The Korean Journal of Applied Statistics
2024, Vol. 37, No. 5, 567–582

DOI: https://doi.org/10.5351/KJAS.2024.37.5.567
Print ISSN1225-066X / Online ISSN 2383-5818

A statistical journey to DNN, the third trip:

Language model and transformer

Yu Jin Kim[a], In Jun Hwang[a], Kisuk Jang[a], Yoon Dong Lee[1,a]

[a]Business School, Sogang University

Abstract

Over the past decade, the remarkable advancements in deep neural networks have paralleled the development and evolution of language models. Initially, language models were developed in the form of Encoder-Decoder models using early RNNs. However, with the introduction of Attention in 2015 and the emergence of the Transformer in 2017, the field saw revolutionary growth. This study briefly reviews the development process of language models and examines in detail the working mechanism and technical elements of the Transformer. Additionally, it explores statistical models and methodologies related to language models and the Transformer.

Keywords: language model, transformer, multi-head attention, encoder-decoder, positional encoding

1. 서론

인공지능 기술을 언급하지 않고 현대통계학을 거론하기 어려운 시대가 되었다. 본 논문에서는, 기본적인 심층신경망(deep neural net) 모형들을 통계적 관점에서 해석하여, 이들 각각을 통계학 교육의 한 주제로 편입할 수 있는 방안을 모색한다. 본 논문은 책임저자가 여러 관련 학회 등에서 2019년 이후 구두로 발표했던 내용을 문헌으로 정리한 여러 편의 논문 중 세 번째 편이다.

여러 가지 심층신경망 모형 중, 최근 가장 큰 관심을 끄는 모형은, 단연 BERT와 GPT 등 다양한 언어모형의 핵심 구성요소인 트랜스포머(transformer)다. 트랜스포머는 자연어 처리 기술과 함께 발전한 방법이다. 앞서 자연어 처리를 위한 다양한 연구가 이루어졌다. Mikolov 등 (2013a; 2013b), Cho 등 (2014), Sutskever 등 (2014), Bahdanau 등 (2015), Chorowski 등 (2015), Luong 등 (2015)의 연구는, Vaswani 등 (2017)이 트랜스포머를 개발하는 과정에서의 중요한 선행연구다.

Mikolov 등 (2013a; 2013b)는 명목변수인 단어를 임베딩(word embedding)하고, 차원축소 기술을 적용하여, 각 단어의 의미를 고차원 벡터공간에서의 점들로 대응시키는 방법을 제시하였다. Cho 등 (2014)과 Sutskever 등 (2014)은 RNN을 복합적으로 구성하여 encoder-decoder 모형을 만들고, 이를 학습시키는 방법으로 어절들을 고차원 벡터공간에 나타낼 수 있음을 보였다. 결국 단어는 벡터공간 상의 점인 벡터에 해당하고, 하나의 문장은 연속된 벡터들의 수열(sequence)로서의 의미를 갖게 된다. 번역이란 벡터들의 수열을 다른 벡터들의 수열로 바꿔주는 작업에 해당한다. Cho 등 (2014)과 Sutskever 등 (2014)이 제안한 RNN encoder-decoder (RED) 모형은, 수열을 수열로 변환하는 Seq2Seq 모형의 한 형태다. Vaswani 등 (2017)이 제안한 트랜스포머는 RED 모형에서, RNN을 어텐션(attention)으로 대체한 Seq2Seq 모형의 하나다.

[1]Corresponding author: Business School, Sogang University, PA 804, BaekBumRo, Mapo, Seoul 04107, Korea. E-mail: widylee@sogang.ac.kr

Published 31 October 2024 / journal homepage: http://kjas.or.kr

Table 1: Encoder-Decoder model using vanilla RNN

VRED : Vanilla RNN Encoder-Decoder	
$\boldsymbol{h}_j = [\boldsymbol{h}_{j-1}W_{hh} + e(\boldsymbol{x}_j)W_{eh}], \quad j = 2,3,\ldots,J, \quad h_1 = [e(\boldsymbol{x}_1)W_{eh}]$	(T.1)
$\boldsymbol{s}_0 = [\boldsymbol{h}_J W_{h0}], \quad \boldsymbol{s}_1 = [\boldsymbol{s}_0 W_{0s}], \quad \boldsymbol{y}_0^* = \boldsymbol{0}$	(T.2)
$\boldsymbol{s}_t = [\boldsymbol{s}_0 W_{0s} + \boldsymbol{s}_{t-1}W_{ss} + \hat{\boldsymbol{y}}_{t-1}W_{ys}], \quad t = 2,3,\ldots,T$	(T.3)
$\hat{\boldsymbol{y}}_t = [\boldsymbol{s}_0 W_{0s} + \boldsymbol{s}_t W_{sy} + \boldsymbol{y}_{t-1}^* W_{yy}]_*, \quad t = 1,2,\ldots,T$	(T.4)
$\boldsymbol{y}_t^* = \text{argmax}\ \hat{\boldsymbol{y}}_t, \quad t = 1,2,\ldots,T$	(T.5)

통계학에서 '요인분석'과 '주성분분석'은 자료의 차원을 축소하고 그 기하학적 의미를 찾는 방법이다. 요인분석은 다차원 변수로 이루어진 자료에서 다차원 변수가 갖는 의미를 낮은 차원의 잠재변수인 요인(factor)으로 축약하여 표현하는 방법이다. 구조방정식모형(structural equatuion model)은 다양한 요인들 사이의 관계를 구조화하여 표현한다는 점에서 encoder-decoder 모형과 그 맥락을 같이 한다. Encoder-decoder 모형에서는, 구조방정식모형(SEM)과 달리, 관측변수와 잠재변수들이 다차원 수열의 형태로 표현되는 내포적 구조를 가지고 있다는 점에서 차이가 있다. 또 입력된 자료의 의미 해석만을 목적으로 하지 않고, 디코더를 통하여 번역된 문장의 생성기능을 함께 수행한다는 차이점이 있다.

어텐션은 통계학에서의 커널평활(kernel smoothing)을 발전시킨 방법이다. 커널평활에서는 모수화된 커널을 미리 설정하고, 자료로부터 커널에 사용될 모수를 추정하는 방법으로 커널을 결정하고, 관측위치 사이의 관계를 커널로 수량화하고, 이를 이용하여 가중치를 적용한 이동평균을 구하는 방법이 통계학에서의 커널평활 방법이다. 어텐션도 동일한 접근법을 사용한다. 각 단어들 사이의 관계를 구하는 적절한 커널을 구하고, 이 커널을 이용하여 각 단어 사이의 관계를 파악하고 이를 반영하여 단어와 문장을 번역한다.

본 논문에서는 Seq2Seq 모형과 어텐션, 그리고 트랜스포머에 대하여 살펴보고, 이들 방법과 통계 방법과의 관련성을 살펴본다. 제2절에서는 RNN 기반 Seq2Seq 모형에 대하여 살펴보고, 제3절은 트랜스포머에 대하여 살펴본다. 다음으로 제4절에서는 트랜스포머와 RNN, CNN과의 관계를 살펴본다. 본 논문에서는 심층신경망을 수식으로 나타낼 때, Kim 등 (2024a)과 Kim 등 (2024b)에서 사용한 표현법과 동일한 표현법을 사용한다.

2. RNN 기반 Seq2Seq 모형

Mikolov 등 (2013a; 2013b)는 차원축소형 잠재변수를 갖는 심층신경망을 구성하고, 이를 적절히 훈련하는 방법으로 여러 단어들의 의미를 추출하여 고차원 공간에서의 벡터들의 집합으로 함축할 수 있음을 보였다. 즉, 임베딩(embedding) 과정을 통하여 각 단어에 대응하는 잠재변수들이 각 단어의 의미를 함축하는 벡터가 되고, 이 벡터들이 속한 벡터공간이 단어의 의미를 반영하는 '함축공간'이 된다. 문장은 함축공간(implicative space)에서의 벡터들로 이루어진 수열에 해당한다. '함축벡터'(implicative vector)의 수열도 벡터일 것이므로, 문장 자체가 더 큰 함축공간에서의 하나의 점에 해당한다. 문장은 하나의 수열 혹은 점에 해당하고, 번역이란 수열(sequence)을 다른 수열로 변환하는 작업, 혹은 점을 점으로 변환하는 작업에 해당한다.

