Week 3 : Generative Pre-trained Transformer

120220210 고재현

2023년 3월 28일

1 Introduction

The GPT class of language models has attracted global attention as they can produce text resembling that written by humans. This article focuses on the OpenAI GPT model, explaining it intuitively. It utilizes semi-supervised learning, pretraining on large unlabeled corpora and fine-tuning on small labeled ones. The decoder segment of the original Transformer is used as the base architecture and hyperparameters are covered. Tasks the GPT model was fine-tuned on are also discussed. Takeaways include the effect of 'locking' layers, zero-shot behavior, and the unsupervised language model's ability to recognize linguistic patterns. The performance of Transformer-based[1] architectures for semi-supervised learning is compared to that of LSTMs.

2 model architecture

3 GPT Training

To understand how GPT works, it's important to understand the approach it utilizes, which is semi-supervised learning. This approach includes an unsupervised component, pretraining, which uses an unlabeled corpus of tokenized text to find a good initialization point for learning a specific task, and a supervised component, fine-tuning, which uses a labeled corpus of tokenized text tailored to a specific language task. This approach improves the performance of language models significantly, as it utilizes the abundance of unlabeled text to extract linguistic patterns, which serves as a better starting point for specialization using smaller labeled datasets. GPT falls under the semi-supervised learning approach and utilizes pretraining and fine-tuning.

The GPT model utilizes both the masked multi-head attention segment and the feed forward segment, along with the residuals and their corresponding addition and layer normalization steps. This involves position embedding of the learned embedding, followed by unidirectional self-attention through a masked multi-head attention segment. Residual is added and the result is layer normalized. The result then passes through a position-wise feedforward network and the residual is again added and layer normalized. The outcome either passes to the next decoder segment or is the output of the model as a whole.

Model	Context Window Size	Batch Size	Word Embedding Vector
GPT-1	512	64	768
GPT-2	Variable up to 2048	1 or 8	768
GPT-3	Variable up to 2048	Up to 8192	768, 1024, or 2048

〈표 1〉 Summary of the hyperparameter values in GPT-1, GPT-2, and GPT-3.

3.1 important hyperparameters

This table summarizes the differences in context window size, batch size, and word embedding vector across the three models. The values for each hyperparameter are listed for each model in their respective columns.

3.2 pretraining

During pretraining, there are no labels to guide the training process, making it unsupervised. However, we can utilize our large corpus of tokens $T_1, ..., T_n$ by applying a (sliding) context window of length k. This allows us to structure the text into windows, such as T_1, T_2, T_3, T_2, T_3, T_4, and so on, with $k = 3$. By feeding a context window to the GPT model, we can predict the next token.

3.3 fine-tuning

After pretraining, the GPT model can be fine-tuned using a labeled dataset C, as explained in paper [2]. Each instance in the dataset contains a sequence of tokens $x_1, x_2, ..., x_m$ and a corresponding label y. The sequence is passed through the pretrained Transformer architecture and then through a linear layer with weights W_y and Softmax activation for multiclass prediction.

3.3.1 auxilary objective in fine-tuning

Mathematically, the auxiliary objective can be expressed as:

$$\mathcal{L}_{aux} = \lambda ||C||_1 \tag{1}$$

where lambda is the regularization parameter and $||C||_1$ is the L1 norm of the weight vector C.

The auxiliary objective, λ∗L1(C), is used to improve the model's generalization ability. This objective function performs L1 regularization on the weight vector C and uses it as an additional loss function multiplied by lambda. In this way, the model can explore a wider range of parameter values and improve generalization performance while preventing overfitting. Therefore, it is reasonable to use pretrained neural network parameter values as the initial values for fine-tuning neural network parameters. This allows the model to leverage the information from the previously learned language model to start optimizing for the new task-specific dataset.

참고 문헌

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.