Is Crowdsourcing Breaking Your Bank? Cost-Effective Fine-Tuning of Pre-trained Language Models with Proximal Policy Optimization

Shuo Yang, Gjergji Kasneci
Technical University of Munich
Richard-Wager-Straße 1
shuo.yang@tum.de, gjergji.kasneci@tum.de

Abstract

Wide usage of ChatGPT has highlighted the potential of reinforcement learning from human feedback. However, its training pipeline relies on manual ranking, a resource-intensive process. To reduce labor costs, we propose a self-supervised text ranking approach for applying Proximal-Policy-Optimization to fine-tune language models while eliminating the need for human annotators. Our method begins with probabilistic sampling to encourage a language model to generate diverse responses for each input. We then employ TextRank and ISODATA algorithms to rank and cluster these responses based on their semantics. Subsequently, we construct a reward model to learn the rank and optimize our generative policy. Our experimental results, conducted using two language models on three tasks, demonstrate that the models trained by our method considerably outperform baselines regarding BLEU, GLEU, and METEOR scores. Furthermore, our manual evaluation shows that our ranking results exhibit a remarkably high consistency with that of humans. This research significantly reduces training costs of proximal policy-guided models and demonstrates the potential for self-correction of language models.

Keywords: Natural Language Processing, Proximal Policy Optimization, Pre-trained Language Model, Reinforcement Learning, Self-supervised Learning

1. Introduction

With the advancement of natural language processing, contemporary pre-trained language models (PLMs) have demonstrated significant commercial value due to their widespread adoption in sectors such as education, healthcare, and finance (Edunov et al., 2019; Sun et al., 2022). However, due to the shortcut learning (Geirhos et al., 2020), degeneration (Holtzman et al., 2020), and other complicated reasons, PLMs often generate topic-irrelevant or unhelpful information, resulting in a loss of resources and reliability (Weidinger et al., 2021). As an existing optimization, models trained through reinforcement learning from human feedback (RLHF) (Wang et al., 2022) are continually supervised by human-ranked data during the training. Therefore, they demonstrate higher performance and reliability across diverse tasks such as dialogue, question-answering, and machine reading comprehension.

Although RLHF has been widely proven to be effective in improving the quality of generative models (Lin et al., 2020a), there are three limitations of applying RLHF: 1) training costs of large-scale PLMs, 2) lack of high-quality prompts for varying user intents (Bodonhelyi et al., 2024) and 3) labor costs associated with crowdsourcing.

Both of the first two limitations have been addressed with diverse solutions. Regarding the first limitation, contemporary lightweight PLMs such as

Stanford Alpaca and ChatGLM (Taori et al., 2023; Zeng et al., 2023) have achieved performance comparable to traditional large-scale PLMs with a hundred billion parameter, substantially reducing training costs. For the second limitation, prompt generation methods, e.g., self-instruction (Wang et al., 2022), have presented solutions for automatically building instruction sets. However, for the last point, there needs to be more focus on exploring the utilization of self-supervised learning as a viable alternative to annotations on crowdsourcing platforms to address the challenge of substantial manual costs.

To achieve this, we propose a Self-supervised Text Ranking (STR) pipeline, simulating the generation of human-ranked data. We derive our theoretical and empirical foundations from two articles: Chen et al. (2023) demonstrated that language models can enhance generation quality through self-checking and correction. Li et al. (2023) established the effectiveness of ensemble learning in assessing the rationality of various interpretations produced by a PLM when it employs different reasoning pathways to address the same question. Building upon these two contributions, we apply the proximal policy optimization (PPO) through the STR pipeline to enable language models to self-assess and self-supervise during fine-tuning (Schulman et al., 2017) as shown in Figure 1, with the following three steps:

1. **Ensemble learning-based text ranking.** We follow the RLHF baseline, generating diverse answers for each question via a PLM. After that, we apply a semantic similarity-based TextRank algorithm (Mihalcea and Tarau, 2004; Zhang and Wang, 2021) to rank generated answers, distinguishing our work from previous efforts. We root our motivation in the theoretical assumption that if a PLM generates different answers to a given question, the semantics among reasonable answers should exhibit a stronger clustering tendency than irrational ones. This is because incorrect or unhelpful statements hallucinated by PLMs always involve various unrelated topics (Zhang et al., 2023).

2. **Extraction of representative answers.** We then cluster answers via the Iterative Self-Organizing Data Analysis Technique Algorithm (ISODATA) and extract cluster centers to build answer pairs (Ball and Hall, 1967). The advantage of clustering is the reduced computational overhead and the avoidance of comparing semantically similar sentences. These constructed answer pairs will be utilized to train a reward model for assessing the quality of an answer.

3. **To update the generation policy.** We finally learn a reward model from the answer pairs. The reward model is then used to update our generative policy, which generates answers for evaluation.