Seq2Seq 모형은 수열을 수열로 바꾸는 기능을 하는 심층신경망이다. Seq2Seq 모형은 입력된 수열을 해석하는 인코더(encoder) 부분과, 해석된 의미를 새로운 수열로 출력하는 디코더(decoder) 부분으로 구성된다. RNN은 수열을 다루는 심층신경망이다. RNN은 수열에서의 순서를 정보흐름의 순서로 대응시킨다. RNN을 이용하여 원문장(source paragrahp)의 의미를 추출하는 인코더와, 이를 다른 언어로 번역한 번역문장(trans-

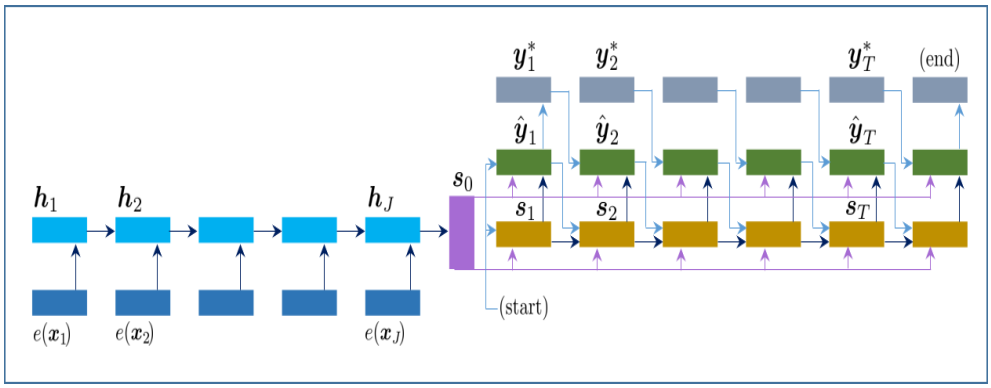

Figure 1: *Architecture of VRED.*

lated paragraph)을 생성하는 디코더를 구성하여 결합한 모형이 RED 모형이다.

Cho 등 (2014)와 Sustkever 등 (2014)가 제안한 encoder-decoder 모형은 대표적인 RED 모형이다. 본 논문에서는 Vanilla RNN을 이용한 encoder-decoder (VRED) 모형을 살펴볼 것이다. 구조가 단순한 VRED 모형을 통하여, RED 모형의 전체적인 작동방식을 간단하게 살펴볼 수 있다.

2.1. VRED 모형

원문장 $(\boldsymbol{x})$가 J개의 단어로 구성되어 있고, 번역할 언어의 대응문장 $(\boldsymbol{y})$가 T 개의 단어로 구성된다고 하자. 즉, $(\boldsymbol{x})$와 $(\boldsymbol{y})$는 다음과 같다.

$$(\boldsymbol{x}) = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_J) \quad \text{이고,} \quad (\boldsymbol{y}) = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T)$$

또 원문장 $(\boldsymbol{x})$를 각 단어별로 임베딩한 함축벡터열을 $(\boldsymbol{e}) = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_J)$ 라 하자. 즉, $\boldsymbol{e}_j = e(\boldsymbol{x}_j)$, $j = 1, 2, \ldots, J$ 이다. Table 1에는 함축벡터열 $(\boldsymbol{e})$로부터 번역문장 $(\boldsymbol{y}^*)$를 생성하기 위한 VRED 모형이 나와 있다.

Table 1에서 대괄호 $[\cdot]$는 적절하게 선정된 활성함수(주로, tanh, 항등함수, ReLU 등)을 의미하고, 아래첨자로 별표가 붙은 대괄호 $[\cdot]_*$는 softmax 함수를 의미한다. Table 1에서 (T.1)과 (T.3)에는 Vanilla RNN 구조가 적용되어 있다. Table 1에서 VRED는 인코더와 디코더로 분리해서 살펴볼 수 있다. 원문장의 의미 해석을 담당하는 인코더 기능은 (T.1)에서 이루어진다. 번역될 단어를 생성하는 디코더 기능은 (T.2) ~ (T.5)에서 이루어진다. (T.1)과 (T.3)는 잠재백터열, $(\boldsymbol{h})$와 $(\boldsymbol{s})$를생성하기 위한 순환식이다. 본 논문에서는 단어들로 이루어진 문장 혹은 벡터들의 수열을 표시하기 위하여, 괄호를 이용하여 표시한다. 예를 들어 문장 $(\boldsymbol{x})$는 여러 단어들로 이루어진 수열 $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_J)$를 의미하고, $(\boldsymbol{s})$는 벡터 $\boldsymbol{s}_t$ 로 이루어진 수열 $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T)$를 의미한다.

Figure 1은 VRED의 전체적인 구성을 그림으로 보여주고 있다. 인코더는 원문장 $(\boldsymbol{x})$를 받아, 임베딩을 통하여 함축벡터열 $(\boldsymbol{e})$로 바뀌고, 함축벡터열 $(\boldsymbol{e})$로부터 인코더를 통하여, 잠재벡터열 $(\boldsymbol{h})$를 얻고, 디코더를 통하여 $(\boldsymbol{h})$로부터 $(\boldsymbol{s})$를 얻고, $(\boldsymbol{s})$로부터 생성확률벡터열 $(\hat{\boldsymbol{y}})$과 생성문장 $(\boldsymbol{y}^*)$를 얻는 과정이 표현되어 있다. $\boldsymbol{s}_0$는 인코더의 마지막 잠재벡터 $\boldsymbol{h}_J$로부터 변환된 값으로, 잠재벡터열 $(\boldsymbol{s})$와 $(\hat{\boldsymbol{y}})$의 모든 원소벡터 각각에 대한 입력으로 사용된다.

VRED 모형은 실용성 있는 좋은 결과를 주는 모형도 아니고, 과거 중요한 역할을 한 모형도 아니다. 그럼에도 불구하고, 본 논문에서 VRED 모형을 언급하는 이유는 VRED 모형이 RED 모형 중 가장 단순한 모형인 이유로, 이를 통하여 RED 모형과 Seq2Seq 모형의 전체적인 구조와 작동방식을 쉽고 간단하게 살펴볼 수 있기 때문이다. Cho 등 (2014)이 제안한 RED 모형은 VRED 모형에서의 (T.3)과 (T.5)에 적용된 Vanilla RNN을

GRU로 대체한 모형에 해당하고, Sustkever 등 (2014)이 제안한 모형은 LSTM을 사용한 모형에 해당한다. Kim 등 (2024a)는 Vanilla RNN, LSTM, GRU의 관계에 대하여 정리한 바 있다.

2.2. 번역문장의 생성과 깁스표집법

VRED의 디코더 부분에서는 앞서 생성된 선행단어가 후행단어의 생성에 영향을 미칠 수 있는 구조가 적용되었다. 번역문장의 t 번째 위치에서 단어집(vocabulary)의 각 단어들이 생성단어가 될 확률은 softmax 함수를 통하여 $\hat{\boldsymbol{y}}_t$ 로 구해진다. $\hat{\boldsymbol{y}}_t$ 는 (T.3)과 같은 방법으로 다음 위치의 단어 생성 과정에서 $\boldsymbol{s}_{t+1}$ 에 영향을 미친다. 이후 $\hat{\boldsymbol{y}}_t$ 벡터의 값에 따라, 그 값이 가장 높은 단어를 선택하여 $\boldsymbol{y}_t^*$ 를 생성한다. 즉, $\boldsymbol{y}_t^*$ 는 $\hat{\boldsymbol{y}}_t$ 에 hardmax 함수를 적용하는 방법, 말하자면 greedy decoding 방법으로 생성한다. 생성된 단어 $\boldsymbol{y}_t^*$ 는 (T.4)과 같은 방법으로 $\hat{\boldsymbol{y}}_{t+1}$ 에 영향을 미친다.

