딥러닝프레임워크 과제#3.
Transformer 구조 조사 및 분석

서론

Transformer 모델의 소개

Transformer 모델은 2017년 Vaswani et al.에 의해 제안된 자연어 처리(NLP) 모델로, "Attention is All You Need"라는 논문에서 처음 소개되었습니다. Transformer 모델은 기존의 순환 신경망(RNN)과 장단기 메모리(LSTM) 모델의 한계를 극복하기 위해 설계되었습니다. RNN과 LSTM 모델은 시퀀스 데이터를 처리하는 데 사용되었지만, 긴 시퀀스를 처리할 때 정보 손실이 발생하고, 시퀀스의 길이가 길어질수록 학습 시간이 기하급수적으로 증가하여 병렬 처리가 어려운 문제를 겪었습니다.

Transformer 모델의 주요 혁신은 Self-Attention 메커니즘을 도입한 것입니다. Self-Attention 메커니즘은 입력 시퀀스의 각 단어가 시퀀스 내의 다른 모든 단어들과의 관계를 동시에 고려할 수 있게 합니다. 이는 RNN과 LSTM이 각 타임스텝마다 순차적으로 정보를 처리하는 방식과는 대조적입니다. Transformer 모델은 이러한 Self-Attention 메커니즘을 기반으로 하여, 입력 데이터 간의 관계를 효율적으로 파악하고, 병렬 처리를 통해 학습 속도를 크게 향상시킵니다.

Transformer 모델은 여러 NLP 작업에서 뛰어난 성능을 보여주었습니다. 예를 들어, 기계 번역에서는 소스 언어의 문장을 타겟 언어로 번역하는 데 있어 높은 정확도와 자연스러운 번역 결과를 제공합니다. 텍스트 요약에서는 긴 문서에서 중요한 내용을 추출하여 짧은 요약문을 생성하는 작업에서 우수한 성능을 발휘합니다. 또한, 질문 응답 시스템에서는 주어진 문맥에서 질문에 대한 정확한 답변을 생성할 수 있으며, 텍스트 생성 작업에서도 주어진 입력으로부터 자연스러운 문장을 생성하는 데 탁월한 성능을 보입니다.

Transformer 모델의 중요성 및 역할

Transformer 모델의 등장은 NLP 분야에서 큰 변화를 일으켰습니다. 기존의 RNN 기반 모델들은 여러 한계로 인해 긴 시퀀스 데이터를 효과적으로 처리하지 못했고, 병렬 처리에 비효율적이었습니다. 반면, Transformer는 Self-Attention 메커니즘을 통해 입력 시퀀스의 모든 부분 간의 상호작용을 한 번에 고려할 수 있습니다. 이는 긴 시퀀스도 정보 손실 없이 효과적으로 처리할 수 있게 합니다.

Transformer의 Self-Attention 메커니즘은 단어 간의 문맥적 상호작용을 강화하여, 문장의 의미를 더 정확하게 이해할 수 있게 합니다. 이는 단순히 단어의 순서를 따르는 것이 아니라, 각 단어가 다른 단어들과 어떻게 상호작용하는지를 파악하는 데 중점을 둡니다. 이로 인해, Transformer는 긴 문맥을 필요로 하는 작업에서도 높은 성능을 발휘할 수 있습니다.

또한, Transformer 모델은 병렬 처리가 가능하여 학습 속도와 효율성을 크게 향상시켰습니다. RNN과 LSTM 모델은 시퀀스 데이터를 순차적으로 처리해야 하기 때문에 병렬 처리가 어

렵고, 학습 속도가 느렸습니다. 하지만 Transformer는 입력 시퀀스를 동시에 처리할 수 있어, 병렬 컴퓨팅을 활용하여 대규모 데이터셋을 빠르게 학습할 수 있습니다. 이는 대규모 언어 모델의 훈련 시간을 단축시키고, 더 복잡한 모델을 효과적으로 학습할 수 있게 합니다.

Transformer 모델의 이러한 특성 덕분에, 다양한 NLP 작업에서 주도적인 역할을 하고 있습니다. 특히, GPT-3와 BERT와 같은 강력한 언어 모델은 Transformer를 기반으로 구축되었습니다. GPT-3는 주어진 텍스트 프롬프트로부터 자연스럽고 일관된 텍스트를 생성할 수 있는 모델로, 챗봇, 자동 글쓰기 도구 등에서 활용되고 있습니다. BERT는 문맥적 의미를 이해하기 위해 양방향 Transformer를 사용하여, 문장 수준에서 단어의 의미를 더 정확하게 파악할 수 있습니다. BERT는 텍스트 분류, 문장 유사도 측정, 질문 응답 시스템 등 다양한 NLP 작업에서 높은 성능을 보입니다.

