중앙고등학교

ML과 LLM의 작동 원리

인공지능 기초 심화탐구 보고서

2025-11-06

황태준
중앙고등학교
2025
2학년 7반 31번

목차

Abstract

인공지능은 말그대로 인간이 컴퓨터로 만든 지능이다. 그런데, 지능이란 무엇인가? 지능이 있다는 것은 생각할 수 있다는 것인가? AI는 생각하는가? 겉으로 보기에는 윤리적 판단도 어느 정도 하고, 수학 문제도 풀어내고, 각종 사소한 건에 대해 물어보면 나름 합리적인 판단도 내려주므로 생각하는 것처럼 보이기도 한다. 하지만 중학교에서 알고리즘을 처음 배울 때, 컴퓨터는 일정한 형식의 명령으로 계산(computation)만 빠르게 할 수 있는 바보라고 하지 않았는가? 구체적이고 특정적이지 않은, 모호한 인간의 자연어를 인공지능은 어떻게 처리하고 이해하여 그로부터 양질의 출력을 제공하는가?

지능(intelligence)의 정의는 아직도 제대로 내려지지 못하고 있다. 하지만 여러 정의에서 공통적으로 언급되는 부분은, 정보나 기술을 획득하여 목표 성취에 적용하고 새로운 것을 창조한다는 부분이다. 이 과정에서는 학습이 필수적이다. 정보나 기술을 획득하여 저장하고 이해(?)하는 과정이 학습이기 때문이다. 지능을 어떻게 정의 내리냐에 따라 컴퓨터가 지능을 가질 수 있는지 없는지는 철학적인 문제가 되겠지만, 적어도 지능을 모방할 수는 있을 것이다.

인공 신경망은 이러한 발상에서 비롯한 기술이자 자료구조로, 일종의 그래프이다. 정점[1]은 뉴런의 신경세포체를, 간선은 축삭을 본따 그 역할을 대신한다고 하는데, 생명과학을 배운 입장에서 이게 정말 맞는 말은 아닌 것 같고, 일반인들의 이해를 돕기 위한 비유적 설명으로 받아들일 수 있겠다.

정점들은 계층별로 모여 있는데, 직접적인 입력이 들어오는 입력 계층, 실질적인 처리가 일어나는 은닉 계층, 출력이 일어나는 출력 계층으로 나뉜다. 입력 계층과 출력 계층은 하나이며, 은닉 계층은 개수 제한은 없으나 그 개수와 해당 계층에서의 정점 개수를 어떻게 하느냐에 따라 학습 결과 성적이 좌우된다. 각 계층 간 정점은 수많은 간선들이 연결하고 있다. 최적화 알고리즘에서 모식적 모형으로 쓰이는 것들과 비슷하게, 여러 정점으로 가는 각 간선에는 서로 다른 가중치가 부여되어 있다. 인공 신경망을 통한 학습의 목적은, 궁극적으로 이 가중치들을 적절히 조절하여 주어진 입력에서 알맞은 출력을 내는 것이다. 하지만 수많은 간선들에 대해 일일이 수동으로 가중치를 설정할 수 없으니, 원하는 결과로부터 가장 가까운 출력을 내도록 역추적을 하는 방법을 사용하여 자동으

[1]앞으로 노드(node)와 혼용할 것인데, 뭔가 문맥 상 자연스러운 어휘가 다르기 때문이다.

로 학습을 진행한다. 이 방법을 역전파(backpropagation)이라고 하고, 이를 실현하기 위해 다변수 미적분학과 선형대수학의 손을 빌린 수학적 아이디어를 경사하강법(gradient descent)라고 한다.

하지만 인공 신경망은 생각보다 오래된 기술이고, 단점이 명확하다. 연산의 효율성은 둘째치고, 인공 신경망은 지능이 아니라 특정 목적 달성을 위해서만 단련된 시스템이기 때문에, 예를 들어 개와 고양이 사진을 구별하는 인공 신경망에 사람의 사진을 넣으면 무조건 틀린 답을 내놓게 된다. 사람은 개도 고양이도 아무것도 아닌 것[2]도 아니기 때문이다. 인공 신경망은 그 태생적 한계 때문에, 범용적이 될 수 없다.