디코더에서 번역문장 ($\boldsymbol{y}^*$) 의 생성을 위하여 사용하는 greedy decoding 방법은 다음과 같이 표현할 수 있다.

$$\boldsymbol{y}_t^* \;=\; \text{argmax}_{\boldsymbol{y}_t}\; P\big(\, \boldsymbol{y}_t \,|\, (\boldsymbol{y}^*)_{t-1}\,,\; (\boldsymbol{x}),\; \boldsymbol{\theta}\,\big)\,,$$

여기서 $(\boldsymbol{y}^*)_t = (\boldsymbol{y}_0^*, \boldsymbol{y}_1^*, \ldots, \boldsymbol{y}_t^*)$ 이다. 확률이 가장 높은 단어를 선택하여 생성하고, 생성된 단어를 다음 단어 생성과정에서 조건으로 대입한다. 이 과정은 깁스표집법(Gibb's sampling)에서 완전조건분포로부터 확률난수 $\boldsymbol{y}_t^*$ 를 뽑고, 이를 다음 단계의 완전조건분포(full conditional distribution)의 조건으로 대입하는 과정과 동일하다. 다만 $\boldsymbol{y}_t^*$ 를 완전조건분포에서 난수를 뽑아 생성하는 것이 아니고, 완전조건분포의 최빈값으로 대체한 것이다. 또 깁스표집법에서의 분포수렴을 달성하기 위한 반복생성 절차를 생략하였다. 즉, 확률난수열을 생성하기 위하여 사용하는 깁스표집법을 변형하여, 생성확률이 높은 원소벡터를(즉, 단어를) 순차적으로 생성하는 방법으로, 전체적으로 높은 확률을 갖는 수열 '하나'를 생성한 것이다. 말하자면, '대표 벡터난수열' 하나를 생성하는 방법으로 번역문장을 생성한 것이다.

3. Transformer에 대하여

Bahdanau 등 (2015)와 Luong 등 (2015)는, Cho 등 (2014)와 Sustkever 등 (2014)가 제안한 RED 모형의 성능을 개선하기 위하여, RNN과 함께 각자 추가로 고안한 어텐션(attention) 구조를 도입한 attention RNN encoder-decoder (ARED) 모형을 제안하였다. Bahdanau 등 (2015)은 인코더에 양방향 GRU를 사용하고, 디코더에 GRU와 함께 어텐션을 채용하였다. Luong 등 (2015)는 global attention과 local attention 두 종류의 어텐션을 동시에 고려하는 모형을 고안하였다.

Vaswani 등 (2017)이 제안한 트랜스포머(transformer)는 앞서 언급한 ARED 모형에서 RNN 구조를 제거하고 어텐션 구조만으로 인코더와 디코더를 구성한 모형이다. 논문의 제목을 "어텐션만 있으면 된다" (attention is all you need)라고 한 것은 RNN을 사용하지 말자는 의미다. 결국 트랜스포머는 RED에서 RNN을 어텐션으로 대체한 것이다. 트랜스포머는 원문장 ($\boldsymbol{x}$) 를 생성문장 ($\boldsymbol{y}^*$) 로 변환하는 작업을 수행하는 모형이고, 원문장 ($\boldsymbol{x}$) 를 함축벡터열 ($\boldsymbol{e}$) 로 변환하는 임베딩 과정도 모형에 포함되어 수행되고, 학습과정을 통하여 적절한 함축벡터열 ($\boldsymbol{e}$) 을 구하게 되지만, 본 논문에서는 편의상 트랜스포머의 기능을 함축벡터열 ($\boldsymbol{e}$) 로부터 생성문장 ($\boldsymbol{y}^*$) 을 생성하는 과정을 중심으로 살펴본다.

트랜스포머에서는 문장을 단어들로 이루어진 수열로 표현한다. 문장을 표현하기 위하여, 문장내 단어의 순서정보와 의미정보를 결합한 벡터들의 수열을 구성한다. 단어의 문장내 순서 j를 위치인코딩(positional encoding)을 통하여 위치벡터 $\boldsymbol{p}_j$ 로 변환한다. 위치벡터와 함축벡터 $e(\boldsymbol{x}_j)$ 를 결합한 결합벡터 $\boldsymbol{c}_j = c(\boldsymbol{p}_j, e(\boldsymbol{x}_j))$ 를 이후 과정에서의 입력으로 이용한다. 트랜스포머에는 위치인코딩(positional encoding) 방법과 어텐션 알고리즘이 사용되고, 효과적인 학습을 위하여 층별정규화(layer normalization)와 잔차연결(residual connection)

방법들이 함께 사용되고 있다. 다음에서는 먼저 이들 각각에 대하여 살펴보고, 이후 이들을 통합하여 구성한 트랜스포머 전체 구성에 대하여 살펴본다.

3.1. 어텐션과 그 변형 방법들

트랜스포머에서 어텐션은 구성방법에 따라 크로스어텐션(cross-attention)과 셀프어텐션(self-attention)으로 구분되고, 또 마스크어텐션(masked attention)과 마스크없는 (그냥)어텐션으로 구분된다. 트랜스포머는 다른 Seq2Seq 모형과 마찬가지로 인코더와 디코더 두 부분으로 구성된다. 인코더는 원문장을 입력받아 그 의미를 추출하는 역할을 하고, 디코더는 번역문장을 생성하는 역할을 한다.

인코더와 디코더는 어텐션층과 잔차연결, 층별정규화로 구성된 과정을 여러 번 반복하는 구조로 이루어진다. 크로스어텐션은 두 개의 벡터열 ($\boldsymbol{a}$)와 ($\boldsymbol{b}$)를 입력받아, 새로운 벡터열 ($\boldsymbol{d}$)를 출력한다. 셀프어텐션은 입력벡터열 ($\boldsymbol{a}$) 와 ($\boldsymbol{b}$)가 동일한 경우로 입력벡터열이 ($\boldsymbol{b}$) 하나다. 인코더에서 셀프어텐션을 처음 적용할 때는 원문장에서 추출한 결합벡터열 ($\boldsymbol{c}$)를 입력벡터열 ($\boldsymbol{b}$)로 받아 처리하고, 이후 반복과정에서는 앞에서 처리한 결과 ($\boldsymbol{d}$)를 다시 입력벡터열 ($\boldsymbol{b}$)로 받아 처리한다. 그 결과로 원문장의 의미를 함축하는 '문맥벡터열'을 구하여 디코더에 전달한다.

디코더에는 두 가지 종류의 어텐션이 개입된다. '마스크어텐션'과 '크로스어텐션'이다. 디코더가 번역문장을 생성할 때는 각 단어별로 순차적으로 하나씩 생성하고, 디코더에 입력되는 대응문장 중 생성할 단어 위치 이후의 단어는 마스킹하여, 각 단어의 생성과정에서는 대응문장의 선행위치의 단어들만으로 구성된, 선행어절이 입력으로 활용된다. 이를 위하여 '마스크어텐션'을 적용한다. 마스크어텐션은 선행어절을 입력받아 선행어절 문맥벡터열을 생성한다. 디코더의 구성부분 중 encoder-decoder attention은 인코더와 마스크어텐션 양 방향에서 문맥벡터열을 전달받아 '크로스어텐션'을 적용한다. 인코더로부터 원문장의 문맥벡터열을 전달받아 입력벡터열 ($\boldsymbol{b}$)에 할당하고, 마스크어텐션으로부터 선행어절 문맥벡터열을 전달받아, 입력벡터열 ($\boldsymbol{a}$)에 할당한다. 입력벡터열 ($\boldsymbol{a}$)로부터 질의벡터열을 생성하고, 입력벡터열 ($\boldsymbol{b}$)로부터 키벡터열과 값벡터열을 생성하여 어텐션을 적용한다.

