A reward-centric reinforcement learning approach for dialogue generation

Luong Pham Bao - 19521242
University of Information Technology

Nguyen Gia Thong - 19520993
University of Information Technology

Pham Ngoc Duong - 19521412
University of Information Technology

Abstract

By utilizing deep learning to simulate future reward in chat-bot dialog, the study *Deep Reinforcement Learning for Dialogue Generation* (Li et al., 2016) aims to combine the strength of neural dialogue generation models and reinforcement learning to develop a neural conversational model based on the long-term success of conversations. Neural models of dialogue generation from recent research have shown to be excellent at generating replies for conversational agents, however they primarily focus on question-answer response instances and do not consider the evolution of the discussion, resulting in generic questions that lead to dead ends. For clear and interesting dialogues, predicting the future direction of a conversation is critical. The proposed RL model simulates two virtual agents conversing and uses policy gradient approaches to reward sequences with three conversation properties: informativeness, coherence, and simplicity of replying in regard to the forward looking function. The suggested model was evaluated based on produced sequence length, variety, and a human review procedure, and it was discovered that the proposed algorithm produces interactive responses and manages to promote a more prolonged discussion in dialogue simulation. Our team look to examine and replicate the research and findings of this study.

1 Introduction

Dialogue Generation is a growing field of research, where state-of-the-art methods still leave a lot to be desired and actively researched as a core problem in NLP. Neural response generation is an approach that has shown many potentials (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016a; Wen et al., 2015; Yao et al., 2015; Luan et al., 2016; Xu et al., 2016; Wen et al., 2016; Li et al., 2016b; Su et al., 2016). The sequence-to-sequence (SEQ2SEQ) LSTM model (Sutskever et al., 2014) is a neural generation model focus on maximizing the probability of generating a response based on previous dialogue turns. This allows for context to be taken into account in consecutive dialogue turns (Sordoni et al., 2015), something machine translation-based dialogue models cannot achieve (Ritter et al., 2011).

Nonetheless, the SEQ2SEQ neural generation model also has two pronounced drawbacks. The SEQ2SEQ models are trained using the maximum-likelihood estimation (MLE) objective function in order to predict the next dialogue turn's response with a given conversational context. Yet, the MLE approximation in objectives may not be fully in line with the desirable goal of a chatbot-to-human conversation: provide diverse, informative, and interesting feedback to keep the conversation engaging. An example of this gap in objective alignment is in the use of highly generic responses like "I don't know" regardless of input, shutting the conversation down (Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a). The high frequency of generic responses in the SEQ2SEQ's training set and generic response versatility in a wide range of contexts are factors that contribute to this behavior. However, prioritizing generic responses alludes to the model itself not weighting engagement in its assessment process.

The second drawback of the SEQ2SEQ model is it frequent fall into an infinite loop of repetitive responses, due to the inability to account for repetition. In one example, the dialogue is trapped in an infinite loop after three turns, with both agents generating filler responses such as "i don't know what you are talking about" and "you don't know what you are saying". This was a result of an agent generating a generic utterance in respond to a reconfirm response "16?" to a statement "I'm 16.". That was a bad action to take, allowing for no alternative to continue the conversation.

To address and improve upon these shortcomings, we need a conversation framework that can (1) incorporate developer-defined rewards that more closely resemble the real aim of chatbot development and (2) reflect the long-term impact of a generated response in an ongoing dialogue.

A neural reinforcement learning (RL) generation technique was decided for our model to achieve these aims, as it has been a widespread practice in MDP and POMDP conversation systems. The approach, which is based on the encoder-decoder architecture, simulates a dialogue between two virtual agents in order to explore the domain of possible actions while learning to maximize the expected reward. Good dialogues are forward-looking (Allwood et al., 1992), interactive, informative, and coherent, and simple heuristic approximations is used to rewards that characterize them. Rather than using the MLE goal stated in typical SEQ2SEQ models, the agent learns a policy by maximizing the long-term developer-defined reward from ongoing conversation simulations using policy gradient techniques (Williams, 1992).

Employ only a simple heuristic reward structure, we has gained an impressive set of results. Thank to its simple design, the heuristic structure helps reduce the total training time of our models, but still allow agents to be trained with a diverse amount of interaction and exploration, as well as improving their ability to generate sequences without relying too heavily on conversation-specific information. Our heuristic approach also outperforms a hierarchical reinforcement learning method equivalent, which allows for easy and rapid reward optimization with topic-specific in mind at the expense of less flexible model configurations. This heuristic approach is, in our opinion, a good starting point that allows for further optimizations.

2 Related Works

The transfer of information from the prior phrase to the current sentence is the initial step in conversation creation, followed by learning how to translate rules from the input message to the response created using the training dataset, which is usually large for open domains). Ritter et al. (2011) classified the dialogue generation problem from statistics as a machine translation problem and developed a set of solution templates. Sordoni et al. (2015) extended on Ritter et al. system's by instead, initializing with a neural model that included previous context based on the result of a dialogue based on SMT phrasal service. Vinyals and Le, 2015 inspired recent development in SEQ2SEQ models by experimenting with mapping encoded messages to a distribution probability vector indicating their semantics and generating a response from the message vector. Serban et al. (2016) proposed a hierarchical neural model for retaining dependencies over long conversation histories. To minimize the rate of generic replies generated by the SEQ2SEQ system, Li et al. (2016a) propose utilizing a mutual information model connecting the message and response as an alternative objective function.

Another area of statistical study is the creation of a problem-solving dialog generating system that performs a certain function. Statistical models such as Markov Decision Processes (MDPs)(Levin et al., 1997; Levin et al., 2000; Walker et al., 2003; Pieraccini et al., 2009), POMDP (Young et al., 2010; Young et al., 2013; Gasic et al., 2013a; Gasic et al., 2014) models, and models that statistically learn generation rules (Oh and Rudnicky, 2000; Ratnaparkhi, 2002; Banchs and Li, 2012; Nio et al.,2014) are among the efforts. This dialogue literature uses reinforcement learning to train conversation policies (Walker, 2000; Schatzmann et al., 2006; Gasic et al., 2013b; Singh et al., 1999; Singh et al., 2000; Singh et al., 2002). These task-specific dialogue systems will rely on either limited number of carefully considered parameters, or manually created samples of states, action and reward signals for each new domain, making the model difficult to extend to the open domain for unexpected situations.

