Reinforcement Learning on Sentiment Poem Generation

COMP0087 GROUP 8

Abstract

Large Language Model with transformers such as T5 and GPT displays excellent performance on numerous important NLP tasks, including machine translations, sentiment analysis, text summarization, etc. There are numerous works that integrate reinforcement learning with language models, such as reinforcement learning from human feedback (RLHF) to train ChatGPT, as a result of the recent advancements in reinforcement learning. This paper applies the reinforcement learning method PPO to the generation of positive poems with varied reward signal designs based on GPT-2. In addition, we train a positive poetry generating model using the conventional NLP technique, clm loss, as a comparison to the RL technique. We may conclude that the RL technique has the ability to achieve a better sentiment accuracy and enhance agent exploration, consequently increasing diversity and novelty.

1 Introduction

Affective natural language generation refer to a Natural Language Generation (NLG) task aim to produce textual content that is able to convey a certain emotion understandable by human. This kinds of task usually have deep connections with cross-disciplines such as affective computing (Picard, 2000), and have vast use-case in the industry, for example, chat-bot capable to understand human's emotion used in psychotherapy (Xu and Zhuang, 2022), generating medicine description in a patient friendly way (Paiva and Evans, 2004), or creative writing of poetry that reflect user's "inner-self" (Aslam and Batenburg).

A usual way to achieve this is through guided neural language model (LM), where a structured latent space of language is learnt under then guidance from a style discriminator (Hu et al., 2017). Such that the stylistic dimensions are disentangled and can be extracted to condition the decoder.

However, the use of language model are usually highly specialized and reported failure when applied on different kinds of natural language (e.g., a Wikipedia LM applied for poetry writing), thus require a from-scratch re-training. As result, large language model (LLM) involve deep neural network was developed and pre-trained on large and vast corpus in an unsupervised fashion, enabling faster transfer to other task with minor fine-tunings. Examples of this kind of model including the well-known transformer-based language model BERT (Devlin et al., 2019) which is reported a decent performance improvement on several NLP task. And recent advancement such GPT2 (Radford et al., 2019), demonstrate an further improvements on the overall performance.

Although the use of LLM as front-end of aforementioned latent space disentanglement method seems to be promised, the diversity is often a short comings when a certain degree of creativity is required in the task, such as poetry continuation from a given beginning. Thus, many researchers shift their interests in combining language models with reinforcement learning (RL). In the context of affective NLG, this kind of learning paradigm involves the use of some reward function that "critic" the style of the overall generated text (state) after a new token is predicted and added (actions), such that the model learns from experience and produce desire language style. The reinforcement learning is famous for it's generalisability and novelty of outcomes due to the continuously exploration of possible actions.

In this report, We argue that the use of reinforcement learning paradigm in the setting of sentiment poem generation can not only increase sentiment accuracy, but also novelty and diversity compared to traditional training paradigms. We will demonstrate a novel fine-tuning method for GPT2 (Radford et al., 2019), utilising on Proximal Policy Optimization (PPO) (**?**) for generating poetry with

positive emotions. We will evaluate our model against a non-RL baseline to show the supremacy in creativity and novelty of generated poems.

2 Related Work

Sentiment generation is a subset of border class of affective NLG, where the emotions conveyed by text usually be either positive or negative. One should aware that this is different from sentiment analysis where it is a classification problem. We will review some works on sentiment NLG in this sections.

2.1 Conventional Sentiment Generation

Earlier attempts in achieving sentiment generation relies on explicit modelling of grammatical structure (Enkvist, 1973) and use of evolution algorithm with handcraft features (Gonçalo Oliveira, 2017). Notable works such as (Oliveira, 2012) proposed a system that based on grammatical template mutations. Modern techniques are commonly based on GAN architecture, where a style or emotion discriminator are jointly trained with decoder-encoder pipeline such as (Hu et al., 2017) learns disentangled latent space for general style transfer, (Rajeswar et al., 2017) proposed a WGAN based method for generating different types of text (e.g., poems and reviews) with sentiment conditioning.

A similar system called PoetryMirror reported in (Aslam and Batenburg) that is able to write poems to reflect user's current affective state identified through facial expression. Their method involves use of several GPT2 models each of them fine-tuned on poetry with different emotions. The facial sentiment classifier is used to decide which generator should be used to generate the poem.

2.2 RL on Language Model

Most NLG task as we surveyed in the previous section are follows traditional supervised deep learning paradigm, where a loss function measuring the predicted and ground truth are used and the model are fine-tuned or trained through gradient descent through the loss surface. Recent development of reinforcement learning demonstrates the impressive performance on trained model such as (Silver et al., 2017), has drawn the NLG community's interest in adopting RL as learning paradigm as alternative to traditional method.

Adaptations on GAN-style model has explored by (Shi et al., 2018), following the Inverse RL framework (Ziebart et al., 2008) where the reward function (discriminator) is learnt from training data and it is optimised together with policy (generator) in an alternating fashion, such that a dense reward signal is encouraged and the generator can be learnt by entropy regularized policy gradient (Finn et al., 2016) to maximize the text diversity. Similar works based on IRL such as (Li et al., 2018) generating paraphrase using RL fine-tuned GAN and discriminator learnt by IRL.

3 Methods

In this paper, we use the reinforcement learning method, which is an Actor-Critic style Proximal Policy Optimization (PPO) with clipped surrogate objective, to train the poem generation model (GPT-2 based) to generate the positive poems.

Figure 1 illustrates a summary of our RL framework for peom generation. We fine-tune a pre-trained GPT-2 model on peom generation task, while the original model serves as a reference. During training, we query the prompts containing a small number of words from dataset. The active model will then provide responses based on these prompts. The initial portion of the reward could consist of the sentiment score based on the response, with or without a bertscore based on the responses from both the active model and the reference model. The KL divergence between the active model and the reference model is another component of reward. After summing these parts, the active model is optimized using PPO by this reward.