3.1.1. 어텐션(attention)

어텐션은 선형필터를 도입하여 적응형(adaptive) 커널평활을 구현한 방법으로, 트랜스포머의 핵심 엔진이다. 구체적인 적용방법에 따라 셀프어텐션과 크로스어텐션으로 구분된다. 먼저 공간자료 혹은 시계열자료 등과 같이, 관측위치 $\boldsymbol{k}$에서 관측값 $\boldsymbol{v}$가 J 회 반복하여 관측된 자료 D가 있다고 하자. 자료 D는 다음과 같다.

$$D = \left\{ \left(\boldsymbol{k}_j, \boldsymbol{v}_j \right) \mid j = 1, 2, \ldots, J \right\}$$

또 위치 $\boldsymbol{k}$와 $\boldsymbol{q}$ 사이의 거리에 따른 가중값은 커널 $w(\boldsymbol{q}, \boldsymbol{k})$에 의하여 결정된다고 하자. 이 경우 관측자료 D로부터 얻게 되는 관심위치 $\boldsymbol{q}$에서의 평활값 $\boldsymbol{u}$는 다음과 같다.

$$\boldsymbol{u} = \sum_{j}^{J} w\left(\boldsymbol{q}, \boldsymbol{k}_j \right) \ \boldsymbol{v}_j .$$

유한 개의 관심위치, $\boldsymbol{q}_i,\ i = 1, 2, \ldots, I$ 에 대한 커널평활은 검색을 통한 테이블 통합(table aggregation) 과정으로 설명할 수 있다. 자료 D는 관측위치와 관측값 2개의 컬럼으로 구성된 테이블과 같고, 관측위치 $\boldsymbol{k}$는 테이블의 키(key)가 되고, 관측값 $\boldsymbol{v}$는 테이블의 값(value)이 된다. 테이블 D를 요약하여, 새로운 테이블 D^*를 만드는 경우를 상정하자. 이때 새로운 테이블 D^*의 키는 관심위치 $\boldsymbol{q}_i,\ i = 1, 2, \ldots, I$ 가 되고, 테이블 D^*의 값은 평활값 $\boldsymbol{u}_i = \boldsymbol{u}(\boldsymbol{q}_i)$ 가 된다. 즉, $D^* = \{ (\boldsymbol{q}_i, \boldsymbol{u}_i) \mid i = 1, 2, \ldots, I \}$ 이다. 이 과정을 데이터베이스 테이블 연산 관점으로 보면, $\boldsymbol{q}_i$는 질의(query)가 되고, $\boldsymbol{k}_j$는 키(key), $\boldsymbol{v}_j$는 값(value)이라고 의미를 부여할 수 있다.

3.1.2. 셀프어텐션과 크로스어텐션

다음의 크로스어텐션에 대한 설명에서 입력벡터열 ($\boldsymbol{a}$)를 ($\boldsymbol{b}$)로 대체하여, 입력벡터열이 ($\boldsymbol{b}$) 하나만 있는 경우로 바꾸면 셀프어텐션에 대한 설명이 된다. 크로스어텐션을 적용하는 과정은, 먼저 입력된 벡터열 ($\boldsymbol{a}$)와 ($\boldsymbol{b}$)의 각 원소벡터 $\boldsymbol{a}_i$, $\boldsymbol{b}_j$ 각각에 3종류의 선형필터, W_q, W_k, W_v 를적용하여, 다음과 같이, 질의벡터 $\boldsymbol{q}_i$, 키벡터 $\boldsymbol{k}_j$, 값벡터 $\boldsymbol{v}_j$ 를 구성한다.

$$\boldsymbol{q}_i = \boldsymbol{a}_i W_q, \quad \boldsymbol{k}_j = \boldsymbol{b}_j W_k, \quad \boldsymbol{v}_j = \boldsymbol{a}_j W_v.$$

다음으로, 질의벡터와 키벡터의 차원을 p 라 할 때, 즉 $p = \dim(\boldsymbol{q})$ 라 할 때, 질의벡터와 키벡터 사이에서 스케일내적(scaled dot-product)으로 어텐션스코어 $r_{ij} = \boldsymbol{q}_i \boldsymbol{k}'_j / \sqrt{p}$ 를 구하고, 어텐션스코어벡터 $\boldsymbol{r_i} = (r_{i1}, \ldots, r_{iJ})$ 에 softmax함수 $[\cdot]_*$ 를 적용하여(즉, $\boldsymbol{w}_i = [\boldsymbol{r}_i]_*$), 어텐션 가중값 $\boldsymbol{w}_i = (w_{i1}, \ldots, w_{iJ})$ 를 구한다. 이 때 $w_{ij} \geq 0$ 이고, $\sum_j w_{ij} = 1$ 이다. 어텐션가중값(attention weights)으로부터 다음과 같이 어텐션결과 $\boldsymbol{u}_i$ 를 얻는다.

$$\boldsymbol{u}_i = \sum_j^J w_{ij} \, \boldsymbol{v}_j. \tag{3.1}$$

어텐션결과(attention output)는 어텐션가중값을 이용하여 값벡터들의 가중평균을 구한 것이다.

3.1.3. 다중어텐션

다중어텐션은 어텐션을 동시에 여러 번 적용하는 것을 말한다. 하나의 어텐션에는 3개의 선형필터로 구성된 어텐션필터 $W_a = (W_p, W_k, W_v)$ 가 적용된다. 다중어텐션은, 하나의 층에 동시에 여러 개의 어텐션필터 $W_{al} = (W_{pl}, W_{kl}, W_{vl})$, $l = 1, 2, \ldots, L$ 을 적용하고, 마지막에 순방향전진층(feed forward layer)을 추가하여 각각의 어텐션에서 나온 결과를 묶어주도록 망을 구성한 것이다. 다중어텐션의 결과 $\boldsymbol{o}_i$ 는 다음과 같다.

$$\boldsymbol{o}_i = \left[(\boldsymbol{u}_{i1}, \ldots, \boldsymbol{u}_{iL}) \, W_o \right].$$

보통 다중어텐션에서 출력벡터 $\boldsymbol{o}_i$ 의 차원은 입력되는 입력벡터 $\boldsymbol{a}_i$ 의 차원과 같다. 다중어텐션에 적용되는 필터는 L 개의 어텐션필터 W_{al} 과 묶음필터 W_o 이고, 입력벡터 $\boldsymbol{a}_i$ 의 차원이 d이고, 키벡터의 차원이 p 라면, 다중어텐션 학습과정에서 사용되는 모수의 개수는 $3p(d+1)L + d(pL+1)$ 이다. 각각의 선형필터를 적용하는 과정에서 바이어스항(절편항)이 추가되어, $(d+1)$ 항과 $(pL+1)$ 항에 1이 더해진 것이다.

Tensorflow에서, MultiHeadAttention(...)(A, B)라고 하면, 텐서 A는 질의벡터를 생성하는 입력벡터열 ($\boldsymbol{a}$)에 대응되고, 텐서 B는 키벡터와 값벡터를 생성하는 입력벡터열 ($\boldsymbol{b}$)에 대응된다. Tensorflow의 MultiHeadAttention은 attention output과 attention score를 결과로 제공한다. Tensorflow에서 제공하는 attention score는 softmax 함수를 적용하여 가로방향 합이 1이 되도록 만든 attention weight 이고, 식 (3.1)에서의 w_{ij} 의 값들로 이루어진 행렬이다.

3.1.4. 마스크어텐션

'마스크어텐션'은 트랜스포머의 디코더 부분에서 적용되는 방법이다. VRED에서 디코더는 매 시점, 앞서 생성된 선행단어 $\boldsymbol{y}^*_{t-1}$ 가 후행단어 $\boldsymbol{y}^*_t$ 의 생성에 영향을 미치는 구조를 가지고 있다. 트랜스포머에서 셀프어텐션을 통해 동일한 구조를 구현한 것이 '마스크어텐션'이고, 트랜스포머를 구성하는 중요한 아이디어 중 하나다. 마스크어텐션은, 기준 t 시점에서, $\boldsymbol{y}^*_t$ 을 생성하는 과정에서, t 시점 이후의 정보는 이용하지 못하도록 마스킹(masking)하고, $(t-1)$ 이전 시점의 정보만을 이용하여 다중어텐션을 적용한다. 또 기준시점을 순차적으로 증가시키면서, 동일한 방법을 적용하여 번역문장 ($\boldsymbol{y}^*$) 전체를 생성한다.

3.2. 위치인코딩

동일한 단어로 이루어진 문장이라 할지라도, 단어의 순서에 따라 그 의미가 완전히 달라진다. 때문에 문장의 의미는 사용된 단어와 그 순서가 함께 입력되어야 한다. 이를 위해 각 단어를 문장 내에서 사용된 순서정보와 결합하여 표현하여야 한다. 각 단어 $\boldsymbol{x}_j$ 를 원핫인코딩(one-hot encoding)으로 표현하고, 이를 임베딩하여 함축벡터 $\boldsymbol{e}_j$ 로 바꾸고, 각 단어의 위치는 위치인코딩(positional encoding)을 통하여, 위치벡터 $\boldsymbol{p}_j$ 로바꾼다. 이후 각 단어의 함축벡터와 그 위치벡터를 결합하여 결합벡터 $\boldsymbol{c}_j = c(\boldsymbol{p}_j, \boldsymbol{e}_j)$ 를 구성한다. 위치인코딩 방법과 결합벡터를 구성하는 방법을 살펴보면 다음과 같다.