Prior work on reinforcement learning for language understanding is also important, such as learning from delayed reward signals by playing text-based games (Narasimhan et al., 2015; He et al., 2016), executing instructions for Windows assistance (Branavan et al., 2011), or comprehending dialogues that provide navigation directions (Vogel and Jurafsky, 2010).

Our objective is to combine SEQ2SEQ with reinforcement learning models, leveraging the benefits of both. We thus draw on work by Wen et al. (2016) and proceed by training an end-to-end system of task-based conversation and association in the database between input representations and position value pairs, or Su et al. (2016), which demonstrated that reinforcement learning enhances conversational performance by combining reinforcement learning with neural generation for real-world user tasks.

3 Recurrent Neural Networks

3.1 Introduction

In a language model, the probability $p(w_t|w_{t-1},\dots,w_1)$ can be calculate by using Markov N-gram model:

$$p(w_t|w_{t-1},\dots,w_1) \approx p(w_t|w_{t-1},\dots,w_{t-N}))$$

this model is yet to accurately represent the probability that a word will occur after a certain word if they are too far away from each other. To address this, we need a model like RNN which can calculate $p(w_t|w_{t-1},\dots,w_1)$. RNN is simply a neural network which has a looping process that takes output of the previous neuron as input of the next neuron. In other words, RNN is capable of remembering previously calculated information. Unlike traditional neural network models which have its input completely independent of output.

RNN takes input as a sequence $(x_1,\dots,x_T)$ which is the output of word embedding model in order to build an output sequence $(y_1,\dots,y_T)$ by using equations defined as follow:

$$h_t = sigm(w^{hx}x_t + w^{hh}h_{t-1})$$
$$y_t = w^{yh}h_t$$

Where:

- x_t: input at time t.
- h_t: hidden state at time t.
- y_t: output at time t.

We use Back-Propagation Through Time (BPTT) to accumulate the gradient of each step in time to avoid gradient converges to 0 (vanishing problem) but at the same time we need to ensure that gradient also does not diverge (exploding problem). To achieve this, we will use Long Short Term Memory.

3.2 Long Short Term Memory

Long Short Term Memory (LSTM) is a RNN architecture which is use to solve long-term dependencies problem. This model is proposed by Hochreiter & Schmidhuber (1997), revisited and improved recently by Ayako Mikami (2016). LSTM's main goal is overcome the vanishing/exploding gradient problem by control the information flow at each RNN's neuron. LSTM use three gates (input, output, forget), responsible for determining what information will be saved or discarded, generate new candidate and decide the output result.

4 Gated Recurrent Units

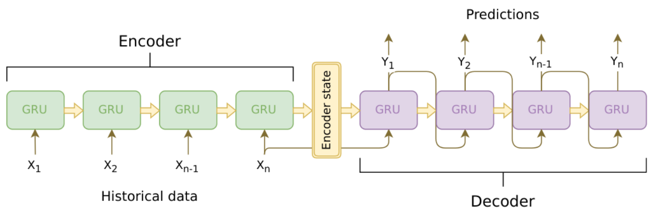
Introduced by Cho, et al. in 2014, Gated Recurrent Unit (GRU) is a gating mechanism in RNNs. In other words, GRU is the improved version of standard RNNs, similar to LSTM in terms of design, in fact, GRU can be considered as a variation of LSTM, they both aim to solve the vanishing/exploding gradient problem and can produce the same excellent results in some cases. GRU uses two gates (update, reset) represented as two vectors, they control the information flow, allowing what should be saved then pass to the output or discard redundancies.

5 Sequence to Sequence Model

5.1 Introduction

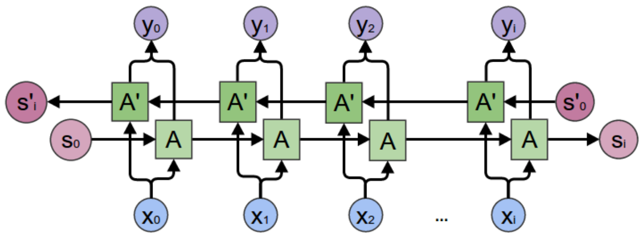
First introduced by Google in 2014, Sequence to sequence (SEQ2SEQ) models is a special class of RNN which is widely use to solve complex language problems and has many applications apply on human daily tasks and being use in variety of large systems/devices like text translation, voice-enabled devices, chatbots, AI assistants,... The model takes a sequence of tokens as input, transform them to word embeddings and fixed dimensional vectors, then run these word embeddings to an LSTM encoder model to produce a vector respresentation of the input, the vector then pass to an LSTM decoder whose job is to predict the response sequence token by token. In other words, this model aims to generate responses according to each input utterances with natural language and the length of input may different from output (e.g translate sentences from a language to other languages).

5.2 RNN-Encoder-Decoder



The RNN-encoder-decoder model (Cho et al., 2014a and Sutskever et al., 2014) is a recurrent neural network architecture that aimed to process a variable length input sequence as a string of input tokens and predict the corresponding output sequence, also as string of tokens. The model is trained to maximize the cross entropy of the true sequence corresponds to the context of that sequence during training, therefore learning can be done through back-propagation, similar to a feedforward neural network. We can only provide the predicted output from the model as input for the next instance, as the actual output sequence is not available to it. This is a "greedy" approach, for the predicted output is not guaranteed to be precise, and the interpretation of the previous input's dialogue context depends on the current output. An alternative approach for this is to apply Beam search and to build a probabilistic model of the previous possible output for future steps as sequence of predictions will be chosen based on its probability. We do, however, continue to use greedy search for its ease of implementation. Encoder-Decoder SEQ2SEQ model consists of two main components, both of them can be LSTM or GRU models and connected by a context vector.