Our contributions are as follows:

- We propose a novel self-supervised text ranking method for simulating manual ranking in RLHF while eliminating human labor costs in fine-tuning PLMs.

- Our experimental results demonstrate that the proposed method significantly outperforms other fine-tuning approaches for two PLMs on three datasets.

- Our manual evaluation experiments demonstrate that our approach can considerably substitute human annotators for generating training data for future PPO-guided PLMs.

2. Related Work

2.1. Pre-trained language models

Transformer-based PLMs (Vaswani et al., 2017; Lewis et al., 2020) have been widely used in assorted downstream tasks due to their versatility and excellent semantic extraction capabilities. Among them, ERNIE (Zhang et al., 2019) incorporates knowledge from knowledge graphs to improve representation learning for NLU tasks. T5 (Ni et al., 2022) achieved transforming various NLP tasks into text-to-text transfer problems. This paper uses the GPT-2 and GPT-Neo (Radford et al., 2019; Black et al., 2021) due to the extensive data sources they used in pre-training and their representative architectures stacked by attention layers.

Before inference, researchers often conduct fine-tuning by retraining particular parameters of a PLM on downstream datasets to adapt and optimize it for specific tasks (Pfeiffer et al., 2020). A fundamental fine-tuning approach involves training all the parameters with smaller learning rates. However, Lin et al. (2020b) demonstrated that by adding and fine-tuning additional 2-3% parameters, PLMs could maintain a similar performance of the full fine-tuning. Furthermore, Ben Zaken et al. (2022) proposed a sparse-fine-tuning method where only the bias terms are being modified. (Hu et al., 2022) injected rank decomposition matrices into the Transformer architecture, reducing the trainable parameters for downstream tasks. However, these methods focus on optimizing model architecture or the size of trainable parameters while neglecting the effect of fine-tuning in enhancing the model's ability to self-correction.

2.2. Policy gradient

As a branch of reinforcement learning (RL) (Minsky, 1961), policy gradient algorithms (Sutton et al., 1999) have been widely applied to NLP tasks, such as addressing the issue of gradient unavailability (Yu et al., 2017) in Generative Adversarial Networks (Goodfellow et al., 2020). As an off-policy improvement, Schulman et al. (2017) propose PPO, which computes the similarity between the generative and sampling policies as part of the objective function during training.

In the InstruchGPT (Ouyang et al., 2022), a proxy model with the same architecture as the training PLM is employed to explore generation strategies for answers. However, updating the policy requires human feedback, significantly increasing the training cost. In this paper, we follow this work and present a cost-efficient solution.

2.3. Text ranking

Contemporary unsupervised text ranking methods rely on statistics, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang* et al., 2020), compute overlaps of n-grams or semantic information to rank answers based on their similarity to questions. However, while answers similar to the question are topic-relevant, their helpfulness to humans cannot be guaranteed.
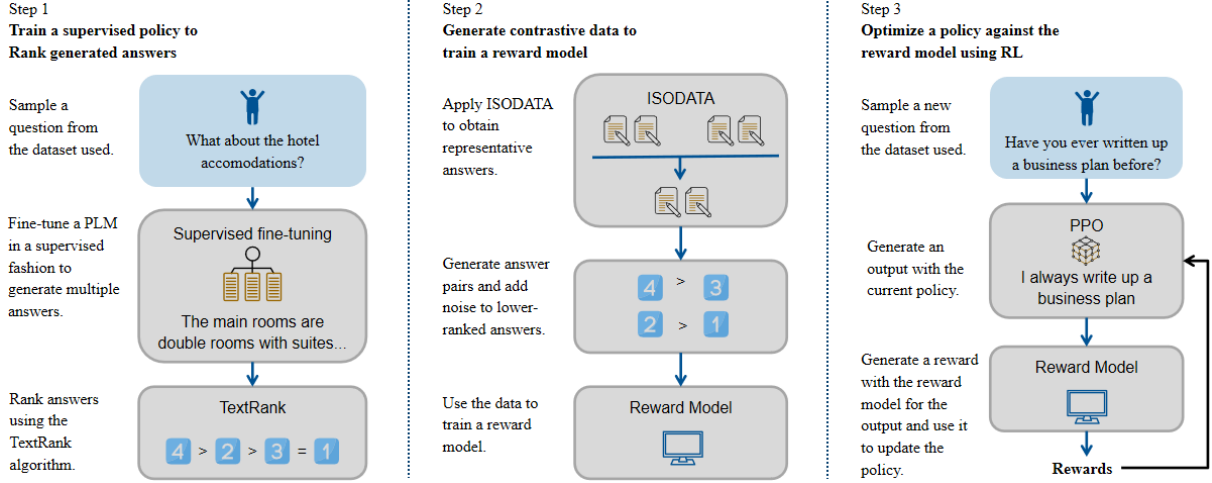
Figure 1: Our pipeline comprises three steps: 1) fine-tuning a language model to generate multiple candidate answers for a given question and using the TextRank algorithm to rank these answers; 2) filtering out non-representative answers using the ISODATA algorithm and training a reward model based on the remaining answers, and 3) scoring the ranked answers using the reward model and updating the generation policy via PPO. Note that we implemented the generative policy through a PLM in our study.

