Reinforcement Learning with GPT-3 The Pre-trained Language Model to Optimize Generating Ads Headlines

Prof. Mohab Mangoud and Asma Shayea
Department of Electrical Engineering, University of Bahrain, Kingdom of Bahrain

Corresponding author: Asma Shayea (e-mail: asmashayea@gmail.com)

Abstract
Companies across almost every industry are looking to optimize their marketing strategies. With online advertising which is one of the marketing approaches, ad headline optimization is essential for attracting potential customers and increasing the achievement of the targeted goals (conversion rates). In this paper, we present a reinforcement learning (RL) approach to generate effective ad headlines using Generative Pre-trained Transformer form OpenAI's GPT-3 model. The GPT-3 model will generate ad headlines text based on the keywords provided by a reinforcement learning agent who is trained to find the best combination of keywords. The performance of the proposed model is evaluated based on the conversion rates achieved during the training process resulting in the potential to generate more attractive ad headlines that lead to improved conversion rates.

Keywords
Reinforcement Learning, Pre-Trained Model, GPT-3, Ad Headline, RL Agent, Conversion Rate.

1.INTRODUCTION

1.1 Literature Survey

[1]The paper "Fine-Tuning Language Models from Human Preferences"[1] proposes a reinforcement learning approach to fine-tune pre-trained language models specifically GPT-2 model based on human preferences and its effectiveness on different tasks such as summarization, and question answering. The Results show how effectively RL can fine-tune the GPT-2 model based on human preferences, and improve the performance on several text generation tasks.

[2]The paper "Survey on Reinforcement Learning for Language Processing" discusses the different types of RL algorithms used in language processing, such as policy gradient methods and Q-learning. It provides a survey of the use of RL to improve language processing tasks using different pre-trained language models, such as BERT and GPT-2, and how these models can be fine-tuned using RL to improve their performance on many tasks like text generation, dialogue systems.

[3] "NARLE: Natural Language Models using Reinforcement Learning with Emotion Feedback". The paper proposes how can use reinforcement learning with emotional feedback to improve natural language models using a proposed approach called NARLE (Natural Language Models using Reinforcement Learning with Emotion Feedback) and the GPT-2 pre-trained model. This model is used in various tasks including text generation, and sentiment analysis.

1.2 Objective

Online advertising is important for businesses as it has become one of the most business revenue sources in recent years. The main problem with online advertising is that the advertisement reaches the untargeted user who is not interested in the

products or services they offered which leads to cost companies time and money spent on these advertisements with no revenue or any conversion achieved. The ad campaign in online advertising like google ad is a group of different ads. Each ad includes text headline and description which play a crucial role to attract the right customer. Deep learning pre-trained models like GPT-3, have shown promising results in automating various tasks such as text generation. The objective is using reinforcement learning (RL) to improve the text generated by GPT-3 which will iteratively generate ad headlines based on the most effective keywords provided by a reinforcement learning agent for optimizing the ad text based on the conversion they will get by user who viewed the ad (i.e., click on the ad, buy a product).

1.3 Organization

The paper is organized as follows: Section 2 describes the proposed model, including its architecture and dataset. Section 3 presents the implementation of the proposed model. Section 4 reports the results and section 5 includes the comparative study. Section 6 concludes the paper, and Section 7 lists the references.

2. PROPOSED MODEL

2.1 Architecture

GPT-3 model: is a language *model* that was created by OpenAI which can fine-tune it to handle common tasks such as natural language processing.

Q-learning algorithm: The Reinforcement learning algorithm that learns the optimal policy by iteratively updating a Q-value function which will estimate the received rewards for taking a particular action in a particular state.

AdEnvironment class: the environment in which the agent interacts. It is initialized with information about the target audience and location of the launched ad campaign. The environment then generates ad headlines using the GPT-3 model and evaluates their conversion rates.

AdAgent class: The reinforcement learning agent is responsible for implementing the RL algorithm using Q-learning algorithm to select actions by defining the optimal keywords and update these keywords (actions) based on their rewards.

Reward: The reward in this approach is the conversion rate which is calculated by dividing the number of conversions which are the targeted goals determined by the advisor (i.e., click on ads, complete payment process, product purchased) by the number of impressions which are how many this ad has been displayed to the user whether they interact with it or not.

