic Emotional Chatting Machine. We have not trained the Deep Reinforcement learning model well by deadline, but we would submit relevant results as soon as possible when we finish testing. We consulted with Jiwei and he said it should take 3-4 weeks and the algorithm is still running on the server.

6.1. Seq2Seq model

The baseline model of our project is a Seq2Seq model trained on 100 thousand dialogues. The vanilla methods is simple but effective.Some of the generated replies are listed in Table 1. We could see that the conceived results are either positive, trying to alleviate the blue mood, such as 'I love you' or articulate reasonably to interest users for further discussion, such as 'He said you were a great girl'. When users' utterance involves swear words, our model may even responds with swear words which makes the communication more natural.

However, there are some problems with Seq2Seq model. First of all, the performance over diversity is poor. For example, when you start a statement beginning with 'I am', the model would respond with 'you are ' or 'you be ', which is not so common in real life.Similarly, whatever users say with one word, such as 'Tired' and 'Unhappy', our model would reply uniformly with 'Clever'. Another question is that, the dialog would always converges to repetitive nonsense sentences. In our case, after 1-2 rounds, it converges to 'I will reward you a bag of rice', which makes non sense. Inevitably, the length of dialogue is limited, for most conversation it is less than 2.

During parameter tuning, when we increase the number of layer in LSTM, the model produces results with grammar mistakes. When we increase the number of layer to 5, the Seq2Seq model fails to conceive replies given whatever inputs. We think it might be the problem of gradient vanishing, with which the complexity of the model undermine the overall performance. We also notice dropout would contribute to diversity of the response.

The biggest problem of Seq2Seq model is that we could not balance the accuracy on both word level and emotion level. Currently, we only use the sum of word distribution cross-entropy and sentiment cross-entropy as loss for optimization. If there is no weight, the sentiment portion would be ignored in the optimization process and it results in same generated responses with different emotion settings. However, if we increase the weight of sentiment distribution cross-entropy, it undermine the coherence of each words in a sentence and results in generated responses disobeying basic grammar rules. For instance, to reply 'Damn it' in 'Angry' emotion setting, introducing large weight on sentiment part causes replies such as 'i i you you fucking', which is emotional but not grammatical. One of the bi-products of the unbalance between word accuracy and emotion accuracy in Seq2Seq model is that, when we prefer sentiment to sentiment coherency, the perplexity of model would be rather high.

6.2. Emotional Chatting Machine

In this section, we would mainly compare the performance of static Emotional Chatting Machine and Dynamic Emotional Chatting Machine. We divide the evaluation into two parts. The first part is to test whether the content is grammatical and relevant. The second part is whether generated responses is emotional corrected.

6.2.1 Word level evaluation

We first adopt perplexity to evaluate the model at the content level. The results are shown in Table 4. As we discussed in last subsection, Seq2Seq is capable of reaching relatively low perplexity while ignoring correctness on sentiment level. Here, we present the results of introducing large weight on sentiment level accuracy and it is comparatively worse than the Emotional Chatting Machine models. It is shown that ECM presents a better combination of word level and sentiment level interests in dialogue generation compared to vanilla Seq2Seq model. In addition, we notice dynamic ECM achieves lower perplexity than static ones. One of possible explanations is that, we decline the l2 norm of internal memory to zero in dynamic model and thus should introduce less influence on the original dialogue generation process while the emotion vector remains static.

Moreover, we present the BLEU results for correlation with human judgments. The results are shown in Table 5. Though dynamic model achieves slightly better BLEU than the static one, the overall performance of Emotional Chatting Machines are less satisfactory. As argued in [5], BLEU is not suitable for measuring conversation generation. Since the open-subtitle datasets consists lines in videos and movies in different topics and contents, we believe it is reasonable the BLEU results for ECM is relatively low over

Input	Response(Original)	Model	Emotion	Response (Generated)
You bastard	I've always been a good boy	Seq2Seq	-	i m always fucking smart
		Dynamic Emotional Chatting Machine	Joy	i have i love and my old love
			Surprise	okay okay
			Sadness	not i could happy
			Fear	i don t care
			Anger	i m not fucking smart
			Disgust	i have not show good ashamed because you this you

Table 2: Responses generated with different sentiment

	static ECM	dynamic ECM	True Labels
Unigram	0.0138	0.0076	0.0331
Bigram	0.0897	0.06377	0.1277

Table 3: Unigram and bigram diversity comparison between Emotional Chatting Machines

	Seq2Seq model	static ECM	dynamic ECM
Perplexity	301.9	67.8	63.5

Table 4: Perplexity comparison between Seq2Seq model and Emotional Chatting Machines

	static ECM	dynamic ECM
BLEU	0.40	0.42

Table 5: BLEU comparison between Emotional Chatting Machines

this evaluation metric.