3. Methodology

3.1. Problem formulation

We aim to automatically rank text by leveraging the semantic information widely learned by PLMs during pre-training, thus implementing the PPO-guided training under unsupervised conditions to further fine-tune PLMs. Our study considers various natural language processing tasks as question-answering.

Formally, given a parallel dataset consisting of a prompt P, a set of questions $Q = (q_1, ..., q_m)$, and their corresponding answers $A = (a_1, ..., a_m)$, we are going to fine-tune a question-answering PLM LM_ϕ and enhance its performance via PPO achieved by using ranked text.

3.2. How to generate diverse answers?

We initially follow the training paradigm of RLHF, wherein we fine-tune a PLM to produce distinct answers for a given question (Ouyang et al., 2022). Specifically, to generate tokens for answering, we compute logits for each input token sequence $w_1, ..., w_{(i-1)}$ using LM_ϕ. The probability of generating the i'th token w_i as $w \in V$ is then given by $P_\phi(w|w_{1:i})$, where V represents the vocabulary. Subsequently, we apply a temperature function (Ackley et al., 1985) with a small τ, to shape the probability distribution:

$$P'_\phi(w_i) = \frac{P_\phi(w|w_{1:i})^{1/\tau}}{\sum_{w' \in V} P_\phi(w'|w_{1:i})^{1/\tau}}. \quad (1)$$

To encourage the LM_ϕ to generate answers via diverse expression while mitigating potential text degeneration, we apply a top-p sampling with a high

p to conduct a subset of V and sample token w_i according to probability distribution P''_ϕ:

$$P''_\phi(w_i) = \begin{cases} P'_\phi(w_i)/\sum_{w' \in V_p} P'_\phi(w') & \text{if } w_i \in V_p \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where top-p vocabulary V_p is the smallest set such that: $\sum_{w_i \in v_p} P(w_i|w_{1:i-1}) \geq p$.

By repeatedly sampling from $P''_\phi(w_i)$, we generate multiple answers for each question in the dataset.

3.3. How to rank generated answers?

For a specific question, we assume that the semantic similarity among different answers can reflect the correctness of these answers in theory. Specifically, high-quality answers should exhibit more similarity in the hidden space than others. For example, given a question "*What does 'apple' mean?*", possible answers include "*a kind of red fruit*", "*a fruit rich in vitamins*", "*a US-based business*", and "*a company involved in electronic production*". Although there may not be a unique correct answer, in this case, the first two answers relate to "apples", while the last two refer to "Apple Inc.". Therefore, in theory, we could recognize their corresponding clusters in semantic space. Conversely, incorrect or irrelevant answers, such as "*a toxic substance.*" or "*I am full.*", would be more dispersed in the semantic space as the hallucinations (Alkaissi and McFarlane, 2023) always involve various unrelated topics or concepts (Azamfirei et al., 2023). Therefore, we score and rank answers by quantifying their relative positions in the semantic space.

To achieve that, we regard the generated an-

swers as nodes and the similarity between these answers as the weights of edges to construct a graph. In such a graph, we use the TextRank algorithm (Mihalcea and Tarau, 2004) to calculate the weights of the nodes to reflect their ranks. Specifically, when computing the weight of an edge connecting two nodes n_i and n_j, we utilize a Sentence-BERT (Reimers and Gurevych, 2019) to embed their corresponding answers a_i and a_j to vectors, calculating the cosine similarity between the two vectors, as shown in eq. (3):

$$S(n_i, n_j) = \frac{\text{SBERT}(a_i) \cdot \text{SBERT}(a_j)}{||\text{SBERT}(a_i)|| \, ||\text{SBERT}(a_j)||}, \quad (3)$$

where $\text{SBERT}(\cdot)$ is the embedding function of the Sentence-BERT. Subsequently, we calculated the weight of a node n_i as follows:

$$W(n_i) = \sum_{n_j \in \mathsf{I}(n_j)} \frac{S(n_i, n_j)}{\sum_{n_k \in \mathsf{O}(n_j)} S(n_j, n_k)} W(n_j), \quad (4)$$

where $\mathsf{I}(n_i)$ represents the set of nodes that point to node n_i, and $\mathsf{O}(n_j)$ represents the set of nodes that node n_j points to.
Following previous study (Page et al., 1998), we incorporate an empirical damping factor $d = 0.85$ into the formulation to ensure the stability and convergence of the algorithm, that is:

$$W'(n_i) = (1 - d) + d * W(n_i). \quad (5)$$

The overall ranking algorithm is shown as Algorithm 1.