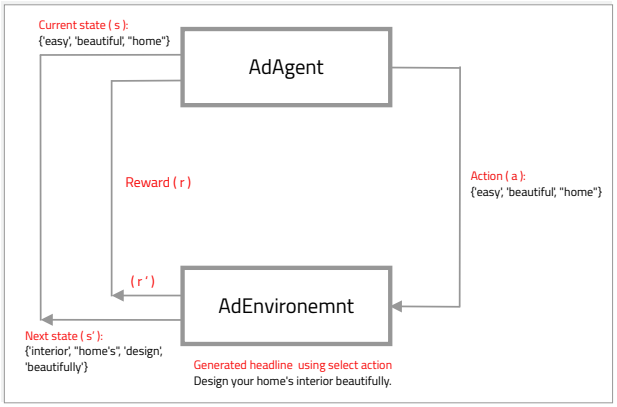

Fig.1: RL Architecture

2.2 Dataset

The dataset is created for training and evaluation purposes consisting of a list of various keywords that will be extracted from the generated ad headline. When GPT-3 generates ad text it will then extract the most common keywords from that text as shown in Figure 1.

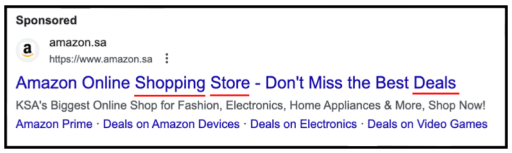

Fig.2: Example of the most common keywords are underlined in the ad headline which will be extracted from ad headline.

3. IMPLEMENTATION

We have implemented our proposed model using Python and OpenAI's GPT-3 API.

A- The agent (AdAgent) is responsible for:

1. Suggesting top keywords from the dataset with the maximum Q-value in the form of action.
2. Updating keywords Q-table based on the rewards received from the environment (i.e., conversion rates).

B- The GPT-3 model is used for both ad headline generation then keyword extraction from the generated headlines.

1. GPT-3 prompt to generate ad headline using top keywords(action) selected by agent:

```
ad_headline_prompt = f"Generate a Google ad headline with a maximum of 40 characters for an interior design platform catering to homeowners. Ensure the headline follows SEO standards and includes the keywords: {keywords_list}.."
```

2. GPT-3 prompt to extract the new keywords from the generated ad headline to be evaluated based on rewards (conversion rate) then add to the keywords dataset with the updated q-value by agent:

```
keywords_prompt = f"Extract the important keywords from the following ad headline text and put them in a Python array:\n\nAd headline text: \"{headline}\"\n\nExample output:"
```

C- The environment (AdEnvironment) is responsible for:

1. Generating ad headlines based on the agent's suggested top keywords.
2. Extracting keywords from the generated ad headlines.
3. Providing a state (randomly chosen between 0 and 1) to the agent.
3. Training the agent for a specified number of episodes and updating its Q-table.

The Q-table will be updated based on rewards (conversion rates) received. As implementing this system will need observing it among a long period of time in a real-world environment, we assign a human (user input) to expect the number of conversions which is used to calculate the reward for each generated ad headline based on its quality and effectiveness. The purpose of updating the Q-table is to help the agent learn the optimal policy (i.e., the best keywords or actions) for generating better ad headlines.

The equation used in the updating method is the Q-learning update rule, which is as follows:

$$Q(s, a) = Q(s, a) + \alpha * (R + \gamma * \max(Q(s', a')) - Q(s, a))$$

In our method in the code for updating q-value, It implements this equation as follows:

1. Retrieve the current Q-value q_current for the given state and action:

```
q_current = self.Q.get((state, action), 0.0).
```

2. Compute the maximum Q-value q_next for the next state and all possible actions:

```
q_next = max([self.Q.get((next_state, a), 0.0) for a in self.actions]).
```

3. Update the Q-value for the current state and action based on the Q-learning update rule:

```
self.Q[(state, action)] = q_current + self.alpha * (reward + self.gamma * q_next - q_current).
```

This will help the agent to improve its Q-table and learn the optimal policy by finding the maximum Q-value in the next state and calculating the new rewards. The agent then will be trained for a specified number of episodes, during which iteratively generates ad headlines and updates the Q function based on the conversion rates achieved. At the end of the training process, the agent's performance is evaluated based on the average conversion rate achieved through the training episodes.