In order to avoid the situation, in which generated responses are similar on word level with different inputs and undermine users' interests of interaction, we compute the unigram and bigram diversity to report the degree of diversity. We use type-token ratio for unigrams and bigrams defined in [5], the number of distinct unigrams and bigrams in generated responses scaled by the total number of tokens. We find that static ECM produces more diverse outputs than the dynamic one.

6.2.2 Sentiment level evaluation

To evaluate the ECM models in sentiment level, we use emotion accuracy, defined as the number of responses with correct sentiment category scaled by total number of items in testing set. It measures the agreement between the expected emotion category and the predicted ones. We use the same classifier to tag emotion category as the one in preprocess phase. Table 6 reports the emotion accuracy of ECM models. We notice the dynamic models achieves almost twice as much as static model. It indicates that internal memory is comparatively better for sentiment representation than emotion embedding. The overall emotion accuracy is below 0.5. One of the reasons is related to multi-category classification problem, in which there is 6 different categories in the Ekmans' Atlas of emotions. Moreover, the emotion distribution of our dataset is not balanced. For instance, nearly one in third of samples are labeled as 'Angry'. Therefore, the trained ECM model with angry emotion is comparatively better than the others, such as Sadness and Disgust.

	static ECM	dynamic ECM
Emotion Acccracy	0.2038	0.4596

Table 6: Emotion accuracy comparison between Emotional Chatting Machines

6.3. Case Study

In this part, we report case study of responses generated given different models and sentiment. In Table2, we report the results of Seq2Seq model and dynamic ECM with input 'You bastard'. The generated results are significantly different with each emotion category. The dynamic ECM even generates opposite results compared to the Seq2Seq model with no emotion input. Moreover, the dynamic ECM presents each sentiment sufficiently, such as 'Not happy' in sadness settings.

We also report the performance of two best trained models (dynamic ECM with angry or joy sentiment category) in Table 7. Angry and joy are the top 2 labeled sentiment In our dataset. We input same utterance 'do you like me' to start the conversation. Note that these two bots replies differently since the beginning and give different replies when users offer same proposal at the end. Angry bot tends to be bad-tempered and refuse user' s invitation while the joy bot is more outgoing and accepts user's invitation.

Angry	Joy
Human: do you like me	Human: do you like me
ECM: i know a little	ECM: i love my friend
Human: i love you	Human: he likes me
ECM: thanks	ECM: oh that s better it
Human: shall we date	Human: let's date
ECM: good night _UNK	ECM: yes

Table 7: Comparison of dynamic ECM with angry and joy emotion

7. Conclusion and feature work

In this paper, we test three mechanisms for generating responses with specific sentiment. Due to time constraints, we have not finished training phase for Deep Reinforcement Learning model. We compare the performance of vanilla Seq2Seq model and both static and dynamic Emotion Chatting Machine. Automatica evaluation shows that the emotion embedding, internal memory and external memory in ECM help to generated reasonable responses on both content and emotion level.

In the future work, we would continue the training phase of Deep Reinforcement Learning and report the results as soon as we can. In addition, we notice the emotion tag for dataset has influential effects on the performance. We would collect more samples on live journal and train better multi-category classifiers with different neural network architecture for sake of better emotion tag results.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] C. Darwin. *The expression of the emotions in man and animals*. Oxford University Press, USA, 1998.

[4] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007, 2014.

[5] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.

[6] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky. Deep reinforcement learning for dialogue generation. *CoRR*, abs/1606.01541, 2016.

[7] B. Pang, L. Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[8] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015.

[9] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. *CoRR*, abs/1506.06714, 2015.

[10] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[11] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*, 2017.