Transformer 모델은 또한 기계 번역, 텍스트 생성, 문서 요약 등 다양한 NLP 작업에서도 주도적인 역할을 하고 있습니다. 기계 번역에서는 입력 문장을 다른 언어로 번역하는 작업에서 높은 정확도와 자연스러운 번역 결과를 제공합니다. 텍스트 생성 작업에서는 주어진 입력으로부터 자연스럽고 일관된 문장을 생성할 수 있습니다. 문서 요약에서는 긴 문서에서 중요한 내용을 추출하여 짧은 요약문을 생성하는 작업에서 우수한 성능을 발휘합니다.

이처럼 Transformer 모델은 NLP 분야에서 큰 변화를 일으키며, 다양한 응용 분야에서 주도적인 역할을 하고 있습니다. Transformer 모델의 등장은 NLP 연구와 실제 응용 프로그램에 새로운 가능성을 열어주었고, 인공지능 기술의 발전에도 큰 기여를 하고 있습니다.

이론적 배경

Self-Attention

Self-Attention 메커니즘은 입력 시퀀스의 각 단어가 시퀀스 내의 다른 모든 단어들과 어떻게 상호작용하는지 파악하는 방법입니다. 이는 Transformer 모델의 핵심 요소로, 입력 시퀀스의 문맥 정보를 효과적으로 반영할 수 있게 합니다. Self-Attention의 작동 원리는 다음과 같습니다:

1. 입력 임베딩:
입력 시퀀스의 각 단어는 고차원 벡터로 임베딩됩니다. 예를 들어, "The cat sat on the mat"이라는 문장이 입력되면, 각 단어는 고차원 임베딩 공간에서 벡터로 변환됩니다. 이 임베딩 벡터는 일반적으로 사전 학습된 임베딩(예: Word2Vec, GloVe) 또는 학습 가능한 임베딩 행렬을 통해 생성됩니다.

2. 쿼리, 키, 밸류 벡터:
각 임베딩된 단어는 쿼리(Q), 키(K), 밸류(V) 벡터로 변환됩니다. 이 벡터들은 학습 가능한 가중치 행렬을 통해 생성됩니다. 구체적으로, 쿼리, 키, 밸류 벡터는 다음과 같은 방식으로 계산

됩니다:
여기서 X는 입력 임베딩 행렬이고, WQ, Wk, Wv는 각각 쿼리, 키, 밸류 가중치 행렬입니다. 이 변환 과정을 통해 입력 시퀀스의 각 단어는 Q, K, V 벡터로 매핑됩니다.

$$Q = XW_Q$$
$$K = XW_K$$
$$V = XW_V$$

3. 주의 점수 계산:
쿼리 벡터 Q와 키 벡터K의 내적(dot product)을 계산하여 주의 점수를 구합니다. 주의 점수는 다음과 같이 계산됩니다:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

여기서 dk는 키 벡터의 차원입니다. 이 내적 계산은 각 단어가 다른 모든 단어들과 얼마나 관련이 있는지를 나타내는 점수를 생성합니다. 점수는 소프트맥스 함수를 통해 확률 분포로 변환되어, 각 단어의 중요도를 결정합니다.

4. 가중합:
각 단어의 밸류 벡터 V에 주의 점수를 가중합하여 최종 출력을 생성합니다. 이를 통해 입력 시퀀스의 각 단어는 다른 단어들과의 상호작용을 고려한 새로운 표현으로 변환됩니다. 이 과정은 다음과 같이 요약할 수 있습니다:

$$\text{Output} = \sum(\text{Attention Score} \times V)$$

Self-Attention 메커니즘은 입력 시퀀스의 모든 단어들이 서로의 관계를 고려할 수 있게 하여, 문맥 정보를 잘 반영할 수 있습니다. 이는 특히 문장의 구조와 의미를 정확하게 이해하는 데 중요한 역할을 합니다.

Positional Encoding

Transformer 모델은 입력 시퀀스의 순서를 고려하기 위해 Positional Encoding을 사용합니다. 순차적인 정보가 없는 Self-Attention 메커니즘은 단어 간의 상대적 위치를 인식하지 못하므로, 위치 정보를 명시적으로 추가해야 합니다. Positional Encoding은 각 단어의 위치 정보를 임베딩 벡터에 추가하는 방식으로, 주로 사인 함수와 코사인 함수를 이용하여 위치 정보를 생성합니다.