Episode	State	Selected action (top keywords)
1	0	['Interior design', 'Homeowner', 'renovation']
2	0	['platform']
3	1	['easy', 'beautiful', 'home's']
4	1	['home's', 'beautifully', 'design']
5	0	['platform']
6	0	['platform']
7	1	['beautifully']
8	1	['design']
9	1	['design']
10	0	['platform']

Episode	Generated Headline	Reward (Conversion Rate)
1	Make your home beautiful with our easy to use interior design platform for homeowners!	0.2
2	Design your home's interior with our platform - it's easy and fun!	0.3
3	Design your home's interior easily and beautifully	0.33
4	Design your home's interior beautifully.	0.34
5	The Best Interior Design Platform for Homeowners	0.35
6	Interior Design Platform for Homeowners - Design Your Home with Ease!	0.35
7	Beautifully designed interiors for your home.	0.45
8	Design your home with our interior design platform.	0.46
9	Design your home with our easy-to-use platform.	0.48
10	Looking for an interior design platform? Look no further!	0.48

Episode	Current State (Keywords)
1	['interior', 'design', 'use', 'easy', 'beautiful', 'homeowners', 'home', 'platform']
2	['interior', 'home's', 'design', 'easy', 'fun', 'platform']
3	['beautifully', 'interior', 'home's', 'design', 'easily']
4	['interior', 'home's', 'design', 'beautifully']
5	['design', 'platform', 'homeowners', 'interior']
6	['interior', 'home', 'homeowners', 'design', 'platform', 'ease']
7	['home', 'interiors', 'design']
8	['home', 'interior', 'design', 'platform']
9	['easy', 'design', 'use', 'to', 'home', 'platform']
10	['design', 'platform', 'interior']

Fig.3: This figure displays the code result for our proposed model. 'Selected action' shows the keywords (action) selected by the agent based on the higher q-value. Using these keywords, the environment using GPT-3 will generate the ad headline and then extract the keywords for the next state to be evaluated by a human assessment, therefore, will be rewarded based on the conversion rate expected by the human as shown in the table.

4. RESULTS

4.1 Evaluation metrics

The average reward per episode (avgReward): is the total rewards divided by the total number of episodes. A higher value means that the agent is generating better ad headlines that lead to higher conversion rates.

The average maximum Q-value per episode (avgQValue): It is calculated by dividing the total maximum Q-values for all episodes by the total number of episodes. A higher average means that the agent has learned better estimation for the state-action pairs.

The average number of actions(avgActions): which is the unique keywords per episode calculated by dividing the total of the number of unique keywords in all episodes by the total number of episodes to shows the diversity of keywords has learned by agent.

avgReward	avgQValue	avgActions
0.345	0.10193603713148405	12.5

Table.1: The model results using different evaluation metrics.

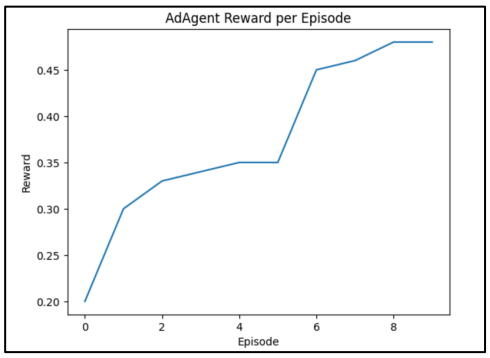

Fig4. the curve shows that the conversion increased by episode.

These results suggest that the reinforcement learning agent was able to learn an effective policy for generating ad headlines, resulting in higher conversion rates and more effective ads. However, additional evaluation and testing may be required to confirm the effectiveness of the proposed approach in real-world scenarios.