5.2.1 Encoder

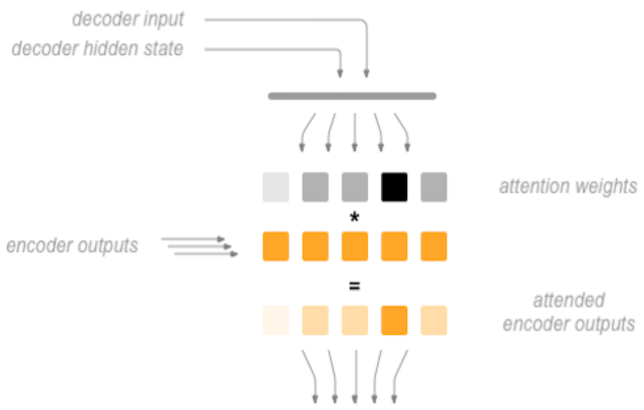


The encoder RNN processes input sentence token sequentially, producing an output vector and a hidden state vector at each iteration. The encoder converts each point in the sequence's context into a set of points in a high-dimensional space. The output vector is recorded while the hidden state vector is passed to the next time step. A multi-layered Gated Recurrent Unit (Cho et al., 2014a) is the basis for our encoder. A bidirectional form of the GRU is used here, where there will be two separate RNNs: one RNN receives the input sequence in normal sequential order, while the other receives it in reverse order. At each time step, the outputs of each network are added. We can encode both past and future contexts by using a bidirectional GRU.

5.2.2 Context Vector

Context vector, the final hidden layer of the encoder RNN is a vector space representation, containing semantic information about the query sentence that is input to the model as an encoder takes the model's input sequence as input and encodes it into a fixed-size context vector, and a decoder utilizes the context vector from above as a seed to create an output sequence.

5.2.3 Decoder



The response sentence is created per-token basis by the decoder RNN. It generates the next word in the sequence using context vectors and internal hidden state generated by the encoder. This process continues until an EOS_token is outputted, meaning end of sentence. Our implementation of the decoder RNN differ and improve upon from the original SEQ2SEQ decoder in one notable aspect. In vanilla SEQ2SEQ, it is likely to experience information loss due to the decoder using the entire fixed context of an input sequence at every step, thus would limit greatly the decoder processing capability when handling with long input sentences. Here, we rectify this limit in processing ability by allow the decoder to pay attention to only certain parts of the input sequence, as the basis of an attention mechanism (Bahdanau et al ., 2015). Our specific implementation here is with global attention, an improved attention mechanism by Luong et al ., 2015.
With attention mechanism, the current hidden state of the decoder and the encoder's outputs are used to calculate attention. Because the output attention weights have the same structure as the input sequence, we can multiply them by the encoder outputs to get a weighted sum that specifies which sections of the encoder output to focus on. In contrast to Bahdanau et al's own local attention technique, global attention examine all the encoder's hidden states, not just from the current time step. Global attention also only requires the hidden state of the decoder from the current time step to calculate attention weights, rather than needing the decoder's state from previous time steps. It also provides score functions to determine the attention energy between the encoder and decoder outputs.

5.2.4 Greedy decoding

Greedy decoding is the decoding method that we use during training when we are NOT using teacher forcing. The input sentence is evaluated using the following computational graph:

1. Forward input through encoder model.
2. Prepare encoder's final hidden layer to be first hidden input to the decoder.

3. Initialize decoder's first input as *SOS* token.
4. Initialize tensors to append decoded words to.
5. Iteratively decode one word token at a time:
6. Forward pass through decoder.
7. Obtain most likely word token and its softmax score.
8. Record token and score.
9. Prepare current token to be next decoder input.
10. Return collections of word tokens and scores.

6 Reinforcement Learning for Open-Domain Dialogue

6.1 Model Components

The model consists of two agents take turns talking to each other, with p and q represents sentences generated by the first and second agent respectively. A dialogue is defined as a sequence of alternating sentences between the two agents. Generated sentences can be treated as actions that are taken according to a policy defined by an encoder-decoder RNN language model.

By using policy search, we are optimizing the parameters of the network in order to maximize the expected future reward which we will describe later in section 9.

6.2 Action

An action a is the generated dialogue utterance. The action space is infinite because of course we can control the length of generated sequences.

6.3 State

A state is represented by the previous two dialogue sentences $[p_i, q_i]$. For further process, the dialogue history will be transformed to a vector by concatenate p_i and q_i then feed them to an LSTM encoder model.

6.4 Policy

A policy has a form based on an LSTM encoder-decoder and is defined by its parameters. In this scenario, we use a stochastic policy because if we use a deterministic policy, the results will likely end up being a discontinuous object which is difficult to optimize using gradient-based methods.

6.5 Reward

When the agent reaches the end of each sentences, we observe the reward r which is obtained for each action. The final reward for action a is the combination of three reward functions:

$$r\left(a, [p_i, q_i]\right) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 \tag{1}$$

Where:

- $\lambda_1, \lambda_2, \lambda_3$: reward distributions.
- r_1, r_2, r_3: results of those following reward functions.

6.5.1 Ease of Answering

We measure the ease of answering a generated sentence by using the negative log likelihood of responding to that sentence with a dull response. We notice that these dull responses frequently occur in SEQ2SEQ models of conversations so we manually constructed a list of dull responses $\mathbb{S}$, this list consists of 8 sentences such as "I don't know what you are talking about", " You don't know what you are saying", etc.. There are many other kinds of dull responses that can be constructed but those responses will likely fall into same regions in the vector space computed by the model. A system less likely to generate sentences in the list is therefore less likely to generate other dull responses. The reward r_1 is given by:

$$r_1 = -\frac{1}{N_{\mathbb{S}}} \sum_{s \in \mathbb{S}} \frac{1}{N_s} \log\left(p_{seq2seq}\left(s|a\right)\right) \tag{2}$$

Where:

- $N_{\mathbb{S}}$: number of dull responses s of $N_{\mathbb{S}}$.
- N_s: number of tokens in the dull response s.
- $p_{seq2seq}$: the likelihood output returned by SEQ2SEQ models.

Note that $p_{seq2seq}$ is learned based on the MLE (Maximum Likelihood Estimation) objective of the SEQ2SEQ model while $p_{RL}(p_i|q_i)$ is the stochastic policy function optimized for long-term future reward in RL setting. r_1 is further scaled by the length of target $\mathbb{S}$.

6.5.2 Information Flow

To have agents introducing new information at each turn to avoid repetitive and keep the conversation continue, we decrease the reward by define a function that penalize semantic similarity from an agent's sentences in consecutive turns. The reward r_2 is the negative log of cosine similarity between them:

$$r_2 = -log(\cos\left(h_{p_i}, h_{p_{i+1}}\right)) = -\log \frac{h_{p_i}.h_{p_{i+1}}}{\left\|h_{p_i}\right\| \left\|h_{p_{i+1}}\right\|} \tag{3}$$

Where:

- $h_{p_i}, h_{p_{i+1}}$: representations value obtained from the encoder for two consecutive turns p_i and p_{i+1}.