1. 위치 정보의 필요성:
Self-Attention 메커니즘은 단어의 순서에 대한 정보를 포함하지 않기 때문에, 위치 정보를 추가적으로 제공해야 합니다. 예를 들어, "I am happy"와 "happy am I"는 같은 단어들로 구성되지만, 순서에 따라 의미가 달라집니다. 이를 해결하기 위해, 각 단어의 위치 정보를 포함한 임베딩이 필요합니다.

2. Positional Encoding 계산:
Positional Encoding은 다음과 같은 방식으로 계산됩니다.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

여기서 pos는 단어의 위치, i는 임베딩 벡터의 차원 인덱스, d는 임베딩 차원입니다. 사인 함수와 코사인 함수는 주기적 성질을 가지며, 이러한 방식으로 생성된 위치 정보는 입력 시퀀스의 각 단어 벡터에 추가됩니다.

3. 위치 정보의 통합:
각 단어 벡터는 해당 단어의 Positional Encoding 벡터와 합쳐져 최종 임베딩 벡터를 형성합니다. 예를 들어, 입력 단어 벡터 X와 위치 벡터 PE가 있을 때, 최종 임베딩 벡터는 $X+PE$로 표현됩니다. 이를 통해 모델은 단어의 순서와 상대적 위치를 인식할 수 있게 됩니다.

Positional Encoding을 통해 Transformer 모델은 단어의 순서와 위치 정보를 효과적으로 통합하여, 시퀀스 데이터의 문맥적 의미를 정확히 이해하고 처리할 수 있습니다. 이러한 위치 정보의 통합은 Transformer가 긴 시퀀스를 처리하고, 문장의 구조적 의미를 유지하는 데 중요한 역할을 합니다.

Transformer 구조 분석

Encoder와 Decoder

Transformer 모델은 인코더(Encoder)와 디코더(Decoder) 두 주요 구성 요소로 이루어져 있습니다. 이 두 구성 요소는 각각 여러 개의 층(layer)으로 구성되어 있으며, 각 층은 다양한 신경망 구성 요소들을 포함하고 있습니다.

인코더(Encoder)

인코더는 입력 시퀀스를 처리하여 고차원 표현을 생성하는 역할을 합니다. 인코더는 입력 단어 시퀀스를 받아 이를 고차원 벡터로 변환하며, 여러 개의 인코더 블록으로 구성되어 있습니다. 각 인코더 블록은 다음과 같은 구성 요소로 이루어져 있습니다.

1. Self-Attention 레이어:
Self-Attention 메커니즘을 통해 입력 시퀀스의 각 단어가 다른 모든 단어와의 상호작용을 고려할 수 있게 합니다. 이를 통해 각 단어의 문맥적 의미를 파악합니다.

2. Feed-Forward 네트워크:
각 Self-Attention 레이어의 출력은 피드포워드 신경망에 전달됩니다. 이 신경망은 두 개의 선형 변환과 그 사이의 활성화 함수(ReLU)로 구성됩니다. 이는 입력 데이터를 비선형적으로

변환하여 모델의 표현력을 높입니다.

3. Residual Connections:
각 레이어의 출력을 다음 레이어의 입력에 더해줌으로써 정보의 흐름을 원활하게 하고 학습을 안정화합니다. Residual Connections은 다음과 같이 표현됩니다:

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

여기서 SubLayer(x)는 Self-Attention 또는 피드포워드 네트워크를 나타냅니다.

4. Layer Normalization:
각 레이어의 출력을 정규화하여 학습 과정에서의 불안정을 줄이고, 훈련 속도를 향상시킵니다. Layer Normalization은 각 단어 벡터의 평균과 표준편차를 이용하여 출력을 정규화합니다.

디코더(Decoder)

디코더는 인코더의 출력을 받아 목표 시퀀스를 생성하는 역할을 합니다. 디코더는 여러 개의 디코더 블록으로 구성되어 있으며, 각 블록은 다음과 같은 구성 요소로 이루어져 있습니다:

1. Self-Attention 레이어:
디코더의 Self-Attention 레이어는 목표 시퀀스의 각 단어가 이전 단어들과의 상호작용을 고려할 수 있게 합니다. 이 레이어는 미래 단어를 보지 못하도록 마스킹(masking)을 사용하여 순차적인 예측을 가능하게 합니다.

2. 인코더-디코더 Attention 레이어:
인코더의 출력과 디코더의 Self-Attention 레이어 출력을 결합하여, 목표 시퀀스의 각 단어가 입력 시퀀스의 관련 단어와 상호작용할 수 있게 합니다. 이는 디코더가 입력 시퀀스의 문맥을 고려하여 더 정확한 출력을 생성할 수 있게 합니다.