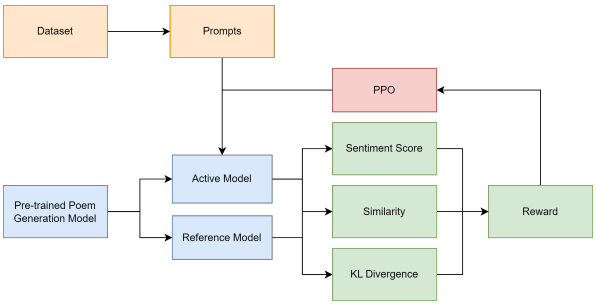


Figure 1: RL on Poem Generation

In this section, the integration of PPO and language model (Poem generation GPT in our implementation) objective and the reward signal we designed for this poem generation task will be explained with the background knowledge of reinforcement learning, actor-critic methods and PPO, respectively.

Reinforcement learning is the problem experienced by an agent that must learn behavior through interactions with a dynamic environment, such as performing an action and receiving a reward (Kaelbling et al., 1996). Workflow of reinforcement learning framework typically includes a set of states ($\mathcal{S}$) representing the current environment, a set of actions ($\mathcal{A}$) from which the agent can choose, and a set of rewards ($\mathcal{R}$) provided by the predefined functions. Agent would select the optimal action based on the current state, and the agent's policy would be adjusted after a reward value is assigned based on the action taken. In our poem generation task, action refers to the tokens generated using the action space of the entire vocabulary dictionary, while state refers to the tokens generated previously. Reward could be manually defined using sentiment score and KL divergence, for example.

3.1 Actor-Critic Reinforcement Learning

There are three distinct classifications of reinforcement learning: actor-only, critic-only, and actor-critic approaches, where actor and critic are equivalents for the policy and value function, respectively (Grondman et al., 2012).

For the actor-only method, policy of actor is parameterized and could be updated directly using optimization procedures like REINFORCE (Williams, 1992). The critic-only method selects actions based on the state-action function and updates the function using temporal difference (TD) learning, as demonstrated by Q-learning (Watkins and Dayan, 1992) and Sarsa (Rummery and Niranjan, 1994). Furthermore, the objective of actor-critic methods is to combine the benefits of actor-only and critic-only approaches (Konda and Tsitsiklis, 1999). Involvement of a critic in actor-critic methods could reduce the large variance in policy gradients of actor-only methods and evaluate the actor's present policy to optimize the speed and direction of convergence.

In our implementation, actor is a pre-trained poem generation GPT-2 model (policy π with parameters θ) and critic is the value head at the top of this actor model, it returns a value that evaluates the current policy based on the generated tokens, which is also the current state ($V\left(s_t\right)$).

3.2 PPO

Proximal Policy Optimization (PPO) is commonly recognized as the most efficient approach in Reinforcement Learning. According to the results presented in the paper (**?**), PPO outperforms its predecessors in nearly all continuous control environments. The PPO method is introduced to our poem generation design. To facilitate comprehension, we will now discuss the various PPO method types. Briefly, there are three different settings of proximal policy gradient (**?**).

Given the policy π with parameters θ and the reference policy model's parameters as θ_{old}, we could denote the probability ratio $r_t(\theta)$ at time t as following:

$$r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{1}$$

Let $\hat{A}_t$ be an estimator of the advantage function, and the method for updating this advantage function, which does not look beyond timestep T, is illustrated below:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1} \tag{2}$$

δ_t shown in (2) could be expressed using a critic, V, we mentioned in the actor-critic reinforcement learning previously and a reward r at the current timestep t.

$$\delta_t = r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{3}$$

Initially, the simplest form of PPO is to maximize the trust region policy optimization (TRPO) objective without a constraint as what is shown in (4), which means there might be no clipping or penalty (Schulman et al., 2015).

$$L_t(\theta) = r_t(\theta)\hat{A}_t \tag{4}$$

Secondly, clipped surrogate objective is proposed by (**?**) as shown in (5) by introduing a clip function. This is intended to prevent a significant update to the policy model. PPO restricts its excessive updating in the positive direction (increased likelihood) when the action is a prudent option and limits its excessive update in the negative direction (decreased probability) when the action is harmful. Specifically, when the value of the advantage function is greater than zero $A_t > 0$, if the ratio of update of current policy (π_θ) under the current action and the old policy ($\pi_{\theta_{old}}$) is too large, which is greater than $1 + \epsilon$, the gradient about this reward would be zero, indicating that its backward would not update the parameters. Otherwise, when the value of the advantage function is less than 0 ($A_t < 0$), indicating that this action is harmful, and

the update ratio is less than $1-\epsilon$, the backward would not update this as well. Moreover, ϵ could be a personalized number such as 0.2.

$$L_t^{CLIP}(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t) \quad (5)$$

Thirdly, add a penalty on KL divergence with a fixed or adaptive parameter, β, is introduced by the (**?**) to the objective as shown in (6). In the simplest implementation of this method, SGD could be performed for each iteration to update the policy model to maximize the objective (6). After this, the β should be updated after the policy updated if adaptive parameter is used. Our work on poem generation makes use of an adaptive parameter to control the KL divergence and details will be presented in the following section.

$$L_t^{KL}(\theta) = r_t(\theta)\hat{A}_t - \beta\text{KL}\left[\pi_{\theta_{\text{old}}}, \pi_\theta\right] \quad (6)$$

3.2.1 Objective

As indicated in the formula (7), when integrating the PPO method into our poem generation task, our objective, total loss, could be considered as the sum of policy loss, $\mathcal{L}_\theta^{PL}$, and the critic loss, $\mathcal{L}_\theta^{CL}$. Since the critic is on the top of the policy, the parameters for the critic and policy models are same, written as θ. In addition, the value loss is controlled by a predefined coefficient, α, that determines its weight.

$$\mathcal{L}_\theta = \mathcal{L}_\theta^{PL} + \alpha\mathcal{L}_\theta^{CL} \quad (7)$$