6.5.3 Semantic Coherence

We also need to calculate the completeness of the answer to avoid cases that the generated responses have high reward but are ungrammatical or incoherent. To solve these cases, we measure the mutual information between action a and sentences of previous turns in the dialogue history. The reward r_3 is defined as follow:

$$r_3 = \frac{1}{N_a} \log\left(p_{seq2seq}\left(a|p_i, q_i\right)\right) + \frac{1}{N_{q_i}} \log\left(p^{backward}_{seq2seq}\left(q_i|a\right)\right) \tag{4}$$

Where:

- $p_{seq2seq}\left(a|q_i, p_i\right)$: the probability of generating response a based on previous dialogue utterances $[p_i, q_i]$.
- $p^{backward}_{seq2seq}\left(q_i|a\right)$: backward probability of generating the previous dialogue utterance q_i based on response a.

To control the influence of target length, both output of the two *log* function in the equation above are scaled by the length of targets.

7 Simulation

The core idea in the reinforcement learning solution being proposed is to simulate the process of two virtual agents taking turn engage in dialogue, where from there, we can explore the state-action space, discover and adopt a policy $p_{RL}(p_{i+1} \mid p_i, q_i)$ to achieve the optimal expect reward. With an AlphaGo-style strategy (Silver et al., 2016), we initialize the RL system by utilizing a general response generation policy, learned from a fully supervised environment.

8 Mutual Information

As we've mentioned, SEQ2SEQ will likely to generate dull responses e.g. "i don't know", "i don't know what are you talking about",... and this is the main drawback of the model. These dull response will end or lead the conversation to an infinite loop (repetitive). Thus we don't want to initialize a model with a policy based on pre-trained SEQ2SEQ model since it makes the RL model lacks experience diversities, which makes it easy to repeat the drawback mentioned above. Li et al. (2016a) showed that modeling mutual information between source and target reduces the chances of generating dull responses and increases the rate of quality responses. We can use the Encoder-Decoder model to generate maximum mutual information responses.

It is not possible to decode directly (Li et al., 2016a) from Equation 4 because the second term requires a complete generated target sentence to be able to generate a complete response. Base on work on sequence level learning by (Ranzato et al., 2015), we consider the problem of generating maximum mutual information responses modeled as an RL problem in which reward value and mutual information represent the model observing until the end of each sequence.

We initialize the policy model p_{RL} using a pre-trained $p_{\text{SEQ2SEQ}}(a \mid p_i, q_i)$ model and policy gradient method (Sutton et al., 1999; Williams, 1992) for optimization, which is previously used by Ranzato et al. (2015). We generate a candidate list $A = \{\hat{a} \mid \hat{a} \sim p_{RL}\}$ for each given input $[p_i, q_i]$. Then for each generated candidate $\hat{a}$, we will obtain mutual information score from using the pre-trained $p_{\text{SEQ2SEQ}}(a \mid p_i, q_i)$ and $p_{\text{SEQ2SEQ}}(q_i \mid a)$, similarly described in Equation 4.

$$m(\hat{a}, [p_i, q_i]) = \frac{1}{N_a} \log p_{\text{seq 2seq}}(a \mid q_i, p_i) + \frac{1}{N_{q_i}} \log p^{\text{backward}}_{\text{seqzseq}}(q_i \mid a) \tag{5}$$

The mutual information score is used as a reward and propagated back to the encoder-decoder model, adjust it to get a higher score. The expected reward for a sequence is given by:

$$J(\theta) = \mathbb{E}[m(\hat{a}, [p_i, q_i])] \tag{6}$$

Where:

- $\mathbb{E}$: expected value.
- m: mutual score function.

We estimate the gradient by using the likelihood trick defined as follow:

$$\nabla J(\theta) = m(\hat{a}, [p_i, q_i]) \nabla \log p_{RL}(\hat{a} \mid [p_i, q_i]) \tag{7}$$

We update Encoder-Decoder model's parameters by using stochastic gradient descent. According to to Ranzato et al. (2015), we will make use of a curriculum learning strategy (Bengio et al., 2009). The strategy proposed that for every sequence of length T, its first L tokens will be used with MLE loss and RL algorithm for the remaining $T - L$ tokens. We gradually decrease the value of L to 0. In order to reduce the learning variance, we will use a baseline policy. In additon, similar to the strategy used by Zaremba and Sutskever in 2015, we added a neural model which takes generated target and the initial source as inputs and return a baseline value. The final gradient is defined as follow:

$$\nabla J(\theta) = \nabla \log p_{RL}(\hat{a} \mid [p_i, q_i]) [m(\hat{a}, [p_i, q_i]) - b] \tag{8}$$

9 Dialogue Simulation between Two Agents

To apply reinforcement learning for our dialogue generation model, simulated conversations between two virtual agents where each agents take turn in dialogue are needed. This simulation is as follow: in the initialization step, a message from the training set is selected for the first agent. This agent then encode the message to a representation of vector and begin the decoding process to generate a response output. The immediate output from the first agent, combine with the dialogue history, then is used by the second agent by encoding the dialogue history as vector representation and utilize the decoder RNN to construct responses, revising the current state. This process is repeated until the simulation is terminated.