3. Feed-Forward 네트워크:
인코더와 마찬가지로, 디코더도 각 Attention 레이어의 출력을 피드포워드 신경망에 전달하여 비선형 변환을 수행합니다.

4. Residual Connections 및 Layer Normalization:
디코더에서도 Residual Connections과 Layer Normalization을 사용하여 정보의 흐름을 원활하게 하고, 학습의 안정성과 속도를 높입니다.

Multi-Head Attention

Multi-Head Attention은 여러 개의 Attention 헤드를 사용하는 방식입니다. 각 헤드는 서로 다른 부분에 집중하여 다양한 시각에서 입력 시퀀스를 분석할 수 있습니다. Multi-Head Attention의 주요 구성 요소와 동작 방식은 다음과 같습니다:

1. 여러 헤드의 사용:
여러 개의 Attention 헤드를 병렬로 사용하여, 입력 시퀀스의 서로 다른 부분에 집중할 수 있습니다. 각 헤드는 독립적으로 쿼리, 키, 밸류 벡터를 학습합니다.

2. 헤드의 병렬 처리:
각 헤드의 Attention 출력은 병렬로 계산됩니다. 이를 통해 모델은 입력 시퀀스의 다양한 문맥적 정보를 동시에 학습할 수 있습니다.

3. 출력 결합:
각 헤드의 출력을 결합하여 최종 출력을 생성합니다. 결합된 출력은 선형 변환을 거쳐 최종 Attention 출력을 형성합니다.
Multi-Head Attention은 정보 손실을 줄이고 모델의 성능을 향상시키며, 입력 시퀀스의 다양한 패턴을 효과적으로 학습할 수 있게 합니다.

Feed-Forward Networks

각 인코더와 디코더 블록에는 피드포워드 네트워크가 포함되어 있습니다. 이는 두 개의 선형 변환과 그 사이의 활성화 함수(ReLU)로 구성되며, 입력 데이터를 더 복잡한 형태로 변환하여 모델의 표현력을 높입니다.

1. 두 개의 선형 변환:
첫 번째 선형 변환은 입력 벡터를 고차원 공간으로 매핑합니다. 두 번째 선형 변환은 고차원 공간의 벡터를 다시 저차원 공간으로 매핑합니다. 이를 통해 입력 데이터의 복잡한 패턴을 학습할 수 있습니다.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

2. 활성화 함수:
두 선형 변환 사이에 ReLU 활성화 함수를 적용하여 비선형성을 도입합니다. 이는 모델이 더 복잡한 함수와 패턴을 학습할 수 있게 합니다.

Residual Connections & Layer Normalization

Residual Connections와 Layer Normalization은 Transformer 모델의 학습을 돕는 두 가지 중요한 기술입니다.

1. Residual Connections:
Residual Connections는 각 레이어의 출력을 다음 레이어의 입력에 더해줍니다. 이를 통해 정보의 흐름을 원활하게 하고, 기울기 소실 문제를 완화하여 깊은 신경망의 학습을 안정화합니다.

$$\text{Output} = \text{LayerNorm}(x + \text{SubLayer}(x))$$

2. Layer Normalization:
Layer Normalization은 각 레이어의 출력을 정규화하여 학습 과정에서의 불안정을 줄이고, 훈련 속도를 향상시킵니다. 각 단어 벡터의 평균과 표준편차를 이용하여 출력을 정규화함으로써, 모델이 더 안정적으로 학습할 수 있게 합니다.

이와 같이, Transformer 모델의 구조와 구성 요소들은 상호작용하며 고성능 자연어 처리 모델을 구현하는 데 중요한 역할을 합니다. 각각의 구성 요소는 입력 데이터의 문맥적 의미를 정확하게 이해하고, 모델의 표현력을 극대화하는 데 기여합니다.

결론

Transformer 모델은 NLP 및 다양한 분야에서 큰 혁신을 가져온 모델입니다. Self-Attention 메커니즘과 Positional Encoding을 통해 입력 시퀀스의 모든 부분 간의 상호작용을 고려하며, 인코더와 디코더 구조를 통해 복잡한 언어 이해 및 생성 작업을 수행합니다. Multi-Head Attention과 피드포워드 네트워크, Residual Connections, Layer Normalization 등의 기술을 활용하여 모델의 성능과 안정성을 극대화합니다. 이러한 특성들 덕분에 Transformer는 현대 NLP 연구 및 응용에서 핵심적인 역할을 하고 있습니다.