3.2.1. 이산퓨리에변환

위치벡터를 구성하는 가장 단순한 방법은, 원핫인코딩이다. 문장에 사용된 단어의 개수를 '문장의 길이'라 하자. '문장의 길이'가 J 라고 하고, 각 단어의 위치를 원핫인코딩으로 표현한 위치벡터를 $\boldsymbol{\delta}_j^p$ 라 하면, $\boldsymbol{\delta}_j^p$ 의 차원은(길이는) J이고, j 번째 원소만 1이고 나머지 원소는 모두 0이다. 즉, $\delta_{jj}^p = 1$ 이고 , $\delta_{jl}^p = 0,\ l \neq j$ 이다. 학습에 사용되는 문장의 길이가 매번 달라질 것이므로 위치벡터의 차원도 매번 달라진다. 이런 문제점을 해결하고, 또 이후 함축벡터와 위치벡터의 효율적인 결합을 위하여, 위치벡터 $\boldsymbol{\delta}_j^p$ 의 차원을 적절하게 고정할 필요가 있다. 이 경우 대안이 될 수 있는 방법이, 이산퓨리에변환(discrete Fourier transformation)이다. 트랜스포머에서는 이산퓨리에변환(DFT)를 이용하여 $\boldsymbol{\delta}_j^p = (\delta_{j1}^p, \ldots, \delta_{jJ}^p)$ 를 다음과 같이 변환한 위치벡터 $\boldsymbol{p}_j = (p_{j1}, \ldots, p_{jm})$ 를 이용한다.

$$p_{jk} \;=\; \sum_l \delta_{jl}^p \exp\left(2\pi i \omega_k l\right) \;=\; \exp\left(2\pi i \omega_k j\right) \;=\; \left(\cos\left(2\pi\omega_k j\right),\ \sin\left(2\pi\omega_k j\right)\right).$$

적당한 주파수 수열 $\omega_k,\ k = 1, 2, \ldots, m$ 에 대하여, 주파수공간에서의 위치벡터 $\boldsymbol{p}_j$ 의 길이는(차원은), 위치(순서)의 최대값 J 와 관계없이 $2m$ 으로 고정된다. 주파수 수열 ω_k 는 $(\omega_k > 0)$ 단조증가하거나 단조감소한다고 하자. 이산퓨리에변환이 갖는 여러 수리적 특성에 따라 위치벡터 $\boldsymbol{p}_j$ 는 여러 가지 장점을 갖는다. 예를 들면, $\|\boldsymbol{p}_j\| = 1$ 로 값의 범위가 유한하고, $\boldsymbol{p}_{i+j} - \boldsymbol{p}_j$ 값은 i 와 무관하고 항상 일정하다. 주파수 ω_k 가 작아질수록 $\boldsymbol{p}_{j+1,k}$ 와 $\boldsymbol{p}_{j,k}$ 의 차이 $\|\boldsymbol{p}_{j+1,k} - \boldsymbol{p}_{j,k}\|$ 는 작아지고, 비슷한 위치가 비슷한 위치벡터 값으로 인코딩된다.

3.2.2. 결합벡터의 구성

결합벡터는 위치벡터 $\boldsymbol{p}_j$ 와 함축벡터 $\boldsymbol{e}_j$ 의 합이 이용된다. 즉 , $\boldsymbol{c}_j = \boldsymbol{p}_j + \boldsymbol{e}_j,\ j = 1, 2, \ldots, J$ 이다. 함축벡터의 차원 d 를 짝수로 잡고, $m = d/2$ 로 하면, 위치벡터의 차원과 함축벡터의 차원이 같아지므로, 두 벡터의 합으로 결합벡터를 구성할 수 있다.

이산퓨리에변환을 이용하여, 위치벡터를 잡고, 위치벡터와 함축벡터의 합으로 결합벡터를 구성하는 방법 이외의 방법에 대한 대안과 연구가 제시되고 있다. 예를 들면, 위치 쌍 (j, l)을 고려하여 인코딩하는 방법 (Shaw, 2018), 문장의 구조적 특성에 따라 단어의 위치를 반영하는 방법 (Wang 등, 2019), 위치벡터에 별도의 필터를 넣어 위치벡터에 대학 학습을 가능하게 하는 방법 (Li 등, 2021), 결합벡터를 구성할 때, 위치벡터를 함축벡터에 대한 회전변환으로 표현하는 방법 (Su 등, 2023), 결합벡터를 위치벡터와 함축벡터의 연결(concatenate)로 구성하는 방법 (Zhou 등, 2024) 등에 대한 다양한 연구가 제안되고 있다. 그럼에도 불구하고, 이산퓨리에변환에 의하여 위치벡터를 잡고, 위치벡터와 함축벡터의 합으로 결합벡터를 구성하는 방법은, 여전히 가장 자주 사용되는 위치인코딩 방법이다. 그 이유는 이산퓨리에변환 과정에서 다양한 주파수에 의하여 충분히 위치 정보가 잘 반영되는 점 때문이고, 학습과정에서 어텐션필터에 의하여, 적절한 가중값이 적용되는 효과가 있기 때문이다.

3.3. 층별정규화와 잔차연결

어텐션과 관련된 사항 외에, 트랜스포머를 구성하는 주요 기술적 요소로는, 층별정규화(layer normalization)와 잔차연결(residual connection) 방법이 있다. 두 가지 방법 모두 처음 들어서는 이해하기 쉽지 않은 난해한 측면을 가지고 있다.

3.3.1. 층별정규화

심층신경망에서 정규화는 표준화를 의미하는 용어로 사용된다. 심층신경망에서 '정규화'라는 용어는, (그냥) 정규화 normalization, 배치정규화 batchnormalization, 층별정규화 layernormalization, 세 가지로 사용된다. 정규화(normalization)는 데이터 전체에서 각 변수들에 대한 평균과 표준편차를 이용한 표준화를 의미한다. 반면 '배치정규화'와 '층별정규화'에는 표준화 과정이 포함되기는 하지만, 전체적인 의미는 표준화와는 다르다. 조금 더 정확하게 표현하자면, '평균과 표준편차의 학습모수화'라고 표현하는 편이 더 타당하다.

층의 깊이가 깊은 심층신경망의 경우, 훈련 과정에서, 인접하는 층들이 서로 간섭하여 학습 속도가 느려지는 internal covariate shift (ICS) 현상이 발생한다. ICS 현상을 방지하기 위하여, Ioffe와 Szegedy (2015)는 배치정규화 방법을 제안하였고, Ba 등 (2016)은 배치정규화를 개선하여 층별정규화 방법을 제안하였다. 배치정규화와 층별정규화는 각 변수별로 평균과 표준편차를 학습가능한(learnable) 모수로 대체하여 후속 과정에 대입하는 방법이다. 작업 대상이 되는 층 $\boldsymbol{h} = (h_1, \ldots, h_k)$ 의 전체 변수(노드) 각각에 대한 평균과 표준편차를 '학습가능한 모수'로 대체한 $\tilde{\boldsymbol{h}}$ 로 변환하는 방법이다. 그 구체적인 방법은 다음과 같다. 먼저 각 변수들에 대하여, 평균은 0이 되고, 표준편차는 1이 되도록 표준화한다. 즉 $z_1 = (h_l - \mu_l)/\sigma_l$, $l = 1, 2, \ldots, k$ 을 구한다. 이후 표준화된 변수 z_l 에 학습가능한 모수 γ_l 과 β_l 를 도입하여, $\tilde{h}_l = \gamma_l z_l + \beta_l$ 을 구하고, 이로부터 $\tilde{\boldsymbol{h}} = (\tilde{h}_1, \ldots, \tilde{h}_k)$ 를 구성하여, 후속 과정에 대입한다.