Use the clipped surrogate objective in (5), we could denote the policy loss for our poem generation model with the ratio defined in (1):

$$\mathcal{L}_\theta^{PL} = E[\sum_{t=0}^{T} \min(r_t(\theta)\hat{A}_\pi^t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_\pi^t)] \quad (8)$$

The way we calculate the critic loss as shown in on language model as shown in 9 shown is quite similar to (Shojaee et al., 2023). Assume V_π to be the critic function that evaluates the current action based on s_t, the previously generated tokens.

$$\mathcal{L}_\theta^{CL} = E[\sum_{t=0}^{T}(V_\pi\left(s_t\right) - \left(\hat{A}_\pi^t + V_{\pi_{old}}\left(s_t\right)\right))^2] \quad (9)$$

3.3 Design of Reward

Our reward function (10) consists of two components: a predefined score and a KL divergence between the current policy model and the reference model, which is the original pre-trained poem generation model.

$$R = r - \beta\text{KL}\left[\pi_{\theta_{\text{old}}}, \pi_\theta\right] \quad (10)$$

β is an adaptive parameter and the equation should be used to update in our training method (Ziegler et al., 2019)

$$e_t = \text{clip}(\frac{\text{KL}\left[\pi_{\theta_{\text{old}}}, \pi_\theta\right] - \text{KL}_{\text{target}}}{\text{KL}_{\text{target}}}, -0.2, 0.2) \quad (11)$$

Then we could update the β through (12):

$$\beta_{t+1} = \beta_t\left(1 + e_t\right) \quad (12)$$

After the entire poem, which correlates to one episode, is generated, we proposed two types of rewards as represented in the figure 1. We use the sparse reward, which means that a reward is only given when a completed poem is generated. In this scenario, only the last token will receive the reward value and all the others will receive nothing. The first approach is to utilize only the sentiment score output. We evaluate the generated poem (a complete trajectory with a set of tokens) using the BERT base model fine-tuned on a dataset of poem sentiments (Devlin et al., 2018). Sentiment output consists of one of the following types: positive, negative, no impact, or mixed, as well as a probability that this poem belongs to this group. Assuming p is the probability, we can write the criterion for the reward as shown in (13).

$$r_s = \begin{cases} p & \text{type} = \text{positive} \\ 0 & \text{type} = \text{no\_impact or mixed} \\ -p & \text{type} = \text{negative} \end{cases} \quad (13)$$

The second strategy is to combine the sentiment score with a bertscore, which measures the distance between the poem generated by the active policy model and the poem generated by the original model, as shown in 14. After each epoch, the generated poems by the fine-tuned model will be compared to those generated by the original GPT2 model, and the similarity (distance) will be factored into the reward function. Using only the sentiment

Emotion	Number of Poems	Positive
anger	46	no
courage	77	yes
fear	31	no
hate	23	no
joy	128	yes
love	161	yes
peace	80	yes
sad	154	no
surprise	16	yes

Table 1: PERC Dataset

score may result in an unfavourable outcome, as the policy model may produce only positive vocabulary and lose its original fluency. The application of bertscore could then prevent future updates from deviating too significantly from the initial model. σ is a hyperparameter that controls the bertscore weight. r_{bert} is the bertscore.

$$r = r_s + \sigma r_{bert} \tag{14}$$

4 Experiments

4.1 Training

The training process involved the use of a dataset *Poem Emotion Recognition Corpus* (P. S. and Mahalakshmi, 2019). The PERC dataset is comprised of 9 different emotion labels, with 5 of them being positive. The details of the dataset is shown in Table 1. All the models will then be trained based on these positive poems.

To observe and evaluate the performance of reinforcement learning in the language model, reinforcement learning models will be trained on a variety of hyper-parameters. Table 2 depicts the hyper-parameter sets we employed. The first different settings is the reward function, which will differ in the percentage of the similarity reward. The reason we did not increase the similarity reward further is to obtain the sentiment score, which is our model's primary objective. Another difference is how training prompts are generated. When generating prompts, it will select five positive poems at random from the dataset and use the first five words as prompts. However, for some models, the prompts are generated prior to training and remain constant throughout training, while others generate new prompts during training epochs. These models have been trained for 500 epochs, and the results will be discussed in Section 5.1.

To provide a comparison, we fine-tuned a baseline model using the same pre-trained GPT2 model and the same dataset as the RL method. The GPT2 was fine-tuned using a causal language modeling (CLM) loss to predict the next token with the highest probability. The outputs were then sampled from the available next token candidates. The training and evaluation data used for fine-tuning consisted of samples with positive labels. The model was trained on the PERC dataset and the maximum generation length was set to 100. The generated outputs were then compared to those of the RL method using various evaluation metrics.

4.2 Evaluation

To evaluate the result, each model will be generating 5000 poems providing the same prompt and these poems will be assessed using the following metrics (Wang and Wan, 2018):

- *Sentiment Accuracy*: To evaluate the sentiment accuracy of our poem generation, we utilized a sentiment classifier trained on a Twitter corpus that categorizes sentiments into positive, negative, and neutral (Pak and Paroubek, 2010). The classifier outputs a soft-max score indicating the probability of each sentiment label. We considered the score corresponding to the positive sentiment label as the sentiment accuracy score for our generated poems. This score is different from the reward function model to make a fair comparison between the RL models and the Baseline models.

- *Fluency*: We model the fluency score in correlation to grammatical error detection (Atwell, 1987; Wagner et al., 2007; Schmaltz et al., 2016). As there is a limited amount of research focused on reference-free fluency evaluation, we utilized a language modelling toolkit to determine the number of grammatical errors for phrases of equal length. Specifically, we assessed 100-word generated poems starting with the same prompt, and based the fluency evaluation metric on the grammatical error rate.