현재는 거대 언어 모델(LLM)[3]의 시대이다. 물론 기초적인 형태의 언어 모델은 특정 목적을 위해서만 만들어지기도 했지만, 구글의 '트랜스포머' 모델 개발 이후 문장 전체의 문맥을 파악할 수 있게 된 언어 모델은 급격히 범용 인공지능을 향해 달려나가 발전하고 있다. 지금 유명한 모든 또는 거의 모든 인공지능 언어 모델은 트랜스포머에 기반한다.

I. 인공 신경망

II. LLM과 트랜스포머

II.1. 개요

LLM은 특히 요즘들어 정말로 생각을 하고 있는 것처럼 보이지만, 사실 그렇지 않다. 현재까지 기계학습의 본질은 크게 다르지 않기 때문이다. LLM은 방대한 학습 자료를 바탕으로 이전 단어(들)를 입력받아 다음 단어로 나올 확률(확실한 정도)가 가장 높은 것을 뱉는 프로그램이다. GPT와 같은 챗봇을 사용할 때는 모델이 자기가 뱉은 단어를 다시 입력해 반복적으로 한 단어씩 뱉는 과정이 반복되는 것이다. 챗봇 UI에서 단어가 타자치듯 생겨나거나 페이드 인 되는 효과는 단순 연출이 아니라 모델이 한 번에 하나의 토큰을 예측하기 때문에 생기는 것이다. 따라서 LLM은 인공신경망으로 구현된 복잡한 하나의 매개변수 함수일 뿐이다.

[2] default 또는 empty 출력 정점
[3] 대규모 언어 모델? 그냥 편하고 엘엘엠이라고 하겠습니다.

LLM을 정확하게 만드는 방법은 단순하다. 언어 모델이 언어를 “정확”하게 구사하게 하려면 단순히 엄청난 양의 데이터를 때려넣으면 된다. LLM에는 인공신경망에서처럼 가중치가 있는데, 정말 수없이 많이 있다. 마찬가지로 처음에는 모든 가중치가 무작위의 값이다가, 데이터를 받아 학습하면서 알맞은 결과를 내도록 조정해 나가게 된다.

학습 방법은, 간단히 예를 들어 실제 문장에서 마지막 단어만 뺀 것을 입력한 뒤 출력되는 결과를 받아 가중치를 수정해 실제 문장의 마지막 단어를 출력할 수 있도록 한다. 이 방법에 대해서는 앞서 역전파와 경사하강법 부분에서 다루었다. 이것을 수천억번 반복하면 학습한 문장은 물론, 처음 보는 문장에 대해서도 더 합리적인 예측을 하게 된다. LLM을 학습시키는데는 엄청난 컴퓨팅 자원이 필요한데, 1초에 10억 번 사칙연산이 가능한 컴퓨터로 적당히 합리적인 문장을 뽑아내는 LLM을 만들기 위해서는 학습시키는데만 1억 년 정도가 걸린다고 한다. 그래서 이러한 막대한 연산을 할 수 있는 하드웨어인 GPU[4]가 부상하기 시작했다. 이제는 왠만한 GPU로도 부족하니 NPU, TPU 등 행렬곱 연산에 최적화된 맞춤 하드웨어를 만들어내고 있다.

기본적으로 LLM은 아무 문장이나 받아서 합리적인 문장을 잇는 것을 잘하면 된다. 하지만 좋은 LLM과 좋은 AI 챗봇의 조건은 다르다. 사용자의 요구를 알아듣고, 사실에 기반해 대답하고, 일어나지 않은 일에 대해 단서를 가지고 추론할 수 있어야 한다. 이를 위해 RLHF(Reinforced Learning with Human Feedback, 인간 피드백을 통한 강화학습)을 수행한다. 사람들이 직접 모델의 응답을 평가해 비용 함수를 구성하여 인간 선호도에 맞게 응답을 조정하는 것이다. 이 과정에서 원치 않은 대답을 걸러내거나, 원하는 대답으로 유도하게 된다.