Optimization: The parameters from the mutual information model outlined in the previous subsection are used to establish the policy model p_RL. Then, we utilize policy gradient techniques to discover parameters that result in a larger expected reward. The expected future reward is the goal of maximization:

$$J_{RL}(\theta) = \mathbb{E}_{p_{RL}(a_{1:T})}\left[\sum_{i=1}^{i=T} R(a_i, [p_i, q_i])\right] \tag{9}$$

Where $R(a_i, [p_i, q_i])$ is the reward earned as a result of action a_i. For gradient updates, we apply the likelihood ratio method (Williams, 1992; Glynn, 1990; Aleksandrov et al., 1968):
Let $(a_i, [p_i, q_i]) = x_i$, we have:

$$\left[\sum_{i=1}^{i=T} R(a_i, [p_i, q_i])\right] = \left[\sum_{i=1}^{i=T} R(x_i)\right] \tag{10}$$

Put $r(x) = \left[\sum_{i=1}^{i=T} R(x_i)\right]$ into the equation, we have:

$$J(\theta) = \mathbb{E}_{p_\theta}[r(x)] = \sum_{x \in \mathscr{X}} p_\theta(x) r(x) \tag{11}$$

Where $p_\theta(x)$ is the probability of receiving reward x after performing an action sequence $p_1, q_1, p_2, q_2, \ldots, p_i, q_i$. By using Monte Carlo Estimator and algebraic transformation, we have:

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{p_\theta}[r(x)] &= \nabla_\theta \left[\sum_x p_\theta(x) r(x)\right] \\
&= \sum_x \nabla_\theta [p_\theta(x)] r(x) \\
&= \sum_x p_\theta(x) \frac{\nabla_\theta [p_\theta(x)]}{p_\theta(x)} r(x) \\
&= \sum_x p_\theta(x) r(x) \nabla_\theta \log p_\theta(x)
\end{aligned} \tag{12}$$

We continue to estimate based on Monte Carlo Estimator:

$$\nabla J_{RL}(\theta) \approx \frac{1}{T}[\sum_{i=1}^{i=T} r(x_i)\nabla_\theta \log p_\theta(x_i) \tag{13}$$

Substitute x_i and p_0, after performing some algebraic transformations, we will have:

$$\nabla J_{RL}(\theta) \approx \sum_i \nabla \log p(a_i \mid p_i, q_i) \sum_{i=1}^{i=T} R(a_i, [p_i, q_i]) \tag{14}$$

10 Curriculum Learning

We then apply curriculum learning strategy, as employed in section 8 by simulating dialogue for two turns at first, and raise the number of turns simulated incrementally to five, as number of candidates grows rapidly. Four candidate responses are generated for each simulation steps.

11 Evaluation

11.1 Human Evaluation

We evaluate the diologue generation system using our selected set of input sentences and judge the result quality manually on different settings and valuation metrics.
For human evaluation, we focused on three primary performance categories: Single-turn general quality, where from an input, subsequent generated output sequences from the SEQ2SEQ + Mutual Information model and a number of RL models are examined for sentence completeness, coherency, grammatical correctness, complexity, relevancy, and other properties of a good, quality response. Score is awarded to the model with the highest rating response. The mean difference in scores amongst the models is used to calculate the improvement made by the RL models over the SEQ2SEQ + Mutual Information model.

The second category is single-turn ease of answering, which uses the same format as the first but judges responses on attributes that allow for a natural and meaningful follow-up in a conversational setting, such as the use of tag questions, open ended statements, or the incorporation of relevant knowledge throughout the sentence.
The 200 sentences that make up our initial set of inputs for single-turn evaluation are divided into two categories: factual and situational questionaires e.g ”How are you feeling?”, ”What’s your favorite color?”, and statement and general expression sentences, e.g ”Nice to meet you”, ”I go to a university”. These sentences were chosen for their readability and range of expression, as well as their use of grammatical structure and vocabulary, with an emphasis on sentences of similar difficulty.
Multi-turn general quality is the third category in the human evaluation procedure. For this, each model presents a five-turn simulated discussion between two virtual agents, where in the first turn, an input is given to the first agent to initiate the dialogue. We then judge these outputs on characteristic of a natural conversation such as conversation flow, coherency, subject relevance exchanges, and information introduction, to then award score to the model having the highest quality output.

11.2 Experimental Results

In this section, we will conduct experiments with the SEQ2SEQ model using 10000 epochs and based on this model to continue training the RL model with 5 different pairs of reward distributions:

1. λ_1=0.2,λ_2=0.3,λ_3=0.5
2. λ_1=0.2,λ_2=0.2,λ_3=0.6
3. λ_1=0.3,λ_2=0.3,λ_3=0.4
4. λ_1=0.4,λ_2=0.2,λ_3=0.4
5. λ_1=0.25,λ_2=0.25,λ_3=0.5