- *Diversity*: Our goal was to determine whether the fine-tuned model was capable of producing diverse outputs for a given prompt, rather than solely focusing on generating a fixed sentence to attain high sentiment accuracy. To evaluate this, we tested the cosine similarity

Index	Reward Function	Generating Prompts	Output Length
1	Sentiment	Within Training	128
2	Sentiment	Predefined	128
3	0.9 Sentiment + 0.1 Similarity	Within Training	128
4	0.7 Sentiment + 0.3 Similarity	Within Training	128
5	0.5 Sentiment + 0.5 Similarity	Within Training	128
6	0.3 Sentiment + 0.7 Similarity	Within Training	128
7	0.9 Sentiment + 0.1 Similarity	Predefined	128
8	0.7 Sentiment + 0.3 Similarity	Predefined	128
9	0.5 Sentiment + 0.5 Similarity	Predefined	128
10	0.3 Sentiment + 0.7 Similarity	Predefined	128

Table 2: set of hyper-parameters we used to train the model

(C) of text embeddings for each pair of generated poems. Given that the cosine similarity score ranges from -1 to 1, where higher values indicate greater similarity, we computed the score as $\frac{1-C}{2}$.

- Novelty: Apart from assessing diversity of the generated outputs, we also aimed to evaluate the generalization ability of the model beyond the training data. To achieve this, we randomly selected a sample from the training data and extracted the prompt, then computed the novelty score using the same metric as diversity for each generated sentence from the prompt and the selected training data.

5 Results & Discussion

5.1 RL results

To My Fairy Fancies NAY,
And I will love her still,
More and more till I die.If there is pleasure in sorrow,
And truly bad things for which we blame
And we feel such pain for them,
Not in tears, in mirth and tears,
Could that be the worst?
For woes of body and soul,
That through our blood flow not no tears

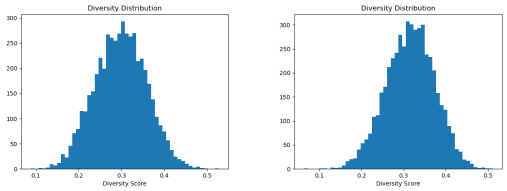
This is one of the poems generated by what we believe to be the relative optimal model, which is trained with 0.5 Sentiment + 0.5 Similarity as a reward and generates prompts prior to training. As shown in Table 3, it has a respectable score across the four metrics. Although it is not the best in each of the metrics along, it is a more balanced choice and the rationale for its selection and some intriguing findings will be discussed in the subsequent sections.

5.1.1 Repetition

Repetition is an interesting finding from our experiment. First observed in the model with sentiment score as the only reward function. Here is a sample of the poem that it generated.

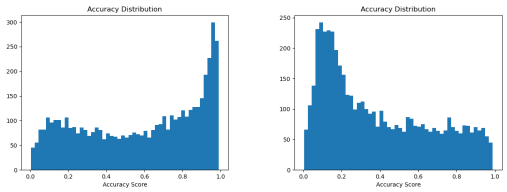
To My Fairy Fancies NAY,
That this blind crew should come to light.
They could wait until after dawn.
My Love.
The fairest thing in the lands,
The brightest star in the east,
The clearest tree in the west,
My Love.
My Love.
My Love.
My Love.

It is evident that it has learned that *My Love* is the keyword to achieve a high result in the sentiment model, as poems like this one have repeatedly resulted in a low diversity in the results. *My Love* occurred over 4000 times in this 5000 poems and another *I am Happy* as another sweet trick occurred around 2000 times. This is where we introduced the new reward where we are comparing the generated sentence of the original model and our fine-tuned model so that we expect it to generate a sentence with a positive sentiment that is similar to its original sentence. And the outcome is straightforward, as illustrated in Figure 2. As we decreased this kind of repetition in the generated poems, the model's diversity grows as the percentage of similarity rises.



(a) Sentiment Only (b) Sentiment + Similarity

Figure 2: Diversity result of different reward function



(a) Sentiment Only (b) Sentiment + Similarity

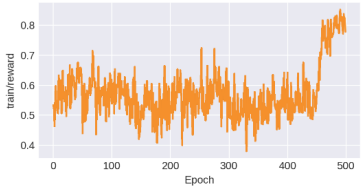
Figure 3: Sentiment Accuracy result of different reward function

5.1.2 Sentiment Accuracy

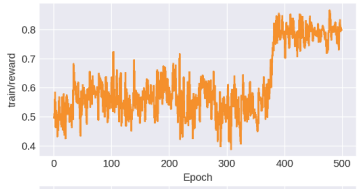
Adding the similarity and increasing its percentage will, however, continue to have a negative effect that will decrease the accuracy of the sentiment. It is an obvious hypothesis, and the results of the experiment showed in Figure 3 that using a high percentage (0.7) of similarity in the reward function significantly decreased the number of positive poems generated. This proved that the hypothesis was correct. As a result, it is absolutely necessary to strike a balance to achieve the accuracy of our model while avoiding its repetition.

5.1.3 Convergence

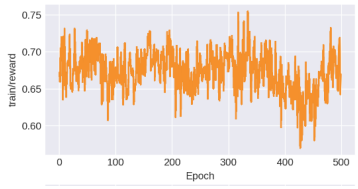
Another significant effect that needs to be carefully taken into account is the time and epochs required for convergence. The percentage of similarity and whether prompts are generated during training are two important hyper-parameters that influence the model's convergence, as shown in Figure 4. The convergence will occur sooner if the result is generated prior to training, as there will be fewer states for the agent to explore. Conversely, if the result is generated during training, there will be a larger space of states for the agent to explore. Meanwhile, we did not notice a discernible decline in the result's diversity when using the fixed prompts during training. The degree of similarity has another effect. Due to the generative model's randomness, it is naturally harder for the model to converge. Even two identical models will not produce the same result, so similarity will always have at least a moderately



(a) 0.7 Sentiment + 0.3 Similarity and prompts in training



(b) 0.7 Sentiment + 0.3 Similarity and prompts before training



(c) 0.3 Sentiment + 0.7 Similarity and prompts before training

Figure 4: Reward after each epoch of different settings

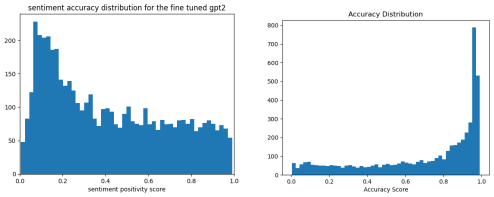
high value and should behave more like a penalty that deducts a smaller portion of the reward than like a main reward objective.