Algorithm 1 TextRank Algorithm

Require:
　　A set of m nodes $(n_1, ..., n_m)$ corresponding to answers $(a_1, ..., a_m)$;
Ensure:
　　Ranking of answers $[a_1, ..., a_m]$;
 1: Initialize all the node weights $W(n)$ to 1.0;
 2: **for** each $i \in [1, m-1]$ **do**
 3: 　　**for** each $j \in [i+1, m]$ **do**
 4: 　　　Calculate the similarity between n_i and n_j according to Eq. (3);
 5: 　　**end for**
 6: **end for**
 7: **for** each $i \in [1, m]$ **do**
 8: 　　Calculate the weight of node n_i using Eq. (5);
 9: **end for**
10: Rank the nodes in descending order according to their weights.

3.4. How to construct contrasting data?

After we rank all the answers, we only extract a subset from them to train a reward model. Our motivation is that the PLM may generate duplicate or similar outputs. For instance:
Question: *How to improve concentration?*
Answer A: *Minimize distractions, use focus techniques, and manage time effectively.*
Answer B: *Avoid interruptions, apply concentration methods, and utilize time management skills.*
In this example, learning the relative ranking of these two answers is irrational as they convey the same meaning. Therefore, we first cluster answers by minimizing the semantic distance within the clusters. Then, we retain only one representative answer within each cluster.

However, it is difficult to pre-determine the optimal number of clusters in real-world cases. For instance, in multiple-choice questions, the optimal number of clusters is expected to be similar to the number of options. On the contrary, open-ended questions may require more clusters to capture the diversity of answers. To solve this problem, we used the ISODATA to select the representative subset. The advantage of employing ISODATA is its ability to adaptively adjust the number of answer clusters by merging or splitting them for different questions, as shown in Algorithm 2. Please note that the effectiveness of TextRank relies on a substantial number of samples. Therefore, we should only filter answers after ranking them.

Algorithm 2 ISODATA Algorithm

Require:
　　A set of answers $A = (a_1, ..., a_m)$;
Ensure:
　　A subset of A: $[a'_1, ..., a'_{m'}]$;
 1: Randomly initialize K cluster $C = (c_1, ..., c_k)$.
 2: **while** Convergence criteria are not reached **do**
 3: 　　**for** each $i \in [1, m]$ **do**
 4: 　　　Assign a_i to the nearest cluster.
 5: 　　**end for**
 6: 　　Calculate the centroid of each cluster.
 7: 　　**while** the number of samples in the cluster $c_i \leq$ min-threshold, $c_i \in C$ **do**
 8: 　　　Merge c_i and c_j, where c_j is the closest cluster for c_i.
 9: 　　**end while**
10: 　　**while** the number of samples in the cluster $c_i \geq$ max-threshold, $c_i \in C$ **do**
11: 　　　Calculate the distance between each sample point in c_i and the cluster centroid, split the farthest sample point.
12: 　　**end while**
13: **end while**
14: Extract centroids of all clusters as outputs.

After the clustering, we build high-low-ranked answer pairs to train a reward model. To increase the quality difference between the two answers in an

answer pair, we propose to select them with a fixed interval. Let $A' = (a'1, ..., a'm')$ be a ranked answer set from ISODATA, the i'th answer pair would be $\mathsf{AP}_i = (a'w, a'l)$, where $l \geq w + \mathsf{IL}$, and IL is the interval length, i.e. distances in the ranking list.

Furthermore, we propose noise injection to the low-ranked answers to ensure the correctness of the ranking. Here, we designed three types of noise injection: 1) n-gram level editing operations involve randomly deleting or replacing an n-gram (where n < 4) with another random n-gram or inserting a random n-gram to the answer. 2) Adding or deleting negation words, where we randomly add negation words before verbs and delete them for answers that already contain negation words. 3) We consider randomly shuffling the order of sentences for answers that contain more than one sentence. When a low-ranked answer is identified for noise injection, we randomly select one of the three noise types. Here, the noise injection is employed to enhance the diversity in the quality of the two answers within an answer pair. Moreover, counterexamples generated based on the above three manually defined rules can assist in training the language model to mitigate the severity of corresponding errors, thereby improving the overall generation quality, as demonstrated in the experimental section below.

3.5. Training

After constructing answer pairs, we fine-tune another PLM as the reward model to convert answer ranks into numerical values to represent their reasonableness. In detail, during the training process of the reward model, we maximize the reward difference between the two answers (a_w, a_l) in an answer pair for a question q:

$$R_\theta(a_w, a_l; q) = \sigma(r_\theta(q, a_w) - r_\theta(q, a_l)). \quad (6)$$

The loss function of the reward model θ is:

$$\mathsf{loss}(\theta) = -E_{(q, a_w, a_l) \in D}[\log R_\theta(a_w, a_l; q)], \quad (7)$$

where D is the set of answer pairs.