배치정규화의 경우, 표준화를 위하여 μ_l 와 σ_l 을 구하는 방법으로, 동일 배치 내에서 각 변수의 평균과 표준편차를 이용한다. 각 배치자료를 모두 처리하기 전 각 변수의 평균과 표준편차를 미리 알 수 없으므로, 평균과 표준편차에 대한 이동평균으로 대체하여 처리한다. 반면, 층별정규화에서 표준화는 각 층 내부의 값만을 이용해서 이루어진다. 즉, μ_l 와 σ_l 는, k 개의 값 h_l, $h = 1, 2, \ldots, k$ 의 평균과 표준편차이다. 층별정규화 작업에서는, 배치자료 전체를 다 볼 필요가 없이, 각 층에 입력되는 값만 있으면 충분하다. 시간축 방향으로 층이 계속 반복되는 RNN의 경우 배치정규화를 적용하기 위해서는 까다로운 작업이 필요하지만 층별정규화는 쉽게 적용이 가능하다. 또 배치 크기가 1인 online 학습의 경우라면 배치정규화는 적용이 불가능하지만 층별정규화는 적용이 가능하다.

층별정규화가 가능한 이유는 잠재변수들의 동질성 때문이다. 심층신경망에 입력하는 입력변수나 출력변수의 경우, 그 입력노드(출력노드)의 위치에 따라 일정한 값이 입력(출력)되므로, 각 노드들에는 특정한 의미가 부여된다. 그러나 은닉층에서의 각 노드들은 그 위치에 의하여 의미가 특정되지 않는다. 은닉층에서의 노드들의 의미는, 가중치로 표현되는 입력층(출력층) 노드들과의 관계로만 특정되는데, 이는 고정적인 값이 아니고, 의미를 부여하기 어려운 값이다. 따라서 동일한 은닉층에 속한 각 노드들은 전체적으로 동일한 자격을 갖춘 변수들이다. 그러므로 동일한 자격을 갖는 여러 변수들을 표준화하는 작업이 가능하다.

3.3.2. 정규화에 필요한 모수의 개수

Tensorflow에서 정규화 층을 적용할 때마다, 모수의 개수가 추가된다. Normalization 층의 경우, 학습에 사용되는 데이터의 변수의 개수를 k 라 하면, 사용되는 모수의 개수는 $2k + 1$ 이다. 모수가 필요할 것 같지 않은 데이터 전체를 대상으로 하는 '정규화'에 모수의 개수가 추가되어 당혹스러울 수 있지만, 잘 보면, 이때 사용되는 모수는 모두 non-trainable 모수다. 모수의 개수 $2k + 1$ 은 각 변수별로 구한 평균과 표준편차의 개수에, 표준편차로 나누는 과정에서 0으로 나누는 것을 방지하기 위한 수치적 목적으로 사용하는 미소값 ϵ 하나를

추가한 개수다.

Batchnormalization 층의 경우, 각 배치에 적용되는 각 변수별로 평균과 표준편차 보정을 위한 모수 2개가 필요하고, 평균과 표준편차에 대한 배치들 사이의 이동평균 구하기 위하여 추가로 2개의 모수가 사용되어, $4k$ 개의 모수가 학습에 사용된다. Layernormalization 층의 경우 해당 층의 노드의 개수를 k 라 할 때, 배치정규화와 달리, 변수들의 평균과 표준편차에 대한 이동평균을 고려할 필요 없이, 해당 노드들의 각 시점 평균과 표준편차를 직접 사용하면 되므로, 각 변수별로 사용하는 모수의 개수 2개가 줄어, 학습에 필요한 모수의 개수는 $2k$ 가 된다.

3.3.3. 잔차연결 방법

잔차연결법은 He 등 (2015)이 제안한 방법으로, 심층신경망의 깊이를 획기적으로 증진시켜, 심층신경망이 성공적 결과를 낼 수 있게 만든 중요한 기술이다. 최근에도 그 작동원리를 밝히려는 연구가 계속 진행 중이다. 본 논문에서는 저자 나름의 시각으로, 다음과 같이 잔차연결법의 의미와 작동원리를 설명한다.

Kim 등 (2024b)은 MLP를 예로 들어, 순방향전진망을 $[\,[[\boldsymbol{x}_0\, A_0]\, A_1]\, A_2]\,]$ 와 같은 형태로 표현할 수 있다고 하였다. 이를 다시 표현하면, $\boldsymbol{x}_{l+1} = [\boldsymbol{x}_l\, A_l]$, $l = 0, 1, 2$ 가 된다. 여기서 $\boldsymbol{x}_{l+1} = [\boldsymbol{x}_l\, A_l]$ 인 관계는 후행층 $\boldsymbol{x}_{l+1}$ 을 선행층 $\boldsymbol{x}_l$ 으로 모형화 한다는 의미를 갖는다. 동일한 경우에 잔차연결법을 도입하면, $\boldsymbol{x}_{l+1} = [\boldsymbol{x}_l A_l] + \boldsymbol{x}_l$ 와 같이, 만능 변환의 결과에 입력값을 다시 더하는 형태가 된다. 우변을 이항하면 다음과 같다.

$$\boldsymbol{x}_{l+1} - \boldsymbol{x}_l = [\boldsymbol{x}_l A_l]\,,$$

즉, 후행층 $\boldsymbol{x}_{l+1}$ 과 선행층 $\boldsymbol{x}_l$ 을 차분하여(differencing) 얻은 잔차 $(\boldsymbol{x}_{l+1} - \boldsymbol{x}_l)$ 를 선행층 $\boldsymbol{x}_l$ 으로 모형화 한다는 의미다.

순방향전진망(feed-forward network)에서, 두 개 층을 함께 표현하면 $\boldsymbol{x}_{l+1} = [\,[\boldsymbol{x}_{l-1}\, A_{l-1}]\, A_l\,]$ 이 된다, 잔차연결법이 적용된 경우라면, 다음과 같이 확장된다.

$$\boldsymbol{x}_{l+1} = [\,([\boldsymbol{x}_{l-1}\, A_{l-1}] + \boldsymbol{x}_{l-1})\, A_l\,] + \boldsymbol{x}_l$$

후행층 $\boldsymbol{x}_{l+1}$ 을 모형화 할 때, 선행층 $\boldsymbol{x}_l$ 뿐만아니라, $\boldsymbol{x}_{l-1}$ 도 이용된다. 여러 층에 대한 잔차연결법을 고려하면 더 많은 항들이 포함되어 확장되고, 잔차연결법을 적용하는 경우, 이전 여러 층을 동시에 기저(bases)로 사용하여 모형을 구성하는 것과 같은 효과가 발생한다. 이와 관련하여, 잔차연결법에서는 일종의 부스팅(boosting) 효과가 발생하여 장점이 된다고 설명하기도 한다 (Veit 등, 2016; Siu, 2019).

잔차연결법의 장점을 그래디언트(gradient)를 구하는 과정을 통하여 설명할 수도 있다. 층이 깊은 심층신경망의 경우, 경사소실(vanishing gradient)이나 경사폭발(exploding gradient) 현상으로 신경망 학습에 어려움이 발생한다. 역전파알고리즘을 고려하면, 그래디언트를 구하는 과정에 층간그래디언트 $(\partial \boldsymbol{x}_{l+1})/(\partial \boldsymbol{x}_l)$, $l = 1, 2, \ldots$ 의 곱이 개입된다. 잔차연결법이 사용된 경우라면, 다음 관계가 성립한다.

$$\frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l} = I + \frac{\partial [\boldsymbol{x}_l A_l]}{\partial \boldsymbol{x}_l},$$

이 경우 층간그래디언트가 단위행렬을 포함하는 식으로 표현되므로, 더 넓은 범위에 속하는 가중행렬 A_l이, 경사소실이나 경사폭발 현상에 대한 저항성을 가질 수 있다고 보인다.

3.4. 트랜스포머의 전체 구성

트랜스포머는 앞서 언급한 다양한 요소들이 결합되어 구성되는 시스템으로 전체적인 구성을 간단하게 표현하기 쉽지 않다. 때문에 이를 그림으로 표현하는 방법이 자주 이용된다. Vaswani 등 (2017)의 Figure 1과 Figure

2는 트랜스포머의 전체적인 구성을 가장 잘 보여 주는 그림이다. 또 Park 등 (2019)의 Figure 2는 다중어텐션의 작동방식을 그림으로 잘 표현하고 있다. 이와 같은 그림 표현은 전체적인 구성은 잘 표현하지만, 구체적인 내용이 모호해지는 단점이 있다. 이를 보완하기 위하여, 본 논문의 부록에 Algorithm 1을 실었다. Algoritm 1은 Vaswani 등 (2017)이 Figure 1으로 표현한 트랜스포머의 구성을 Python 언어를 빌어 알고리즘 형태로 표현한 것이다.