5.2 Compare RL method and CLM

Using the CLM loss, the baseline model is fine-tuned to produce an error-free, highly readable poem with a reasonable outcome. However, as shown in Table 3, the sentiment accuracy is one of the key differences between RL and CLM. As shown in Figure 5, although the baseline model is also fine-tuned in the positive datasets, it receives a lower score (0.410) than the RL model (0.701). RL models provide a more stable and well-performing outcome, while generating primarily positive poems with only a small amount of other sentiments. However, the baseline demonstrates a tendency not to generate positive poems and has a relatively more uniform distribution across various sentiment levels. This demonstrates that RL methods are highly capable of fine-tuning a pre-trained model towards a given objective. Similar observations are made in regard to the novelty of the result, and the RL model demonstrates a great potential to produce a more innovative result rather than being limited by the dataset. Although RL models do not

Model	RL	Baseline
Sentiment	0.701	0.410
Fluency	0.753	0.801
Diversity	0.319	0.301
Novelty	0.370	0.359

Table 3: The mean evaluation scores for RL and CLM



(a) Baseline (CLM) (b) RL Model

Figure 5: Sentiment Accuracy result

demonstrate a significant improvement in diversity and have demonstrated a disadvantage in fluency, we may still conclude that they are superior solutions.

To formally prove our idea, We conducted a student t-test to determine the significance of the differences observed between the two models. The results are presented in Table 4. It is clear from the table that the RL model performs considerably better than the baseline in terms of sentiment and novelty metrics, with improvements that are statistically significant. Despite the fact that the fluency metric suffers as a result of using the RL model, the diversity metric still demonstrates some degree of improvement nonetheless.

6 Conclusion

Using reinforcement learning, the GPT-2-based poetry generation model is trained to generate positive poems with varying reward and hyperparameter settings. We employ a clipped surrogate objective and an actor-critic style of proximal policy optimisation. Comparing a variety of settings, we are able to conclude a relatively optimal model and achieve a balanced and effective outcome. In addition, we train a positive poem-generating model using the conventional NLP technique CLM as a comparison to the RL model and conclude that RL is superior to the conventional method of fine-tuning models using a specific reward in terms of not only the fine tuning objective but also diversity and novelty.

Metric	T-statistic	p-value
Sentiment	48.9	0
Fluency	-7.73	10^{-14}
Diversity	2.22	0.026
Novelty	51.4	0

Table 4: The mean evaluation scores for RL and CLM

References

Suhaib Aslam and Victor Batenburg. Poetrymirror: An affective, poetic reflection of you through natural language generation.

Eric Steven Atwell. 1987. How to detect grammatical errors in a text without parsing it. In *Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805 [cs].

Nils Erik Enkvist. 1973. *Linguistic stylistics*. De Gruyter.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR.

Hugo Gonçalo Oliveira. 2017. A Survey on Intelligent Poetry Generation: Languages, Features, Techniques, Reutilisation and Evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20, Santiago de Compostela, Spain. Association for Computational Linguistics.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward Controlled Generation of Text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596. PMLR. ISSN: 2640-3498.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Sreeja P. S. and G. S. Mahalakshmi. 2019. Perc-an emotion recognition corpus for cognitive poems. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0200–0207.

Daniel S Paiva and Roger Evans. 2004. A framework for stylistically controlled generation. In *Natural Language Generation: Third International Conference, INLG 2004, Brockenhurst, UK, July 14-16, 2004. Proceedings*, pages 120–129. Springer.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Rosalind W Picard. 2000. *Affective computing*. MIT press.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Gavin A Rummery and Mahesan Niranjan. 1994. *Online Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart Shieber. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 242–251, San Diego, CA. Association for Computational Linguistics.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K Reddy. 2023. Execution-based code generation using deep reinforcement learning. *arXiv preprint arXiv:2301.13816*.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Joachim Wagner, Jennifer Foster, and Josef van Genabith. 2007. A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 112–121, Prague, Czech Republic. Association for Computational Linguistics.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *International Joint Conference on Artificial Intelligence*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8:279–292.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32.

Bei Xu and Ziyuan Zhuang. 2022. Survey on psychotherapy chatbots. *Concurrency and Computation: Practice and Experience*, 34(7):e6170.

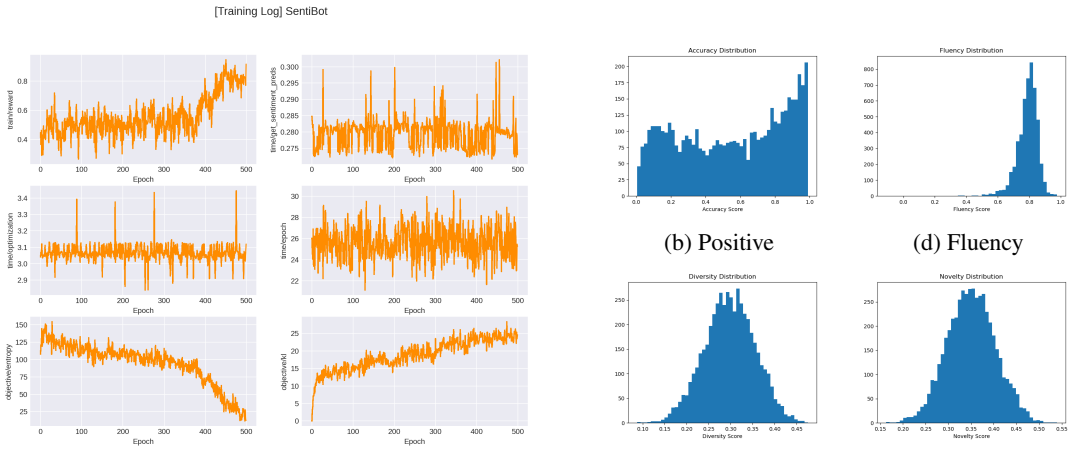
Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Appendix