Regarding the training of our generative policy, we maximizing our objective function, as shown in eq. (8). For a given input x and a model output y, the objective function comprises two components: The score computed by the reward model and the KL divergence between the generative policy π_ϕ^{RL} and a sampling policy π^{SFT}.

$$\begin{aligned} \mathsf{objective}(\phi) = E_{(x,y) \in D_{\pi_\phi^{\mathsf{RL}}}}[R_\theta(x, y) \\ -\beta \log(\pi_\phi^{\mathsf{RL}}(y|x)/\pi^{\mathsf{SFT}}(y|x))], \end{aligned} \quad (8)$$

where the sampling policy π^{SFT} is an original copy of the generative policy.

Through this approach, our generative policy generates answers with high rewards while preserving its original question-answering capabilities (Ouyang et al., 2022).

4. Experiments

To validate the effectiveness of our approach, we conducted experiments on three tasks: dialogue, story generation, and natural language understanding (NLU). To enhance the uniformity of our evaluation, we consider them all as question-answering tasks with different prompts (Qi et al., 2022). Furthermore, we employed three human annotators to rank model outputs on two datasets with open-domain answers, computing the similarity between the results of our annotations and human annotations.

4.1. Data sets used

DailyDialogue is an extensive English conversation dataset that covers various topics and is collected from English learning websites (Li et al., 2017). The conversations are authored by English speakers and showcase a natural language style with high complexity and diversity.

The Cornell Movie-Dialogue Corpus is an English movie dataset collected by Cornell University (Danescu-Niculescu-Mizil and Lee, 2011). The dataset includes dialogues between movie characters, covering the conversations from more than 600 movies. These dialogues cover romantic, sci-fi, and thriller films.

Stanford Question-Answering Dataset v2.0 (SQuAD) is a reading comprehension dataset, which comprises a question set raised by crowdworkers on a variety of Wikipedia articles (Rajpurkar et al., 2016). For each question, the answer is a text segment, also known as a span, obtained from the corresponding reading passage.

4.2. Details

In this paper, we carried out experiments with the GPT-2 with 124 million parameters and GPT-Neo with 125 million parameters. [1]

In terms of the TextRank, we set the maximum iterations to 1,000. For ISODATA, we set the cluster splitting variance threshold to 0.05. As for training, we used a mini-batch size of 16, and the optimizer is AdamW (Loshchilov and Hutter, 2019) with a learning rate of $3e-5$. In addition, we set the value of β to 0.5 in eq. (8). Regarding inference, we controlled the maximum generated length below 100 tokens; the top-p value is 0.95 and the temperature is 0.8.

To ensure the model's generalization, we conducted all dataset fine-tuning in an autoregressive

---

[1]Our code is available at GitHub.

Dataset	DailyDialogue			CornellMovie		
Models	BLEU ↑	GLEU ↑	METEOR ↑	BLEU ↑	GLEU ↑	METEOR ↑
Original	1.00 ± 0.15	2.33 ± 0.23	6.53 ± 0.52	3.25 ± 0.30	6.92 ± 0.27	15.76 ± 0.38
Full fine-tuning	1.99 ± 0.22	4.42 ± 0.39	9.82 ± 0.63	7.75 ± 0.37	12.72 ± 0.34	20.97 ± 0.48
Adapter (Lin et al., 2020b)	1.79 ± 0.18	4.13 ± 0.28	9.79 ± 0.74	7.74 ± 0.36	12.62 ± 0.42	20.92 ± 0.56
BitFit (Ben Zaken et al., 2022)	1.87 ± 0.23	3.96 ± 0.43	9.92 ± 0.93	7.88 ± 0.37	13.02 ± 0.94	22.76 ± 0.53
LoRA (Hu et al., 2022)	2.10 ± 0.41	4.89 ± 0.54	**10.98 ± 1.03**	6.93 ± 0.28	12.02 ± 0.94	20.76 ± 0.33
Ours	2.12 ± 0.33	4.72 ± 0.68	10.52 ± 1.08	8.93 ± 1.05	12.09 ± 1.01	21.58 ± 1.31
Ours (with adding Noise)	**2.40 ± 0.28**	**4.90 ± 0.41**	10.77 ± 0.68	**10.12 ± 1.02**	**14.08 ± 0.96**	**27.05 ± 1.12**

Table 1: The evaluation results with GPT-2 on two data sets (open-domain tasks) with 0.95 confidence level. 'Bitfit' stands for Bias-only fitting. 'LoRA' stands for Low-rank adaptation.

