AutoGPTs

Pranav Kuppa

April 2023

1 Introduction

Autoregressive Generative Pre-training Transformers (AutoGPTs) are a class of deep learning models that have gained popularity recently due to their impressive performance on various natural language processing (NLP) tasks. After the release of GPT-4, these models are being developed in the hundreds. The AutoGPT models we will be discussing use GPT-4 to make the user's process fully autonomous seamlessly moving from one task to the next until the user's request is achieved.

In this paper, we will compare and contrast different AutoGPT models and their corresponding algorithms. We will start by discussing the Baby AGI and Jarvis models before focusing on the Auto-GPT model developed by Significant Gravitas as well as the mathematical theory behind it.

2 Baby AGI and Jarvis

Baby AGI is an AutoGPT model developed by Yohei Nakajima that was designed to learn from a small amount of data and generalize to new tasks. The model achieves this by using a combination of unsupervised and supervised learning. The unsupervised learning phase involves pre-training the model on a large corpus of text data, while the supervised learning phase involves fine-tuning the model on a smaller dataset specific to the task at hand.

Jarvis, on the other hand, is a more advanced AutoGPT model developed by Microsoft that is designed to be more robust and flexible than previous models. Jarvis is trained on a large and diverse dataset of text, images, and audio, and is capable of understanding and generating text, speech, and images. The model achieves this by using a combination of supervised and unsupervised learning, as well as reinforcement learning.

3 Auto-GPT

Auto-GPT is a state-of-the-art algorithm for automatically generating neural network architectures, specifically based on the transformer architecture, using a reinforcement learning approach. Auto-GPT is built on top of the GPT (Generative Pre-trained Transformer) architecture, which was introduced in 2018 by Radford et al. The GPT architecture is a type of neural network that is designed to generate sequences of text, which has been used in various natural language processing tasks such as language translation, text classification, and text generation.

The idea behind Auto-GPT is to automate the process of designing neural networks by treating it as a search problem, where the goal is to find the optimal architecture that maximizes some performance metric. In this paper, we will describe the theory behind Auto-GPT and how it works, as well as discuss the implementation of the Auto-GPT algorithm by Significant Gravitas, which is available on GitHub.

Auto-GPT is based on the idea of neural architecture search (NAS), which is the process of automatically searching for the best neural network architecture for a given task. The goal of NAS is to automate the process of designing neural networks, which can be a time-consuming and expensive process, especially for complex tasks.

The Auto-GPT algorithm uses a reinforcement learning approach to search for the best architecture. In reinforcement learning, an agent interacts with an environment and learns to take actions that maximize a reward signal. In the case of Auto-GPT, the agent is the neural network architecture, and the environment is the task that the architecture is being designed for. The reward signal is a performance metric, such as accuracy or F1 score.

Let T be the task that the architecture is being designed for, and let P be the space of possible neural network architectures. The goal of Auto-GPT is to find the optimal architecture $p^* \in P$ that maximizes the performance metric $M(T, p)$, where $M(T, p)$ is the performance of architecture p on task T. This can be written as the following optimization problem:

$$p* = *argmax_{p \in P} M(T, p)$$

To solve this optimization problem, Auto-GPT uses a reinforcement learning algorithm called Proximal Policy Optimization (PPO), which is a type of policy gradient algorithm. The algorithm then evaluates each candidate architecture by training it on a subset of the training data and measuring its performance on a validation set. The performance metric is used as the reward signal for the reinforcement learning algorithm. The algorithm then updates the parameters of the neural network architecture based on the reward signal, with the goal of maximizing the reward. The updated parameters are then used to generate a new population of candidate architectures, and the process is repeated until the optimal architecture is found.

The policy network in PPO is a neural network that maps the current state (i.e., the current architecture) to a probability distribution over actions (i.e., possible modifications to the architecture). Let $\pi_\theta(a|s)$ be the probability of taking action a in state s, where θ are the parameters of the policy network.

The goal of the policy network is to learn a mapping from states to actions that maximizes the expected reward, which is given by the following equation:

$$J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

where r_t is the reward at time step t, T is the time horizon, and γ is a discount factor that determines the weight of future rewards. The policy network is updated using the policy gradient, which is given by the following equation:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) A(s,a)\right]$$

where $A(s,a)$ is the advantage function, which measures how much better action a is than the average action in state s. The advantage function is given by:

$$A(s,a) = Q(s,a) - V(s)$$

where $Q(s,a)$ is the state-action value function, which measures the expected reward starting from state s and taking action a, and $V(s)$ is the state value function, which measures the expected reward starting from state s and following the policy. The state value function can be estimated using a separate neural network called the value network. Both functions can be estimated using the Bellman equations:

$$Q(s,a) = E[R + \gamma \max_{a'} Q(s',a')]$$
$$V(s) = E[R + \gamma V(s')]$$

The Auto-GPT algorithm uses PPO to update the parameters of the policy network and the value network based on the reward signal. The policy network is used to generate new candidate model architectures, and the value network is used to estimate the value of each candidate architecture.

4 Applications

The Auto-GPT algorithm can be used for various natural language processing tasks, such as language translation, text classification, and text generation. The Auto-GPT algorithm can also be applied to other domains, such as computer vision and speech recognition.

The Auto-GPT program by Significant Gravitas, which is available on GitHub, can be used to automatically generate transformer architectures for natural language processing tasks. The implementation provides an interface for specifying the task and the search space, and it allows users to customize the search process by specifying various hyperparameters. Once all are chosen, the Auto-GPT algorithm can be run to automatically generate a transformer architecture for the task. The output of the algorithm is a neural network architecture, which can be trained and evaluated on the task.

While the program is nowhere near perfect, it is important to consider that this project is less than a month old. It is still undergoing development, and will undoubtedly improve significantly within the next month.

5 Conclusion

In this paper, we compared and contrasted different AutoGPT models and their corresponding algorithms. We discussed the Baby AGI and Jarvis models before focusing on the Auto-GPT model developed by Significant Gravitas. We explained how Auto-GPT uses the Proximal Policy Optimization (PPO) algorithm to optimize the policy network by maximizing the expected reward over a given time horizon. We also discussed the policy gradient, advantage function, state-action value function, and state value function, and how they are used in the PPO algorithm.

AutoGPTs have made significant contributions to the field of natural language processing and have shown great promise in a variety of applications. As research in this area continues to evolve, we can expect to see the development of even more sophisticated technologies involving the extension of GPT-4 and its automation.

6 Try Them Yourself!

- https://github.com/yoheinakajima/babyagi
- https://github.com/microsoft/JARVIS
- https://github.com/Significant-Gravitas/Auto-GPT