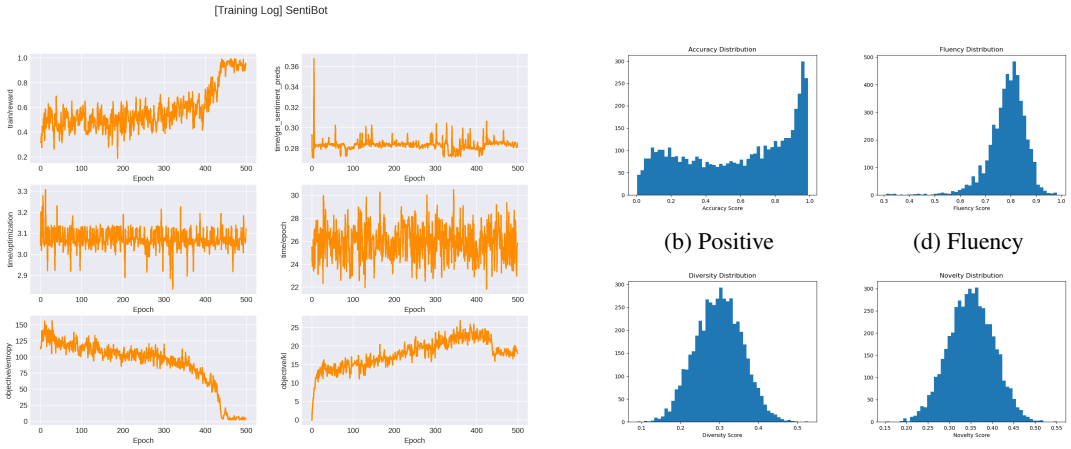
A Experiment Result

The complete results of our experiments are listed in the exact same order as in Table 2. These figures include the training logs, beginning with the rewards log, and our evaluation results, which include the positive sentiment score, the fluency, the diversity, and novelty of the model-generated poems.



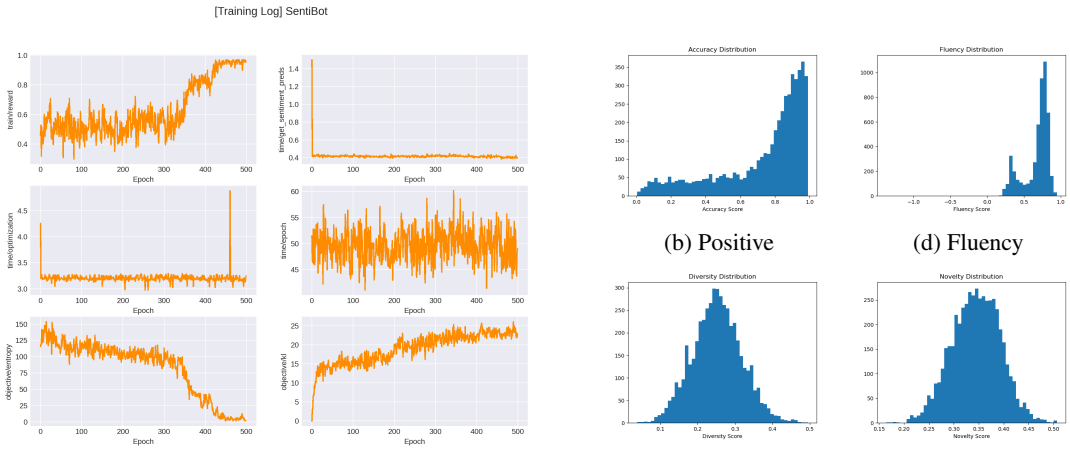
(a) Training Log (b) Positive (c) Diversity (d) Fluency (e) Novelty

Figure 6: Only Sentiment as Rewards, Generating Prompts While Training



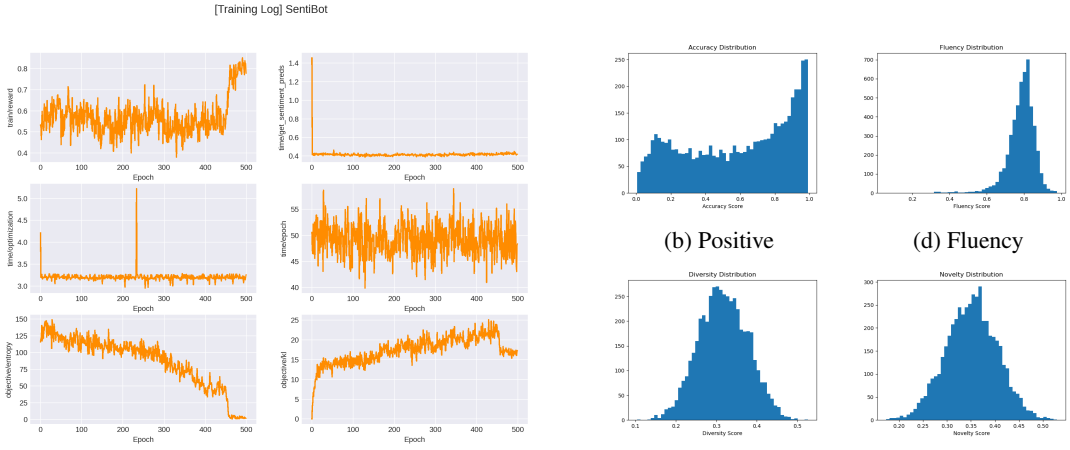
(a) Training Log (b) Positive (c) Diversity (d) Fluency (e) Novelty