Dataset	DailyDialogue			CornellMovie		
Models	BLEU ↑	GLEU ↑	METEOR ↑	BLEU ↑	GLEU ↑	METEOR ↑
Original	1.21 ± 0.17	3.53 ± 0.40	7.14 ± 0.89	3.75 ± 0.28	7.75 ± 0.56	8.53 ± 0.46
Full fine-tuning	2.03 ± 0.30	5.72 ± 0.48	10.86 ± 0.90	8.38 ± 0.46	13.95 ± 1.04	**25.68 ± 0.84**
Adapter (Lin et al., 2020b)	1.87 ± 0.28	5.33 ± 0.46	9.89 ± 0.66	7.74 ± 0.40	13.26 ± 0.50	21.87 ± 0.74
BitFit (Ben Zaken et al., 2022)	2.12 ± 0.32	5.85 ± 0.62	**11.92 ± 1.21**	8.46 ± 0.46	14.12 ± 1.25	23.64 ± 0.72
LoRA (Hu et al., 2022)	**2.52 ± 0.83**	6.30 ± 0.54	11.89 ± 1.45	8.27 ± 0.36	13.58 ± 1.17	23.52 ± 0.45
Ours	2.26 ± 0.75	6.28 ± 0.96	11.27 ± 1.87	10.46 ± 2.01	14.13 ± 2.53	23.28 ± 3.07
Ours (with adding Noise)	2.33 ± 0.67	**6.86 ± 0.76**	11.78 ± 0.84	**10.96 ± 2.00**	**14.64 ± 2.17**	23.83 ± 2.55

Table 2: The evaluation results with GPT-Neo on two data sets with 0.95 confidence level.

method of standard language models. However, this fine-tuning approach may lead to lower performance in our experiments than those reported in works focused on specific domains.

4.3. Automated evaluation eMetric

Exact Match Score (EM) is used to evaluate the prediction accuracy of a classification model. It refers to the proportion of questions for which the model provides the correct answer.

The Bilingual Evaluation Understud (BLEU) is a metric used to evaluate the quality of generation (Papineni et al., 2002). We used the BLEU-4 metric via NLTK (Bird et al., 2009) to quantitatively assess the similarity between machine outputs and human reference.

GLEU is designed to estimate text fluency solely based on parser outputs (Mutton et al., 2007). The metric was examined by analyzing its correlation with human judgments of text fluency.

METEOR is a metric based on word-level exact matching (Banerjee and Lavie, 2005). It considers factors such as lexical overlap, word order differences, and stem changes between the generated text and the reference text.

4.4. Manual evaluation

We employed three human annotators through the crowdsourcing platform of MolarData to validate our hypothesis. We randomly selected 100 questions from each of the two open-ended question-answering datasets used and constructed four answers for each question. Our evaluation includes 1. the similarity between human and automated ranking and 2. the kappa (Cohen, 1960) consistency coefficient among different annotators.

5. Results and Discussion

5.1. Analysis

Table 1 and Table 2 present the comparative results of our fine-tuning method in the open-domain question-answering task against the baselines. Our two models achieved state-of-the-art (SOTA) results in almost all evaluation metrics. Regarding the GPT-2 model, our method performed similarly to the LoRA algorithm while outperforming the direct fine-tuning and BitFit algorithm by nearly 1 point in three evaluation metrics on the Dialogue dataset. Additionally, on the CornellMovie dataset, our method achieved a significant improvement of almost 30% in BLEU and METEOR. Furthermore, we surpassed the BitFit algorithm by approximately five points in the text fluency metric GLEU. Moreover, we discovered that introducing noise during the training of the reward model led to improved performance. We attribute this enhancement to the additional contrastive information brought by the noise, which prevented the model from generating incorrect answers to some extent. For the GPT-Neo model, our model outperformed the baselines with a relatively modest advantage. This may be attributed to GPT-Neo having more parameters and better performance than GPT-2, resulting in a lower probability of generating incorrect answers and thus benefiting less from our self-correction approach. Furthermore, the GPT-Neo demonstrated superior performance to the GPT-2, which may be attributed to its larger size and pre-training data scale.

Table 3 and Table 4 present the measurement results on the SQuAD dataset. Our models outperformed baselines by approximately two points in the EM metric and slightly surpassed the LoRA al-

Dataset	SQuAD v2.0						
Models	BLEU ↑	GLEU ↑	METEOR ↑	EM ↑	Precision ↑	Recall ↑	F1 ↑
Original	2.70±0.75	4.07±1.01	11.66±2.17	10.40%	0.06%	0.12%	0.08
Full fine-tuning	19.56±2.67	39.75±4.23	38.24±3.75	52.60%	50.64%	51.94%	0.51
Adapter (Lin et al., 2020b)	18.32±1.63	38.77±4.84	38.20±3.26	52.40%	50.76%	51.38%	0.51
BitFit (Ben Zaken et al., 2022)	18.42±1.99	36.27±3.58	38.40±3.77	53.20%	50.82%	51.63%	0.51
LoRA (Hu et al., 2022)	**19.67±1.68**	40.42±3.98	38.24±3.75	54.40%	52.63%	53.12%	0.53
Ours	19.33±2.64	40.00±4.25	38.44±3.76	55.60%	53.87%	52.19%	0.53
Ours (with adding Noise)	19.47±2.61	**41.81±4.28**	**38.45±3.75**	**56.80%**	**54.23%**	**54.61%**	**0.54**

Table 3: The test results with GPT-2 on SQuAD v2.0 (NLU task) with 0.95 confidence level.