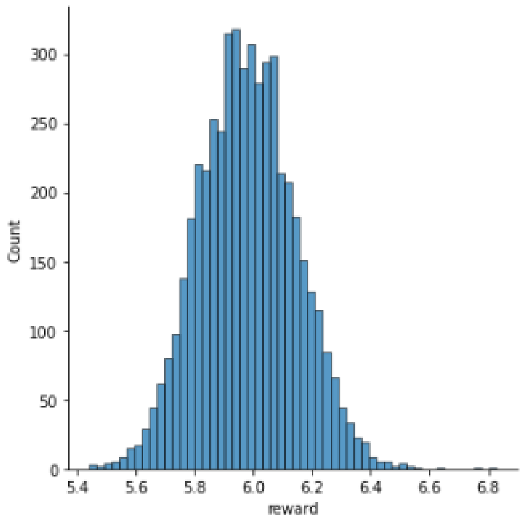


Figure 1: Reward distributions with λ_1=0.2,λ_2=0.2,λ_3=0.6

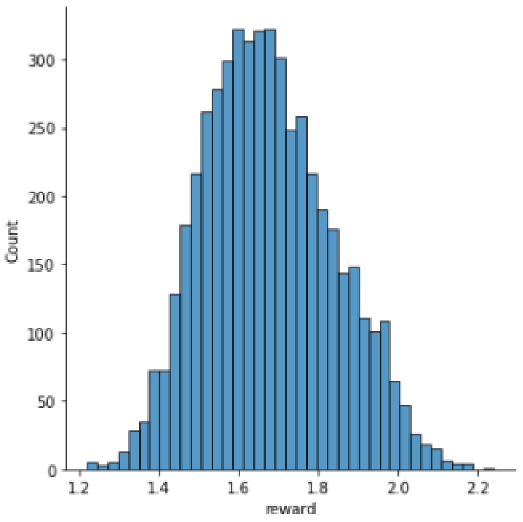


Figure 2: Reward distributions with λ_1=0.3,λ_2=0.3,λ_3=0.4

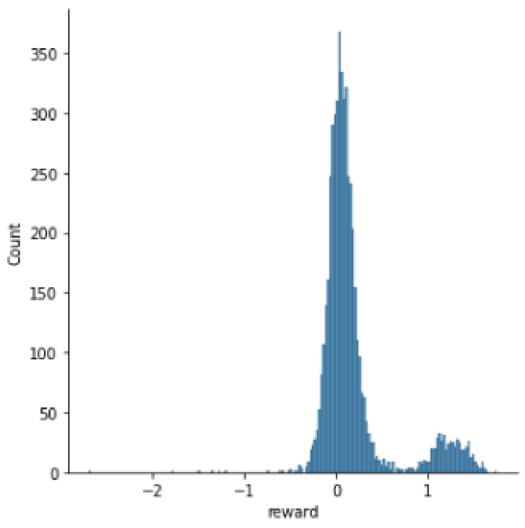


Figure 3: Reward distributions with λ_1=0.4,λ_2=0.2,λ_3=0.4

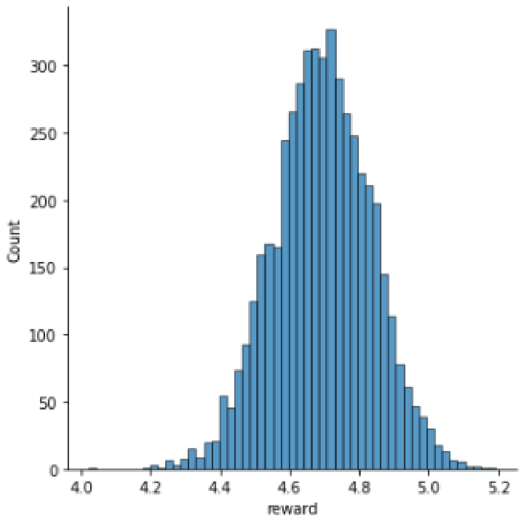


Figure 4: Reward distributions with λ_1=0.25,λ_2=0.25,λ_3=0.5

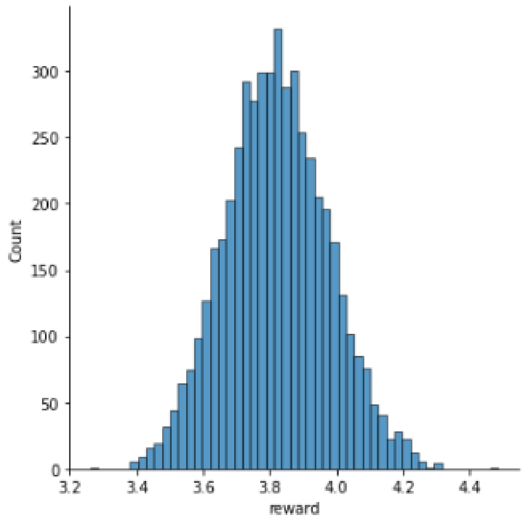


Figure 5: Reward distributions with λ_1=0.2,λ_2=0.3,λ_3=0.5

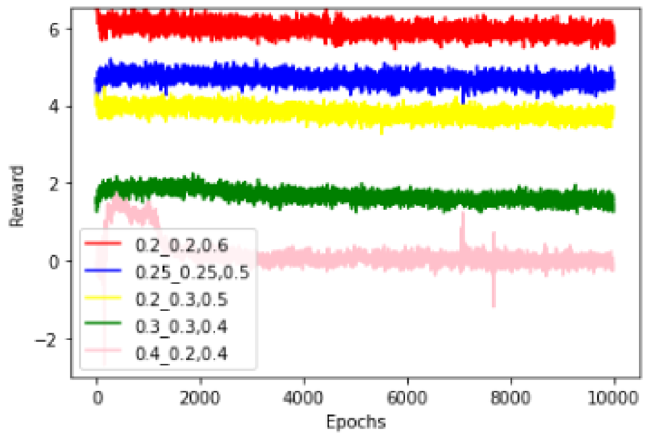


Figure 6: Reward distributions summary

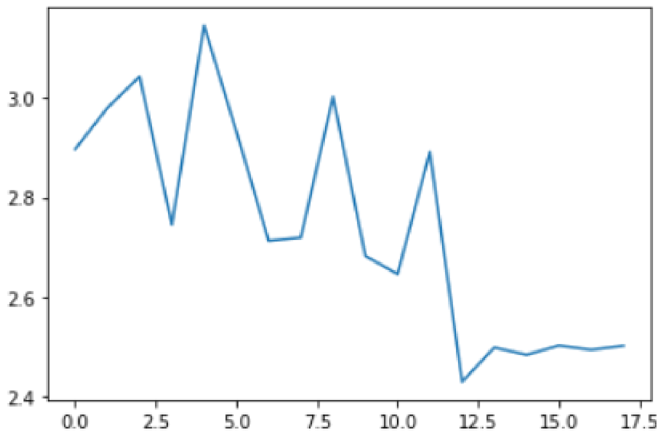


Figure 7: Training Loss with λ_1=0.2,λ_2=0.2,λ_3=0.6

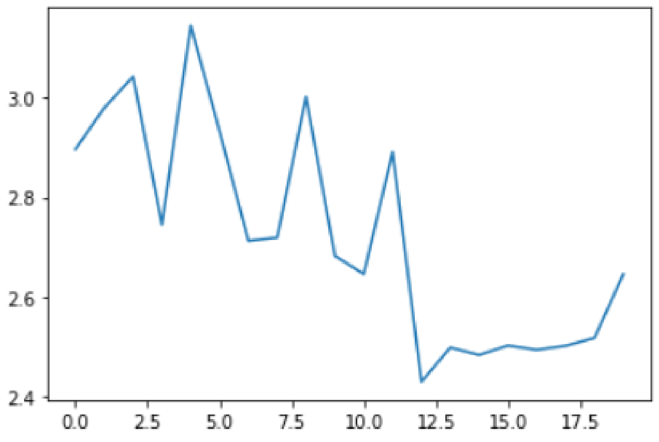


Figure 8: Training Loss with λ_1=0.2,λ_2=0.3,λ_3=0.5

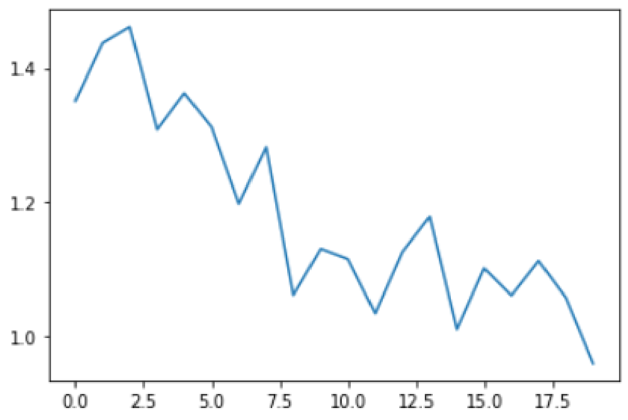


Figure 9: Training Loss with λ_1=0.3,λ_2=0.3,λ_3=0.4

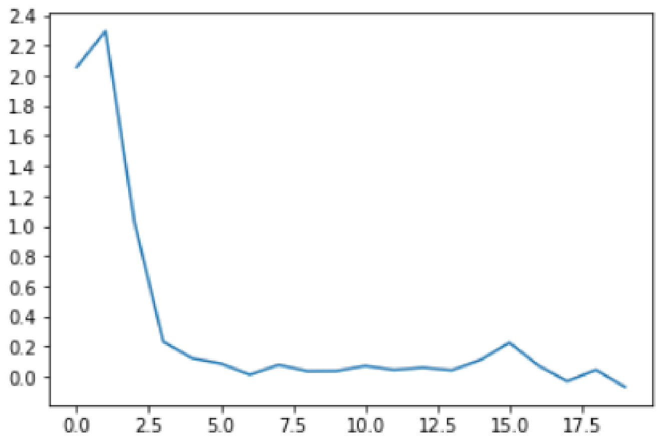


Figure 10: Training Loss with λ_1=0.4,λ_2=0.2,λ_3=0.4

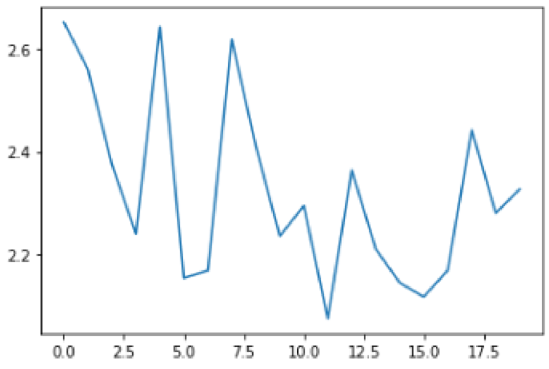


Figure 11: Training Loss with λ_1=0.2,λ_2=0.2,λ_3=0.6

We can clearly see that the results from these models all follow a standard distribution which is simply mean that the training process is relatively stable.

Meanwhile, loss functions (except from λ_1=0.25,λ_2=0.25,λ_3=0.5 and λ_1=0.2,λ_2=0.2,λ_3=0.6 models) are all converged after 10000 iteration in training process, though, if we consider in terms of the reward then those two models have the best reward.

We can notice that the reward distributions for each model are relatively stable in which the model with λ_1=0.2,λ_2=0.3,λ_3=0.5 has the highest reward and the model with λ_1=0.4,λ_2=0.2,λ_3=0.4 has the lowest reward. We find that the obtained reward will change after adjusting the parameters. In the beginning of the training process, the reward gradually increases and slowly becomes stable after approximately 2000 epochs.

So in the end, does the amount of reward decide the quality of generated sentences? We have a comparison table of the responses quality for the same question.

Setting	RL-win	RL-lose
Single-turn general quality	0.57	0.43
Single-turn ease of answering	0.61	0.4
Multi-turn general quality	0.6	0.4

Table 1: RL take the lead based on human judgements

Result for the human evaluation process are presented in the table below. We can see the performance in single-turn general quality and ease of answering of our chosen RL models are higher by a significant margin (winning 57 and 61 percents respectively) to the SEQ2SEQ + Mutual Information model. This while reflects the RL model's ability to generate on-par to superior sentences, might also be effected by the input chosen or evaluation methodology as the result shown here is deviates from the original research. The RL models also outperform the SEQ2SEQ + Mutual Information model on multi-turn general quality, generating consistently more natural and coherent conversations at a 60 percent win rates, in line with known result.