4. 셀프어텐션과 **RNN, CNN**의 비교

4.1. 필터의 역할

셀프어텐션은 여러 결합벡터 $\boldsymbol{c}_j = \boldsymbol{p}_j + \boldsymbol{e}_j,\ j = 1, 2, \ldots, J$ 에 포함된 정보를 추출하기 위하여, 어텐션필터 $W_a = (W_q, W_k, W_v)$ 를 사용한다. 값필터 W_v 는 결합벡터 내부의 정보를 추출하는 역할을 하고, 질의필터 W_q 와 키필터 W_k 는 벡터들 사이의 정보를 추출하는 역할을 한다. 이와 같은 관점으로 셀프어텐션을 RNN 혹은 CNN과 비교하면 다음과 같다.

셀프어텐션은 RNN의 대안적 방법으로 고안된 방법이다. 셀프어텐션을 $\boldsymbol{h}_j = [\boldsymbol{h}_{j-1}W + \boldsymbol{e}_j U]$ 인 순환적 구조를 갖는 Vanilla RNN과 비교하자. 이때 행렬 U 는 함축벡터 $\boldsymbol{e}_j$ 각각의 내부의 정보를 추출하는 필터 역할을 하고, 행렬 W 는 순환적 구조 하에서 함축벡터들 사이의 정보를 추출하는 필터 역할을 한다. 말하자면, Vanilla RNN에서의 행렬 U 의 역할은 셀프어텐션에서 값필터 W_v 의 역할에 대응되고, Vanilla RNN에서의 행렬 W 의 역할은 질의필터 W_q 와 키필터 W_k 의 역할에 대응한다.

셀프어텐션의 구조와 역할은 CNN과 대비시켜 살펴볼 때 더욱 쉽게 파악할 수 있다. CNN에서는 합성곱(convolution)과 풀링(pooling) 두 가지 과정이 적용된다. 합성곱 과정은, 원자료의 일정부분(이를 '블럭'이라 하자)에 '합성곱필터'를 적용하여 가중합을 구하는 작업을 하고, 이 작업을 블럭을 옮겨가면서 반복하여 적용하는 과정이다. 풀링은, 여러 개의 블럭에 대하여 합성곱을 적용하여 얻은 결과를 통합하는 과정이다. 풀링의 경우 보통 최대값풀링(max pooling) 혹은 평균값풀링(average pooling)이 이용된다. 평균값풀링은, 풀링 대상이 되는 모든 값들에 대하여 동일한 가중치를 부여하여 평균을 구한다. 평균값풀링에서 동일한 가중치를 부여하는 대신, '가중치필터'(weight filter)를 도입하여, 적절한 가중치에 따라 평균을 구하는 '가중평균풀링'(weighted average pooling)을 고려하면, CNN과 셀프어텐션의 관계는 더욱 명확해진다.

CNN에서 합성곱에 사용되는 '합성곱필터'는 블럭(즉, 벡터) 내부의 정보를 추출하는 역할을 하고, '가중평균풀링'에서 사용하는 '가중치필터'는 블럭단위로 추출된 정보를 통합하는 역할을 한다. 셀프어텐션의 '값필터'의 역할은 '합성곱필터'에 대응하는 역할이고, '질의필터'와 '키필터'의 역할은, '가중치필터'에 대응하는 역할이다. CNN의 합성곱층에서는 여러 개의 필터를 이용하여 이미지가 갖는 다양한 특징을 찾아낸다. 마찬가지로 셀프어텐션을 적용할 때도 다중어텐션을 적용하여, 여러 가지 특징이 추출될 수 있도록 한다.

Model 1: 2MHA model in Keras

```
inputs = Input(shape=(49, 16))
    x = MultiHeadAttention(num_heads=16, key_dim=4)(inputs,inputs)
    x = MultiHeadAttention(num_heads=16, key_dim=4)(x, x)
    x = Flatten()(x)
    x = Dense(128, activation='relu')(x)
outputs = Dense(10, activation='softmax')(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

Table 2: Performance comparison of networks

	3D-NLP	2CP-CNN	Bi-LSTM	Bi-GRU	**2MHA**
# of parameters	242,762	18,106	321,738	244,682	**110,378**
Training Time	14.044초	16.050초	67.171초	67.163초	**27.072초**
accuracy	0.9769	0.9845	0.9829	0.9844	**0.9710**
AUC	0.99956	0.99988	0.99973	0.99979	**0.9994**

4.2. 이미지 분류의 사례

LSTM이나 GRU와 같은 RNN은 각각 순차성을 갖는 자료를 해석하는 용도로 사용되고, 그 중요한 응용으로 문장번역에서 encoder-decoder 모형에 사용된다. Transformer의 경우, 처음 발표될 때, 문장번역을 목적으로 하는 encoder-decoder 모형으로 발표되었으나, 문장번역 이외에도 사용이 가능하다. 대표적인 용도가 이미지 처리다. Parmer 등 (2018)은 트랜스포머를 이미지 변환에 적용하는 image transformer를 발표하였다. Dosovitskiy 등 (2018)은 transformer를 이용한 이미지 분류 방법을 제안하였다. Transformer라는 단어가 인기를 얻다보니, 최근에는 '다중어텐션'이 사용되는 다양한 목적의 심층신경망에 모두 트랜스포머라는 이름을 붙이고 있다.

본 논문에서는, Dosovitskiy 등 (2018)이 제안한 방법에 따라, MNIST 데이터의 이미지를 분류하는 작업을 수행하고, Kim 등 (2014a)에서의 LSTM, GRU 모형 등의 경우와 비교하고자 한다. 본 논문에서는, Kim 등 (2014a)에서 시험한 조건을 가능한 동등하게 유지하고, 비슷한 수준으로 매우 단순한 형태의 트랜스포머를 구성하여 시험하였다. MNIST 데이터 이미지를 4×4 픽셀 단위로 분리하여 7×7 패치로 나누어 트랜스포머를 적용하였다. 본 논문에서는 다중어텐션 층 2개가 있고, 이후 노드의 수가 (128, 10)인 밀집층(dense layer) 2개를 갖는 MLP 모형(2MHA)을 시험하였다. 2MHA에서 사용한 다중어텐션에서 키벡터의 차원(p), 헤드의 개수(L)는, 여러 가지 경우를 간단히 시험하여, 그 중 $p = 4$ 와 $L = 16$ 인 경우를 선정하여 그 결과를 제시하였다. 다른 경우들도 약간씩의 차이가 있었으나, CNN, RNN 등의 경우와 비교하면, 대체로 비슷하여, 한 경우만을 선정하여 그 결과를 제시하였다. Model 1은 Tensorflow의 Keras로 작성된 2MHA 모형의 구성이다.

Table 2에는 Kim 등 (2024a)에 제시된 결과에 2MHA 모형의 경우를 추가한 것이다. 2MHA 모형은 Bi-LSTM 이나 Bi-GRU 모형보다 parameter 개수가 약간 적고 학습 속도는 빠르지만 이미지 분류 성능은 약간 떨어진다. Table 2의 결과로만 보면, 트랜스포머를 이용한 2MHA 모형이 다른 모형들에 비하여 장점이 있다고 보이지는 않는다. 그 이유는, 2MHA 모형이 매우 단순하게 구성된 모형이기 때문이기도 하고, 시험에 사용된 MNIST 데이터가 분류가 매우 잘 되는 쉬운 데이터이기 때문에 비교되는 다른 신경망들도 상대적으로 좋은 분류 성능을 보여주기 때문이다. Dosovitskiy 등 (2018)은, ImageNet, CIFAR-100, VTAB 등 고난도 이미지 분류 시험자료에서 트랜스포머로 구성한 이미지 분류기가 매우 우수한 성능을 보이고 있음을 제시하고 있다.

5. 결 론

심층신경망 중에서 최근 가장 큰 관심 대상이 되는, transformer를 중심으로 언어모형(language model)에 대하여 간략하게 살펴보았다. 특히 이들 모형들과 통계적 방법들과의 관련성에 대한 부분을 강조하여 살펴보았다. 트랜스포머를 포함한 다양한 Seq2Seq 모형에서 디코더는, 단순한 통계적 추론의 기능만 포함되어 있는 것이 아니고, 단순화된 난수생성 기능이 함께 구현되어 있음을 살펴보았다. 또 트랜스포머의 핵심 알고리즘인 다중어텐션은 커널평활에 학습필터를 끼워 고안한 방법임을 살펴보았다. Kim 등 (2024a)와 Kim 등 (2024b)에서 다양한 심층신경망들은 회귀분석에 기초하여 발전한 방법임을 살펴보았다. 회귀분석이 만들어 놓은 현대의 발전된 심층신경망 기술은 눈이 부실 정도다. 심층신경망이 발전하는 만큼 통계학과 통계학자들의 적극적인

발전에 대한 기여와 활용, 그리고 다양한 분야에 심층신경망을 활용하기 위한 연구가 기대된다 (Hwang 등, 2024).

Appendix:

Algorithm 1: Architecture of transformer