Dataset	SQuAD v2.0						
Models	BLEU ↑	GLEU ↑	METEOR ↑	EM ↑	Precision ↑	Recall ↑	F1 ↑
Original	3.95±0.88	7.64±2.12	23.23±3.98	12.20%	0.10%	0.18%	0.12
Full fine-tuning	22.23±4.00	41.76±5.68	40.72±4.25	54.20%	53.82%	52.97%	0.53
Adapter (Lin et al., 2020b)	16.97±2.33	34.98±5.60	38.05±4.62	52.40%	51.66%	52.88%	0.52
BitFit (Ben Zaken et al., 2022)	18.42±2.00	40.46±2.87	39.96±3.53	53.40%	53.20%	50.65%	0.52
LoRA (Hu et al., 2022)	22.58±3.66	**43.82±4.38**	40.24±4.75	55.80%	**54.80%**	55.20%	**0.55**
Ours	22.16±3.46	42.05±4.88	**41.04±3.66**	56.00%	53.67%	54.28%	0.54
Ours (with adding Noise)	**22.67±3.73**	41.88±5.32	**41.04±3.46**	**57.20%**	54.76%	**55.25%**	**0.55**

Table 4: The test results with GPT-Neo on SQuAD v2.0 (NLU task) with 0.95 confidence level.

gorithm in the F1 metric. Additionally, our model outperformed baselines regarding text fluency and semantic similarity with standard answers. Furthermore, we found that our models exhibited significant improvement in the SQuAD dataset. We suppose this is because the SQuAD dataset has deterministic answers, making it easier for the model to learn a deterministic mapping function and achieve better performance during fine-tuning. These findings indicate that our method applies not only to dialogue systems but also brings improvements in general NLP tasks.

5.2. Analysis of manual evaluation

Due to the relatively poor performance of our GPT-2-based generative policy, we conducted a manual evaluation for the ability of answer ranking of the reward model used. Our manual evaluation demonstrated that the reward model achieved an 83.33% probability of ranking answers in the same order as the human beings on the DailyDialogue dataset, and a 63.00% of that was observed on the Movie dataset. Notably, the figure of that was only 4.17% for random ranking. Consequently, our reward model exhibited a high level of consistency with human judgments of answer quality and can serve as a substitute for manual annotation to a significant extent.

Furthermore, as an ablation study, we observed that human annotators achieved higher agreement coefficients when ranking answers generated with ISODATA. Specifically, on the DailyDialog dataset, the kappa coefficient of the three annotators for ranking filtered answers was 46.67. In contrast, the kappa coefficient for unfiltered answers was only 26.67. Similarly, on the Cor-

nellMovie dataset, the corresponding kappa coefficients for filtered and unfiltered answers were 18.51 and 11.11, respectively. Our analysis suggests that using the ISODATA algorithm can enhance the representativeness of the answers generated by our method.

5.3. Additional study

Can we observe clusters of answers in the semantic space? We conducted an additional experiment with GPT-2 to validate our hypothesis in Section 3.3, i.e., clusters of answers generated exist in the semantic space. Specifically, we randomly selected a question from the SQuAD dataset and applied principal component analysis (Pearson, 1901) and singular value decomposition (Golub and Kahan, 1965) to visualize semantic vectors of answers generated, as illustrated in Figure 2. Here, we used a BERT model (Devlin et al., 2019) not utilized in the above experiments to embed these answers to prevent possible information leakage.

Our analysis indicates that most of the answers generated exhibit apparent clustering. At the same time, the scattered data points are irrelevant or incorrect answers generated with relatively high probabilities during decoding. These low-quality answers were ranked lower as negative samples for training the reward model. Furthermore, we observed that the cluster centers obtained by the ISODATA algorithm can represent the distribution pattern of the answers.

6. Limitations

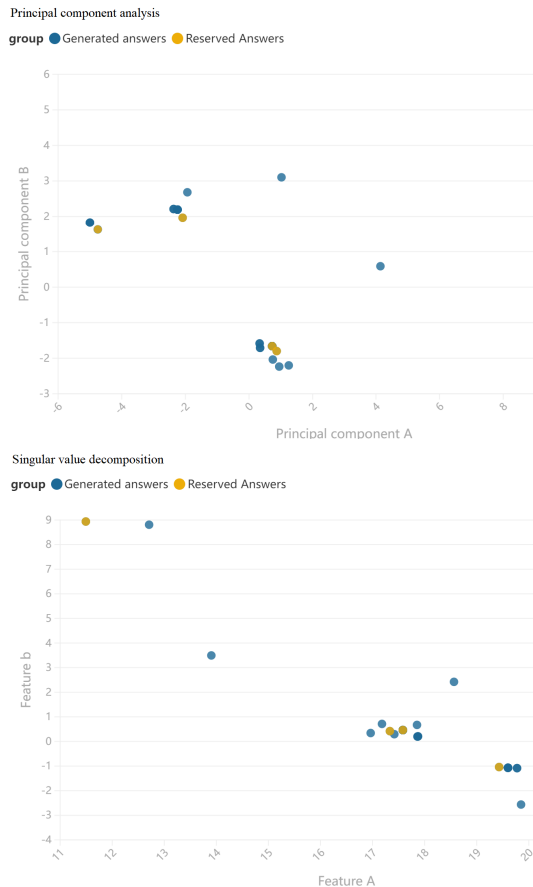While our model outperforms the baseline on various evaluation metrics, our fine-tuning approach