5. COMPARATIVE STUDY

We have conducted a comparative study to evaluate the performance of our proposed model against a traditional approach. The traditional approach involves generating ad headlines using GPT-3 without any reinforcement learning-based optimization therefore without suggesting specific keywords as shown in the following prompt text:

```
ad_headline_prompt = f"Generate a Google ad headline with a maximum of 40 characters for an interior design platform catering to homeowners. Ensure the headline follows SEO standards"
```

The results of the comparative study shows that our proposed model compared to the traditional approach achieved more effectively generated ad headlines with diversity in context and more relative to what the advertisement should marketing to. This indicates the potential of our reinforcement learning-based approach in generating ad headlines that lead to improve conversion rates, therefore, increasing the revenue for the business.

Generated Headline
Looking to renovate your home? Check out our interior design platform for homeowners!
Interior design platform for homeowners
Interior Design for Homeowners - Get Inspired for Your Next Renovation
Interior design platform for homeowners
Homeowner? Get interior design inspiration & tips for your renovation.
Interior Design for Homeowners - Get Inspired for Your Next Renovation
Homeowners: Get Inspired for Your Next Interior Design Project
Looking to renovate? Get interior design ideas & advice for homeowners.
Interior design platform for homeowners
Interior Design Platform for Homeowners

Fig.5: Traditional approach

Generated Headline
Make your home beautiful with our easy to use interior design platform for homeowners!
Design your home's interior with our platform - it's easy and fun!
Design your home's interior easily and beautifully
Design your home's interior beautifully.
The Best Interior Design Platform for Homeowners
Interior Design Platform for Homeowners - Design Your Home with Ease!
Beautifully designed interiors for your home.
Design your home with our interior design platform.
Design your home with our easy-to-use platform.
Looking for an interior design platform? Look no further!

Fig.5: Our model with the RL approach

6. Conclusion

In conclusion, we have presented a deep reinforcement learning-based approach to optimize ad headlines using OpenAI's GPT-3 model. Our proposed model iteratively generates ad headlines based on the keywords provided by a reinforcement learning agent, which is trained to find the optimal combination of keywords that lead to improve conversion rates.

The results of our comparative study show that the proposed model achieves higher average conversion rates compared to traditional approach, demonstrating the potential of reinforcement learning-based optimization in generating an effective ad headlines.

Future work could explore more advanced algorithms in reinforcement learning that may offer better performance like Deep Q-Networks (DQN) or Actor-Critic methods. we also can consider adding more features like time of day, or day of the week. We are looking to using GPT-3 to generate headlines not just based on keywords, it could also use it to analyze user behavior data by summarizing user feedback, reviews, or social media comments about our products and services then generate headlines based on these insights. Furthermore, the performance of the proposed model could be evaluated on a larger scale and in real-world ad campaigns.

7. References

[1] Ziegler, Daniel & Stiennon, Nisan & Wu, Jeffrey & Brown, Tom & Radford, Alec & Amodei, Dario & Christiano, Paul & Irving, Geoffrey. (2019). Fine-Tuning Language Models from Human Preferences.

[2] Uc-Cetina, Victor & Navarro-Guerrero, Nicolás & Martin-Gonzalez, Anabel & Weber, Cornelius & Wermter, Stefan. (2022). Survey on Reinforcement Learning for Language Processing.

[3] Zhou, Ruijie & Deshmukh, Soham & Greer, Jeremiah & Lee, Charles. (2021). NaRLE: Natural Language Models using Reinforcement Learning with Emotion Feedback.

[4] Deng, Mingkai & Wang, Jianyu & Hsieh, Cheng-Ping & Wang, Yihan & Guo, Han & Shu, Tianmin & Song, Meng & Xing, Eric & Hu, Zhiting. (2022). RLPrompt: Optimizing Discrete Text Prompts With Reinforcement Learning. 10.48550/arXiv.2205.12548.

[5] Arulkumaran, Kai & Deisenroth, Marc & Brundage, Miles & Bharath, Anil. (2017). A Brief Survey of Deep Reinforcement Learning. IEEE Signal Processing Magazine. 34. 10.1109/MSP.2017.2743240.

[6] Otter, Daniel & Medina, Julian & Kalita, Jugal. (2020). A Survey of the Usages of Deep Learning for Natural Language Processing. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2020.2979670.