12 Results Analysis

From the results below, we can conclude that for single-turn, the RL with λ_1=0.25,λ_2=0.25,λ_3=0.5 and λ_1=0.2,λ_2=0.2,λ_3=0.6 has the highest reward, generated turns from these two model are both grammatical and easy to continue the conversation, while the other threes don't. Thus we eliminated those models and continue training the two best model (in terms of both reward and actual results).

We then compared the three remaining models and the results we get back is relatively equal between them in terms of gramma and coherence of generated sentences. However, RL is more likely to generate ease to answer turns (sentences relevant to the conversation topic as well as making question for the other agent).

Input	Mutual Information + SEQ2SEQ	RL 0.2 0.2 0.6
How old are you?	twenty one.	That's a secret .
What is your full name?	my name is james.	my name is jean .
Where are you going?	i told you already ! I'm going to get my nails done.	i have a big date tonight .
What are you going to do today?	i'm not sure yet . and you ?	i'm going shopping.

RL 0.25 0.25 0.5	RL 0.4 0.2 0.4	RL 0.3 0.3 0.4	RL 0.2 0.3 0.5
i am twenty five years old .	i m fine . .	i m fine . .	twenty one . .
my name is sam.	my name is james . .	my name is sam lin . my .	my name is . how is my .
i m going to to the hospital .	i m ... to the the center .	i m visit station department .	i m go to to nanchang nanchang n .
i m not sure yet .	i m not . yet	i m not . yet	i m going to i . . . .

Table 2: Sampled responses generated from the mutual information + seq2seq model and five proposed RL models

Sentences	Mutual Information + SEQ2SEQ
Can I pay by credit card?	sorry . you can only pay in cash .
How are you feeling?	bad. my nose keeps running.
Are you free tomorrow?	yeah what's up ?
Would you like to have dinner with me?	oh that would be lovely. shall we say seven o'clock ?
Where do you want to go?	i think we should go out to eat.

RL 0.2 0.2 0.6	RL 0.25 0.25 0.5
of course . please sign your name please .	of course. can i have your name please ?
bad. my nose keeps running. my forehead is burning .	bad . my forehead is burning.
yes i am . what's up ?	yeah what s up ? ? ?
oh that would be lovely . what time will you be here .	oh that would be nice . what do you want then ?
i want to go to the capital hotel .	i'd like to go to the italian restaurant.

