Deep Reinforcement Learning For Language Models

Eyal Ben David

Abstract

Recent work on neural models of dialogue generation offers great promise for generating responses for conversational agents, while presenting a reward function that reflects "how good is a target answer", aiming towards a clear and fluent conversation. In order to train a good dialogue model, a good reliable language model is necessary, as these model's objective is calculated using a fixed language model (Li et al., 2016). In this work, we show how to use traditional evaluation methods used in NLP to improve a language model. We construct couple of models, using different objective for each model, and comparing between them, while offering some intuition regarding the results. For establishing this task we use Deep Reinforcement Learning methods, specifically policy gradient methods, to reward sequences that achieve high evaluation values. We evaluate these models on traditional NLP evaluation methods and present generated dialogues by agents. Most of these models outperform a baseline cross entropy (CE) seq2seq model both using automatic evaluation and qualitative evaluation.

Model/Method	BLEU	Mutual Information	Perplexity	Cosine Similarity
BLEU	**0.1209**	-7.4535	**1.4720**	0.2953
MMI	0.1186	**-7.2094**	1.6988	0.3272
Perplexity	0.1189	-8.4770	1.5711	**0.3647**
Cosine Similarity	0.1088	-7.6886	1.9681	0.3231

Table 1: Comparing all models using traditional NLP automatic evaluation methods.

1 Reinforcement Learning in Language Models

A dialogue can be presented as a sequence of sentences generated by couple of agents $q_1, p_1, q_2, p_2, ..., q_i, p_i, ...$, where p represents sentences generated by the first agent and q represents sentences that were generated by the second agent. Each sentence generated, p_i, is a sequence of words generated one at a time, i.e

$p_{i,1}, ..., p_{i,n}$. In the following subsections i describe each of the components used to generate a target response, p_i, given a source sentence, q_i.

action

An action $\tilde{a}$ is the subsequent word to be generated in a dialogue for completing the sentence. The action space is the size of the unique words from training corpus.

state

The state $\tilde{s}$ at each time step is the hidden value and the output value of the previous LSTM cell. The hidden value represents all accumulated history, i.e the source sentence and the already generated sequence of the target sentence. The output value represents the final word generated.

policy

A policy is defined as $p(\tilde{a}|\tilde{s})$, i.e probabilities choosing the next word in the target sentence given the hidden value accumulated so far and the last word generated.

reward

r denotes the reward obtained for each action. In practice, the agent achieves a reward only when it selects the 'EOS' token, i.e the End Of Sentence token.

2 Defining Rewards

In this section i presents different reward methods that have been long used as evaluation measures in NLP tasks, such as BLUE, MMI, Cosine Similarity and Preplexity. Previous works have tried to use some of these objectives as a loss function for a supervised learning back propagation, resulting in models that did not perform well in different tasks. Here i use these objectives as a reward for the sequence generation agent.

BLEU

BLEU (Papineni et al, 2002) is a score for comparing a candidate target sentence in a text to one or more reference sentences. A perfect match results in a score of 1, whereas a perfect mismatch results in a score of 0. The approach works by counting matching n-grams in the candidate target to n-grams in the reference text. I chose to check up to 2-grams sequences while weighting them equivalently. The reward is defined as

$$r_{BLUE} = \frac{1}{2}(\frac{M_{uni}}{T_{uni}} + \frac{M_{bi}}{T_{bi}})$$

where M_{uni} and M_{bi} are the total amount of uni-grams and bi-grams respectively that appear both in target sequence and in reference sequence, and T_{uni} and T_{bi} are the total amount of uni-grams and bi-grams which appear in a target sequence respectively.

Maximum Mutual Information

As mentioned in (Li et al., 2016a), the MMI aims to prevent the use of dull sentences as answers, as the basic Seq2Seq model often does tend to generate (e.g. "i don't know"). For the MMI reward we pre-train a basic Seq2Seq model and a backward Seq2Seq model that generates source sentences given the target sentences. Hence the MMI reward is defined as

$$r_{MMI} = p_{seq2seq}(p_i|q_i) + p_{b\_seq2seq}(q_i|p_i)$$

where $p_{seq2seq}(p_i|q_i)$ is the probability, based on the pre-trained Seq2Seq model, of generating the sentence p_i given the source sentence q_i and $p_{b\_seq2seq}(q_i|p_i)$ is the probability, based on the pre-trained backward Seq2Seq, that the sentence p_i was generated from the source sentence q_i.