[7] Shi, Z., Chen, X., Qiu, X., & Huang, X. (2018). Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*.

[8] Tutorials helped with the code: https://medium.com/@mlblogging.k/reinforcement-learning-for-tuning-language-models-how-chatgpt-is-trained-9ecf23518302, https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/.

Appendix

```python
# Set up OpenAI API credentials
openai.api_key = "sk-n7oclVsXXWDd2nu4XIjuT3BlbkFJ0SPW9IexI3
# Set up GPT-3 model
model_engine = "text-davinci-002"

# Define function to generate ad headline
def generate_ad_text(prompt):
    response = openai.Completion.create(
        engine=model_engine,
        prompt=prompt,
        max_tokens=100,
        n=1,
        stop=None,
        temperature=0.5,
    )

    text = response.choices[0].text
    return text

headline_keywords = []
```

Figure.1: setup GPT-3 API

```python
class AdEnvironment:
    def __init__(self):
        self.num_states = 2

    def generate_headline(self, top_keywords):
        keywords_list = ", ".join(keyword.lower() for keyword in top_keyword
        print("keywords_list: ", keywords_list)

        ad_headline_prompt = f"Generate a Google ad headline with a maximum
        +"platform catering to homeowners. Ensure the headline follows SEO s
        headline = generate_ad_text(ad_headline_prompt).strip()
        print("Generated headline: ", headline)

        keywords_prompt = f"Extract the important keywords from the followin
        +"Python array:\n\nAd headline text: \"{headline}\"\n\nExample outpu
        extracted_keywords = generate_ad_text(keywords_prompt).lower()
        extracted_keywords_list = re.findall(r"[\w']+", extracted_keywords)

        unique_matches_list = list(set(extracted_keywords_list))
        new_keywords = [element.strip("'").strip('"') for element in unique_

        return new_keywords, headline

    def get_state(self):
        return random.randint(0, self.num_states - 1)
```

Fig.2: AdEnvironment

```python
class AdAgent:
    def __init__(self, alpha=0.1, gamma=0.9, epsilon=0.1):
        self.alpha = alpha
        self.gamma = gamma
        self.epsilon = epsilon
        self.actions = []
        self.Q = {}

    def add_new_keywords(self, new_keywords):
        for new_keyword in new_keywords:
            if new_keyword not in self.actions:
                self.actions.append(new_keyword)
    def get_action(self, state):
        actions = []

        if not self.Q:
            return ["Interior design", "Homeowner", "renovation"]

        if np.random.uniform() < self.epsilon:
            action = random.sample(self.actions, min(len(self.actions), 3))
        else:
            q_values = [self.Q.get((state, a), 0.0) for a in self.actions]
            max_q = max(q_values)

            if q_values.count(max_q) > 1:
                best_actions = [i for i in range(len(self.actions)) if q_values[i] == max_q]
                action_idx = random.sample(best_actions, min(len(best_actions), 3))
            else:
                action_idx = [q_values.index(max_q)]

            actions = [self.actions[idx] for idx in action_idx]

        return actions
```

Fig.3: AdAgent Class

```python
def train_agent(self, num_episodes=10):
    agent = AdAgent()
    total_reward = 0
    total_q_value = 0
    total_actions = 0
    rewards = []
    table = []

    for i in range(num_episodes):
        environment = AdEnvironment()
        current_state = environment.get_state()
        top_keywords = agent.get_action(current_state)
        current_state_keywords, headline = environment.generate_headline(top_keywords)
        agent.add_new_keywords(current_state_keywords)

        impression = 100
        conversion_no = float(input("Enter conversion_no: "))
        conversion_rate = conversion_no / impression
        reward = conversion_rate

        total_reward += reward
        rewards.append(reward)
        next_state = environment.get_state()
        agent.reinforce(current_state, tuple(top_keywords), reward, next_state)

        for action in current_state_keywords:
            agent.update_q(current_state, action, reward, next_state)

        total_q_value += max([agent.Q.get((current_state, a), 0.0) for a in agent.actions])
        total_actions += len(agent.actions)

        table.append([i+1, current_state, top_keywords, current_state_keywords, headline, conversion_rate])
```

Fig.4: Training agent for 10 episode