Table 3: Sampled responses generated from the mutual information + seq2seq model and two best proposed RL models for further measures

Meanwhile, in multi-turn generated dialogues from both the SEQ2SEQ + Mutual Information and reinforcement learning models, as shown in Table 4, the SEQ2SEQ model failed to generate relevant sequence between turns, while with the RL model, the conversation transitions smoothly from topic to topic, while still retaining relevance and maintaining a natural flow with easy to answer follow-ups, suffer only from sentences with less coherency at the end.

	RL	Mutual Information + SEQ2SEQ
A/Input	How are you at school?	How are you at school?
B	im doing great..	i m doing good for my parents . .
A	what movies have you seen lately ? ?	don t be so happy . i was a happy new year .
B	i saw forrest gump the other day . .	what about your new job ?
A	what type of movie is that ?	i m a keyboard operator . what s it doing ?
B	the movie type is drama .	i m not sure . .
A	like my table ?	i didn t know you knew . .
B	you can be there no . that s about the regular .	don t you know i ll beat him ? .
A	i ll tell be right t be .	i ve never got money . .

Table 4: A simulated dialogue with eight turns

12.1 Qualitative Analysis and Discussion

We will compare generated sentences based on 3 criteria:

1. Ease of Answering
2. Information Flow
3. Semantic Coherence

Information flow and Semantic Coherence are important because it is obvious that if the question is incoherent or ungrammatical then it is infeasible to generate a valid response. While Information flow provides necessary information to generate relevant responses, Ease of Answering will keep the conversation continue.

As you can see the general quality of sentences generated by

SEQ2SEQ model compared to RL is less interactions. In other words, responses as questions will increase interaction between two agents. With RL, there is high chance that one agent will generate a question for the other one which means that interactions between agents are significant higher than Mutual Information + SEQ2SEQ model.

In the error analysis process, repetitive words and sentences still occur although we have penalty for them (maximum turns for the simulated conversation is 8), mainly because the information flow from the previous turns is not enough to generate sentences to prevent the conversation from becoming boring.

Another problem is that newly generated sentences tend to have less relevance to the current conversation topic. There is a strong correlation between information flow and semantic coherence (less boring) as describe in Equation 1. The main idea of those problems above, of course, is that designed heuristic reward functions neither cover nor affect the entire ideal conversation (many important aspects of a conversation are unmentioned) and reward functions are manually constructed. When the heuristic reward function we defined above is adaptive to the automatic calculation and implements some aspects to make a good conversation. An ideal system would directly receive rewards from humans (judged by real human or at least the amount of reward will correspond to the quality of generated turns). Another problem with the model is that for now we are only able to explore a tiny set of candidates and simulated turns since the grow rate of cases need to be covered is exponental (similar to Breath First Search).

13 Limitations

We can see that the Greedy Search to find the response is really inefficient because the solution will only select the words with the highest score (making the responses incoherent or grammatically incorrect because the selected word has no compatibility with the neighboring words).

So, just considering the question criteria based on the reward types does not completely reflect the results for the response because adjusting the parameters $\lambda_1, \lambda_2, \lambda_3$ we also need to maintain the correlation between them. In conclusion, having a high reward does not lead to a good response.

14 Conclusion

By simulating conversations between two agents, we experimented with building a reinforcement learning model for neural response generation, merging the strengths of neural SEQ2SEQ systems in sequence generation with reinforcement learning for its heuristic reward. Our approach, like previous neural SEQ2SEQ models, captures the compositional models of the meaning of a dialogue turn and provides semantically relevant answers. Our approach, like reinforcement learning dialogue systems, can create utterances that optimize future reward, successfully capturing broad features of a good discussion. Despite the fact that our model captures these global features using relatively simple, operable criteria, the model creates more diverse, engaging answers that encourage more prolonged dialogue, as well as increasing penalties for grammatical errors and training models for longer periods of time, in order to build a model with high learning abilities and the capacity to communicate in a natural, engaging, and engaging fashion.

15 Acknowledgement

We would like to thank Dr. Luong Ngoc Hoang for his insightful recommendations and for providing us with valuable materials in our attempts to write this paper. We also want to thank each individual in this team for their hard work. Since this is our very first scientific paper, any findings or judgements we reach are solely based on our knowledges and experiences as second year college students, and they may not be completely accurate or reflect the actual outcomes. Any feedbacks or suggestions from readers would be greately appreciated.

References

[0] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao & Dan Jurafsky (2016). *Deep reinforcement learning for dialogue generation.* Arxiv. https://arxiv.org/abs/1606.01541.

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk & Yoshua Bengio (2014). *Learning phrase representations using RNN Encoder-Decoder for statistical machine translation.* Arxiv. https://arxiv.org/abs/1406.1078.

[2] Oriol Vinyals & Quoc Le (2015). *A neural conversational model.* Arxiv. https://arxiv.org/abs/1506.05869.

[3] Dzmitry Bahdanau, Kyunghyun Cho & Yoshua Bengio (2015). *Neural machine translation by jointly learning to align and translate.* Arxiv. https://arxiv.org/abs/1409.0473.

[4] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville & Joelle Pineau (2016). *Building End-To-End dialogue systems using generative hierarchical neural network models.* Arxiv. https://arxiv.org/abs/1507.04808.

[5] Abdelrhman Saleh, Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen & Rosalind Picard (2019). *Hierarchical reinforcement learning for Open-Domain dialog.* Arxiv. https://arxiv.org/abs/1909.07547.

[6] Shakir. (2015). *Log derivative trick.* Shakir's Machine Learning Blog. http://blog.shakirm.com/2015/11/machine-learning-trick-of-the-day-5-log-derivative-trick/.

[7] Marcella Cindy Prasetio, Mustafa Abdool, & Lam, C. (unknown year). *Dialogue generation using reinforcement learning and neural language models.* Stanford. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.-1184/reports/6851458.pdf.

[8] Matthew Inkawhich. (unknown year). *Chatbot tutorial.* PyTorch. https://pytorch.org/tutorials/beginner/-chatbot_tutorial.html.

[9] Lilian Weng. (2018). *Policy gradient algorithms.* Lil'Log. https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html.