Perplexity

Perplexity is a measurement of how well a probability model predicts a sample. In the context of Natural Language Processing, perplexity is one way to evaluate language models. The definition of the perplexity loss is:

$$L_{per} = 2^{-l}, \textbf{ where } l = \sum p_{seq2seq}(p_i|q_i) \log p_{seq2seq}(p_i|q_i)$$

and the reward is: $r_{per} = -L_{per}$

Cosine Similarity

Cosine similarity is a quantitative measurement of the similarity between two given vectors. We use the cosine similarity in order to measure how similar are the generated target sequence to the reference sequence. For this task we use the representations of the target sequence and the reference sequence out as the model LSTM layer represents them:

$$r_{cosine} = \frac{h_{target}}{|h_{target}||} \cdot \frac{h_{ref}}{|h_{ref}||}$$

3 Training

In this section we show how to train an agent in order to maximize the accumulated reward it receives during a trajectory, while each of the rewards, which we defined in the previous section, may be its objective. We use a policy gradient methods for optimization. We initialize the policy model p_{RL} using a pre-trained $p_{seq2seq}(a|p_i, q_i)$ model.

We choose to use an Actor-Critic method for optimizing the model. The objective of the network is to minimize the following loss function:

$$L(\theta) = -J(\theta)$$

where $J(\theta)$ is the expected return.

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\sum_t \gamma^t r(s_t, a_t)\right] = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right]$$

Practically, we estimate the gradient of this loss as:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}(s_t, a_t)$$

where $\hat{A}(s_t, a_t) = Q(s_t, a_t) - \hat{V}_\theta(s_t)$. Instead of using another system to calculate the value function given a state, we use the argmax sentence generation method to do so. That is, for each step in the generated sequence, we choose an action such that it maximizes the policy given the state at the beginning of the step.

Furthermore, we normalize the reward for every batch to stabilize the learning process.
Applying this algorithm to the text generation problem might diverge or converge to a bad solution, far from the original MLE parameters. To prevent this from occuring, we use curriculum learning method which we further elaborate on in the following subsection. Using this method, the policy gradient algorithm fine tunes the basic seq2seq model, and as we demonstrate in the experiments section, yields a system that generates better responses than the original seq2seq.

3.1 Curriculum Learning

Curriculum learning (Bengio et al., 2009) methods are implemented during training session in order to get a steady and fast convergence, experiments also show that using these methods yields a system which has an abillity to generalize better. During training session, for each batch, in probability $\hat{P}$, we use the cross-entropy criteria for back-propagating, where $\hat{P}$ decrease during training such that at an advanced stage of training the system back-propagates using only policy gradient.

3.2 Imitation Learning

Imitation learning approaches the sequential prediction problem by seeking clever ways of using expert demonstrations (training dataset) to learn a policy that makes predictions (actions) that approximate expert actions. In the case of dialogue generation, expert actions are widely available, as they are simply the reference answers from the dataset. During training episode, once in a few batches, we force the agent to answer the reference sequence and back-propagate it to achieve a better policy. Empirically it leads to a steadier and faster convergence.

4 Experimental Results

In this section, we present experimental results along with qualitative analysis. We evaluate language model systems using both common NLP automatic metrics such as BLEU, Perplexity, Cosine Similarity and Mutual Information.

We divide experiment to two part: (1) we first want to compare between a MMI model trained using reinforcement learning methods (Ritter et al., 2016) and MMi model which is using the criteria only at test time (Li et al., 2015). (2) We display a comparison between models which use reinforcement learning methods with different reward targets.

4.1 Evaluate MMI criteria as a reinforcement reward

The traditional MMI model (Lee et al., 2015), is implemented using two pre-trained model, (1) Cross entropy (CE) seq2seq and (2) backward CE seq2seq. After pre-training, on test time, the forward model is being used to generate a beam of response sentences with the highest probabilities, using beam search methods. The MMI criteria is then calculated on the entire beam and the response with the highest MMI criteria is chosen as the model response. Theoretically, using a very high beam size should generate the best response. We demonstrate in Table 2 that using RL methods which have been explained earlier, yields a model which is competitive with the beam search model, while its run-time is much faster, varying from 1,000 to 10,000 times faster.