Figure 2: We applied principal component analysis and singular value decomposition to the BERT embeddings of answers generated by GPT-2.

first exhibits higher computational complexity. Regarding space complexity, the proposed method requires using three pre-trained models simultaneously when applying the PPO: a reward model, a PLM for the generative policy, and another PLM for the sampling policy. Since the space complexity counts three times that of the fine-tuned PLMs, our method is much more computationally demanding than the baselines. Regarding the time complexity, all fine-tuning approaches presented in this study exhibit a linear one, i.e., proportional to the sequence's length. However, in practical training, our training time is more than twice that required for full fine-tuning since the input sequences should be input to three models for loss computation. Compared to non-full-parameter fine-tuning methods like LoRA, although our approach has improved performance, it may also require more noticeable resource consumption.

Secondly, the extensive knowledge base of large language models may result in not generating answers with noticeable deviations, i.e., error points, in the semantic distribution when developing answers. This also makes the three noise introduction rules we proposed unable to create negative

samples that could significantly improve their performance. Due to limitations in our computational resources, we can only make theoretical assumptions the proposed approach yields performance improvements when we apply it to advanced large-scale PLMs. In summary, while the limitations exist, our approach addressed the reliance on human labor in the RLHF training pipeline and demonstrated its effectiveness on popular PLMs with relatively fewer parameters.

7. Conclusion

This paper introduces STR, a self-supervised pipeline that leverages proximal policy optimization for fine-tuning language models. We aim to reduce the need for manual labor, making RLHF-based algorithms more accessible and practical for researchers. Experimental results with two models across three tasks show that our fine-tuning method improves three points over the baselines regarding BLEU, ROUGE, and ME-TEOR. Additionally, manual evaluations indicate that our proposed text ranking algorithm generates annotations similar to human-generated ones. This discovery provides a cost-effective framework for future PPO-guided models to automatically generate training data. As a result, our research contributes to the advancement of self-supervised learning in fine-tuning pre-trained language models, opening up new possibilities for applying reinforcement learning in natural language processing.

8. References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

Hussam Alkaissi and Samy I McFarlane. 2023. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2).

Razvan Azamfirei, Sapna R Kudchadkar, and James Fackler. 2023. Large language models and the perils of their hallucinations. *Critical Care*, 27(1):1–2.

Geoffrey H Ball and David J Hall. 1967. A clustering technique for summarizing multivariate data. *Behavioral science*, 12(2):153–155.

Satanjeev Banerjee and Alon Lavie. 2005. ME-TEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*,

pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Anna Bodonhelyi, Efe Bozkir, Shuo Yang, Enkelejda Kasneci, and Gjergji Kasneci. 2024. User intent recognition and satisfaction with large language models: A user study with chatgpt. *arXiv preprint arXiv:2402.02136*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Gene Golub and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona,

Spain. Association for Computational Linguistics.

Jinying Lin, Zhen Ma, Randy Gomez, Keisuke Nakamura, Bo He, and Guangliang Li. 2020a. A review on interactive reinforcement learning from human social feedback. *IEEE Access*, 8:120757–120765.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020b. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. GLEU: Automatic evaluation of sentence-level fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351, Prague, Czech Republic. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: Bring order to the web. Technical report, Technical report, stanford University.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Karl Pearson. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Kunxun Qi, Hai Wan, Jianfeng Du, and Haolan Chen. 2022. Enhancing cross-lingual natural language inference by prompt-learning from cross-lingual templates. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1910–1923, Dublin, Ireland. Association for Computational Linguistics.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Kaili Sun, Xudong Luo, and Michael Y. Luo. 2022. A survey of pretrained language models. In *Knowledge Science, Engineering and Management*, pages 442–456, Cham. Springer International Publishing.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Laura Weidinger, John F. J. Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zachary Kenton, Sande Minnich Brown, William T. Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models. *ArXiv*, abs/2112.04359.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Hao Zhang and Jie Wang. 2021. An unsupervised semantic sentence ranking scheme for text documents. *Integrated Computer-Aided Engineering*, 28(1):17–33.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.