Figure 7: Only Sentiment as Rewards, Generating Prompts Before Training



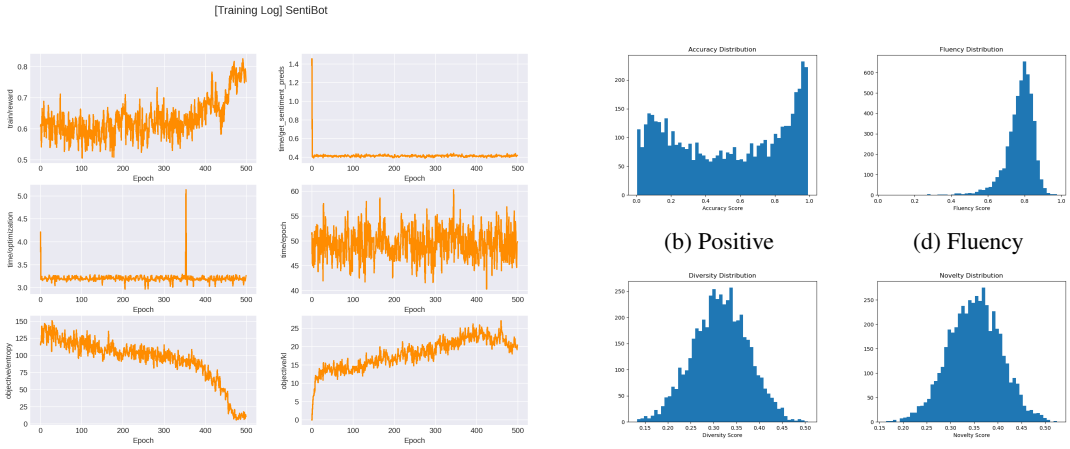
(a) Training Log (b) Positive (d) Fluency (c) Diversity (e) Novelty

Figure 8: 0.9 Sentiment + 0.1 Similarity as Rewards, Generating Prompts While Training



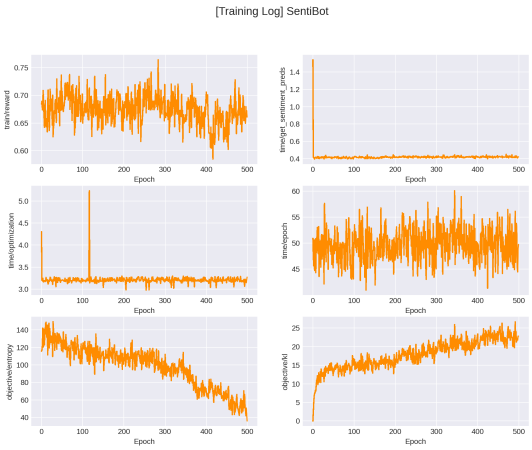
(a) Training Log (b) Positive (d) Fluency (c) Diversity (e) Novelty

Figure 9: 0.7 Sentiment + 0.3 Similarity as Rewards, Generating Prompts While Training

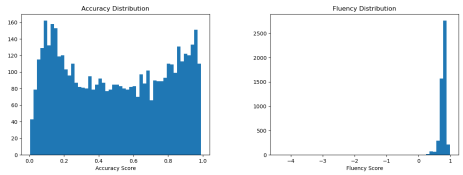


(a) Training Log (b) Positive (d) Fluency (c) Diversity (e) Novelty

Figure 10: 0.5 Sentiment + 0.5 Similarity as Rewards, Generating Prompts While Training

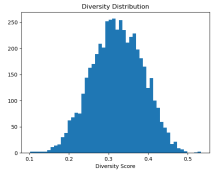


(a) Training Log

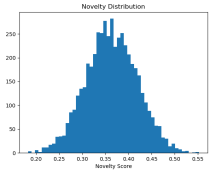


(b) Positive

(d) Fluency

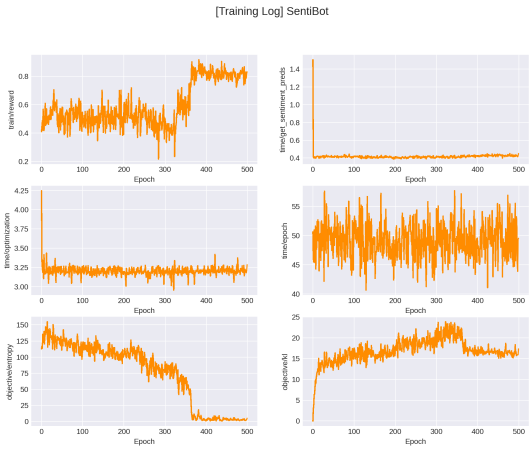


(c) Diversity

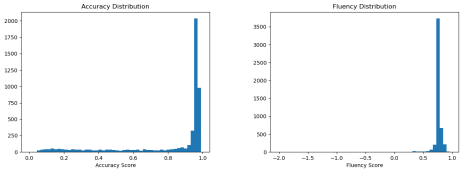


(e) Novelty

Figure 11: 0.3 Sentiment + 0.7 Similarity as Rewards, Generating Prompts While Training

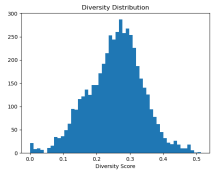


(a) Training Log

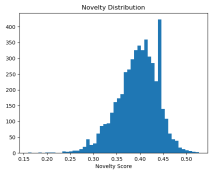


(b) Positive

(d) Fluency

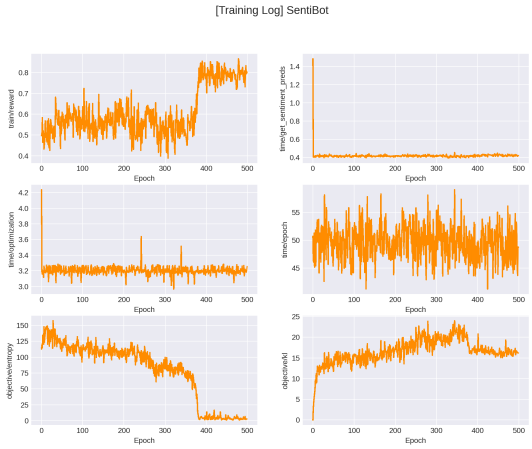


(c) Diversity

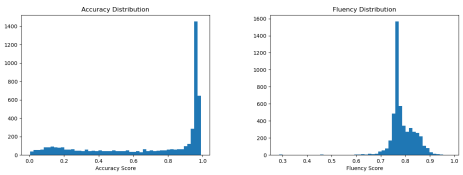


(e) Novelty

Figure 12: 0.9 Sentiment + 0.1 Similarity as Rewards, Generating Prompts Before Training