Input Sentence	MMI Model	Answer Sequence	Runtime (sec)
How old are you?	RL	Thirty-five.	0.001
	Beam Search (100)	Thirty-five.	3.220
Hi, Joey.	RL	Hi, stacy.	0.001
	Beam Search (100)	Hi, stacy.	3.745
What would you like to eat?	RL	Nothing.	0.001
	Beam Search (100)	Big terrible.	6.539
Do you like to play football?	RL	No.	0.001
	Beam Search (100)	No. but i liked jesus.	4.505

Table 2: Comparison of target sentences generation given a source sequence between a model of MMI trained using reinforcement learning policy gradient and a model which is calculating MMI only at test time.

4.2 Evaluation of different rewards for policy gradient method

As explained earlier, we have trained couple of language models, using different reward for each model. From Table 3 and Table 1 we deduce that automatic evaluation is a problematic method for language models. As summarized in the former, it stands out that the perplexity model generates the most appropriates sentences given an input sentence, while looking at the latter may imply that BLEU model generates best responses.

Input Sentence	Model	Answer Sequence	Input Sentence	Model	Answer Sequence
	CE Seq2Seq (Baseline)	You know good.		CE Seq2Seq (Baseline)	Hm?
	BLEU	I don't know.		BLEU	Hm.
What's your full name?	MMI	I'm not bad.	Whats that?	MMI	The letter.
	Perplexity	Simon.		Perplexity	Nothing.
	Cosine Similarity	I don't know.		Cosine Similarity	Hm.
	CE Seq2Seq (Baseline)	Fine.		CE Seq2Seq (Baseline)	I'm you and i'm myself to her the holy,.
	BLEU	Fine.		BLEU	I'm- if you.
How old are you?	MMI	Thirty-five.	Where are you?	MMI	I'm going.
	Perplexity	Thirty-five.		Perplexity	I'm coming.
	Cosine Similarity	Fine.		Cosine Similarity	I'm you.. Can i.
	CE Seq2Seq (Baseline)	Christy, you hello?		CE Seq2Seq (Baseline)	Isn't he incredible, the for two.
	BLEU	Christy you?		BLEU	Isn't he incredible,.
Hi, Joey.	MMI	Hi, Stacy.	Let's go.	MMI	Isn't he incredible, gets along with everybody.
	Perplexity	Hi, Mary.		Perplexity	Where?
	Cosine Similarity	Christy you.		Cosine Similarity	I ...
	CE Seq2Seq (Baseline)	For a while.		CE Seq2Seq (Baseline)	Using filthy language.
	BLEU	Head hunting.		BLEU	Using filthy.
Where are you going?	MMI	Head hunting.	Excuse me?	MMI	Using filthy language in my garden.
	Perplexity	Head hunting.		Perplexity	I'm sorry.
	Cosine Similarity	Head hunting.		Cosine Similarity	Using filthy.

Table 3: Comparison of target sentences generation given a source sequence between Perplexity, MMI, BLEU and Seq2Seq.

Automatic evaluation for NLP tasks such as BLEU and Cosine Similarity represents evaluation methods taken from another task originally. For example, the BLEU method is widely used for MT tasks, as it measures the number of sequences of changing lengths that appear both in the reference sentence and in the generated sentence. For the dialogue generation task, many sentences might be valid for a given input, while deferring in all its sequences, making the evaluation problematic.

Furthermore, cosine similarity model is based on representing an entire sentence with a vector. Here we use, as most often do, the mean of all words in a sentence as a representation of the sentence. This is a naive approach because a sentence is much more complicated than a single word, hence it cannot be represented as a vector in the same dimension. This probably causes the model trained with the cosine similarity reward to converge into a model which generates syntactic illogical sentences.

5 Conclusions

In this work, we had shown how to use traditional evaluation methods used in NLP to improve a language model. We had constructed a couple of models and compared them to a baseline model. Two models, Perplexity and MMI, reached good results and definitely outperformed the baseline. We had also conducted some automatic evaluations, while giving some intuition to the reason of why some of them defer from a human evaluation. On top of these improved language models a dialogue generation model can be developed, as in Li et al.(2016).