```

def transformer_encoder(input_tokens, N, d_model, num_heads, d_ff):
    # Step 1: Embedding and Positional Encoding
    embeddings     = embed(input_tokens)  # Shape:(seq_len, d_model)
    pos_encodings = positional_encoding(seq_len=len(input_tokens),
                    d_model=d_model)
    x = embeddings + pos_encodings

    # Step 2: Encoder Layers
    for _ in range(N):
        # Multi-Head Self-Attention
        attn_output = multi_head_attention(Q=x, K=x, V=x,
                            num_heads=num_heads, d_model=d_model)
        x = layer_norm(x + attn_output)   # Add & Normalize
        # Feedforward Network
        ff_output = feedforward(x, d_ff=d_ff)
        x = layer_norm(x + ff_output)     # Add & Normalize

    return x  # Sequence of hidden states


def transformer_decoder(target_tokens, encoder_output, N, d_model,
    num_heads, d_ff):
    # Step 1: Embedding and Positional Encoding
    embeddings = embed(target_tokens)  # Shape: (seq_len, d_model)
    pos_encodings = positional_encoding(seq_len=len(target_tokens),
                    d_model=d_model)
    x = embeddings + pos_encodings

    # Step 2: Decoder Layers
    for _ in range(N):
        # Masked Multi-Head Self-Attention
        masked_attn_output = multi_head_attention(Q=x, K=x, V=x,
              num_heads=num_heads, d_model=d_model, mask=True)
```

```
        x = layer_norm(x + masked_attn_output)  # Add & Normalize

        # Multi-Head Encoder-Decoder Attention
        enc_dec_attn_output = multi_head_attention(Q=x,
               K=encoder_output, V=encoder_output,
               num_heads=num_heads, d_model=d_model)
        x = layer_norm(x + enc_dec_attn_output) # Add & Normalize

        # Feedforward Network
        ff_output = feedforward(x, d_ff=d_ff)
        x = layer_norm(x + ff_output)           # Add & Normalize

    return x  # Sequence of decoded tokens


def transformer(source_tokens, target_tokens, N, d_model, num_heads,
     d_ff):
    # Step 1: Encode
    encoder_output = transformer_encoder(source_tokens, N, d_model,
        num_heads, d_ff)

    # Step 2: Decode
    decoder_output = transformer_decoder(target_tokens,
        encoder_output, N, d_model, num_heads, d_ff)

    # Step 3: Generate final output (logits or probabilities)
    output = generate_output(decoder_output)

    return output  # Predicted sequence
```

References

Bahdanau D, Cho K, and Bengio Y (2016). Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, Available from: https://arxiv.org/abs/1409.0473

Ba JL, Kiros JR, and Hinton GE (2016). Layer normalization, Available from: pre-print: arXiv:1607.06450

Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, and Bengio Y (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 1724–1734.

Chorowski JK, Bahdanau D, Serdyuk D, Cho K, and Bengio Y (2015). Attention-based models for speech recognition, *Advances in Neural Information Processing Systems*, **28** (NIPS 2015), 577–585.

He K, Zhang X, Ren S, and Sun J (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 770–778.

Hwang IJ, Kim HJ, Kim YJk, and Lee YD (2024). Generalized neural collaborative filtering, *The Korean Journal of Applied Statistics*, **37**, 311–322.

Ioffe S and Szegedy C (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, Lille, 448–456.

Kim HJ, Kim YJ, Jang K, and Lee YD (2024a). A statistical journey to DNN, the second trip: Architecture of RNN and image classification, *The Korean Journal of Applied Statistics*, **37**, 553–563.

Kim HJ, Hwang IJ, Kim YJ, and Lee YD (2024b). A statistical journey to DNN, the first trip: From regression to deep neural network, *The Korean Journal of Applied Statistics*, **37**, 541–551.

Li Y, Si S, Li G, Hsieh CJ, and Bengio S (2021). Learnable fourier features for multi-dimensional spatial positional encoding, *Advances in Neural Information Processing Systems*, **34** (NeurIPS 2021), 15816–15829.

Luong MT, Pham H, and Manning CD (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, 1412–1421.

Mikolov T, Chen K, Corrado G, and Dean J (2013a). Efficient estimation of word representations in vector space, Available from: pre-print: arXiv:1301.3781

Mikolov T, Sutskever I, Chen K, Corrado G, and Dean J (2013b). Distributed representations of words and phrases and their compositionality, *Advances in Neural Information Processing Systems*, **26** (NIPS 2013), 3111–3119.

Park C, Na I, Jo Y *et al.* (2019). SANVis: Visual Analytics for Understanding Self-Attention Networks. In *Proceedings of 2019 IEEE Visualization Conference (VIS)*, Vancouver, BC, 146.

Parmar N, Vaswani A, Uszkoreit J, Kaiser Ł, Shazeer N, Ku A, and Tran S (2018). Image transformer. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, Stockholmsmässan, Stockholm, 4052–4061.

Shaw P, Uszkoreit J, and Vaswani A (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*, New Orleans, Louisiana, 464–468.

Siu C (2019). Residual networks behave like boosting algorithms. In *Proceedings of 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Washington DC, 31–40.

Su J, Ahmed M, Lu Y, Pan S, Bo W, and Liu Y (2024). Roformer: Enhanced transformer with rotary position embedding, *Neurocomputing*, **568**, 127063.

Sutskever I, Vinyals O, and Le QV (2014). Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems*, **27** (NIPS 2014), 3104–3112.

Vaswani A, Shazeer N, and Parmar N (2017). Attention is all you need, *Advances in Neural Information Processing Systems*, **30** (NIPS 2017), 5998–6008.

Veit A, Wilber MJ, and Belongie S (2016). Residual networks behave like ensembles of relatively shallow networks, *Advances in Neural Information Processing Systems*, **29** (NIPS 2016), 550–558.

Wang X, Tu Z, Wang L, and Shi S (2019). Self-attention with structural position representations. 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), Hong Kong, 1403–1409.

Zhou X, Ren Z, Zhou S, Jiang Z, Yu TZ, and Luo H (2024). Rethinking position embedding methods in the transformer architecture, *Neural Process Letters* **56**, 41.

Received July 31, 2024; Revised August 9, 2024; Accepted August 12, 2024

심층신경망으로 가는 통계 여행, 세 번째 여행:

언어모형과 트랜스포머

김유진[a], 황인준[a], 장기석[a], 이윤동[1,a]

[a]서강대학교 경영학부

요 약

지난 10년의 기간 심층신경망의 비약적 발전은 언어모형의 개발과 그 발전을 함께 해 왔다. 언어모형은 초기 RNN을 이용한 encoder-decoder 모형의 형태로 개발되었으나, 2015년 attention이 등장하고, 2017년 transformer가 등장하여 혁명적 기술로 성장하였다. 본연구에서는 언어모형의 발전과정을 간략하게 살펴보고, 트랜스포머의 작동원리와 기술적 요소에 대하여 구체적으로 살펴본다. 동시에 언어모형, 트랜스포머와 관련되는 통계모형과, 방법론에 대하여 함께 검토한다.

주요용어: 언어모형, 트랜스포머, 다중어텐션, 인코더-디코더, 위치인코딩

[1]교신저자: (04107) 서울시 마포구 백범로 바오로관 804호, 서강대학교 경영학부. E-mail: widylee@sogang.ac.kr