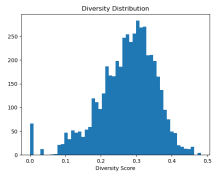


(a) Training Log

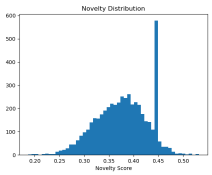


(b) Positive

(d) Fluency

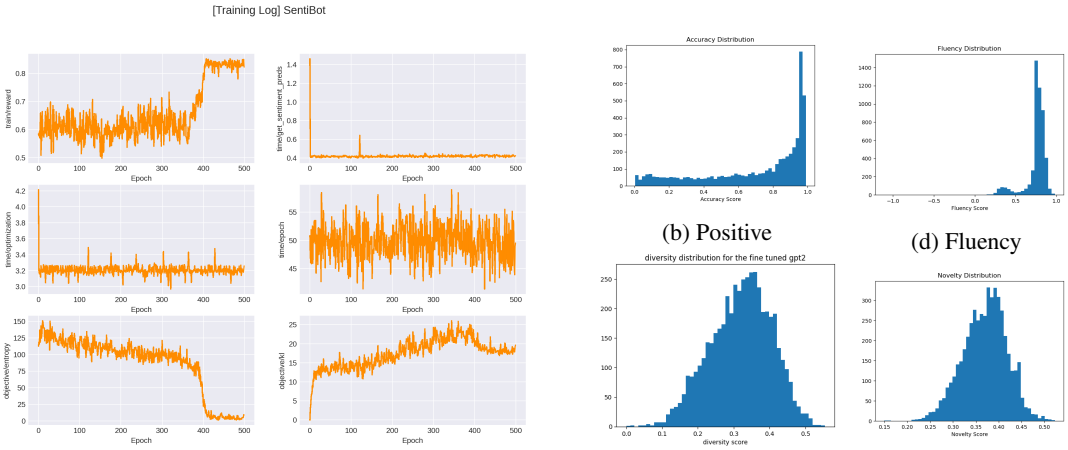


(c) Diversity



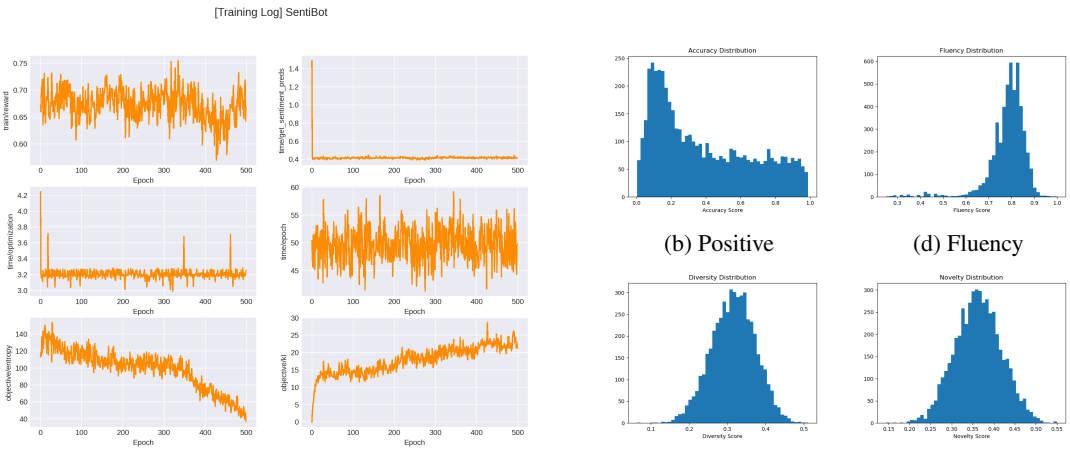
(e) Novelty

Figure 13: 0.7 Sentiment + 0.3 Similarity as Rewards, Generating Prompts Before Training



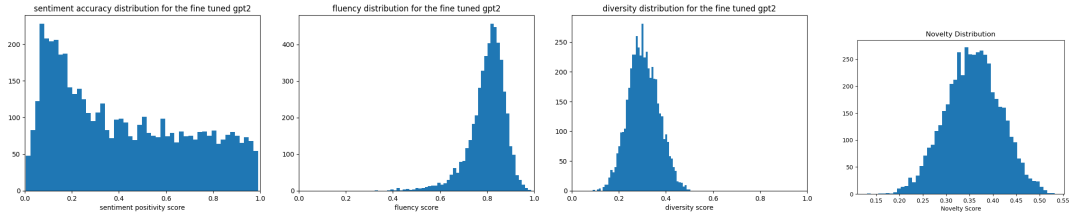
(a) Training Log (b) Positive (c) Diversity (d) Fluency (e) Novelty

Figure 14: 0.5 Sentiment + 0.5 Similarity as Rewards, Generating Prompts Before Training



(a) Training Log (b) Positive (c) Diversity (d) Fluency (e) Novelty

Figure 15: 0.3 Sentiment + 0.7 Similarity as Rewards, Generating Prompts Before Training



(a) sentiment accuracy score (b) fluency score (c) diversity score (d) novelty score

Figure 16: Fine tuned GPT (Baseline) Result