그런데, 사실 GPU가 주목받기 시작한 것은, LLM의 연산 자체를 병렬로 처리할 수 있게 된 트랜스포머 모델의 등장 때문이다. 이전까지는 LLM은 한 번에 한 단어만 읽어오며 문장을 입력받았다. 하지만 구글이 2017년 발표한 트랜스포머는 문장의 단어들을 한꺼번에 읽어온다.

모든 LLM은 결국 언어를 숫자로 바꾸어야 하는데, 이때 숫자는 자기 나름의 의미를 나

[4]고성능 GPU는 동시에 수많은 연산을 병렬로 하기 위해 연산 코어를 아주 많이 때려넣어놓은 연산 장치로, 게임 그래픽, 3D/동영상 렌더링, 유체 시뮬레이션 등을 목적으로 만들어졌다.

타내는 값들로 표현된다[5]. 트랜스포머는 각 단어를 단순히 자기가 학습한 일반적인 의미에 따른 숫자로 변환하는 것뿐만 아니라, attention 이라는 과정을 수행해 문장 내에서 각 단어의 상호작용을 고려해 각 단어가 문맥적으로 어떤 의미를 가지는지도 파악할 수 있게 된다. 예를 들어 "배가 아프다"의 '배'와 "배를 타다"의 '배'는 다른 의미를 가지고 있는데, 트랜스포머는 문장 전체를 보아 각각이 어떤 의미인지를 명확히 알 수 있다는 뜻이다.

의미를 기반으로 수치 벡터를 형성하고 attention을 거친 트랜스포머는 일반적으로 결과 벡터를 사전 학습된 인공신경망에 순전파하는 과정을 거친다. 순전파를 하여 단순한 선형결합인 attention 출력 결과를 시그모이드나 ReLU 같은 활성함수에 통과시켜 비선형적으로 만듦으로써 복잡한 표현을 조성한다. 즉, 언어적 능력(capacity)을 향상시킨다. 이 출력값을 다시 받아 attention을 수행하고 또 순전파를 하고… 하는 것을 충분히 반복한다. 이렇게 반복하는 과정에서 모델은 어떤 방향으로 단어를 사용해 대답을 내놓아야할지 결정한다. 마침내 마지막에는 입력 문장의 모든 단어에 영향을 받아 수치가 조정된 가장 마지막 단어에 해당하는 수치벡터를 바탕으로 다음 단어를 예측하게 된다.

기계학습은 이렇게 엄청난 반복을 통한 확률적인 원리로 이루어지므로 학습 알고리즘은 인간이 설계하지만, 실제로 학습이 일어나는 과정과 학습 데이터를 기반으로 예측할 때 거치는 구체적 과정들은 그저 학습 데이터에 끼워맞춘 가중치에 의해 일어나는 무작위적인 현상에 불과하다. 이러한 특성은 모델이 특정 상황에서 특정 예측을 내놓는 원인을 파악하는 것을 매우 어렵게 한다.

II.2. 트랜스포머

본래 트랜스포머는 구글에 근무하던 8명의 연구원이 쓴 "Attention Is All You Need"라는 논문에서 처음 제안되었다. 제목은 비틀즈의 노래인 "All You Need Is Love"를 패러디한 것이며, 처음 제안된 트랜스포머 구조의 목적은 영어와 독일어 간 번역이라고 한다. 이 트랜스포머는 AI 업계에 엄청난 파급력을 행사했고 AI 붐을 일으켰다. 현재 우리가 쓰는 ChatGPT, Gemini, Claude, 딥시크, LLaMA, Grok 등등의 챗봇 모델은 모두 트랜스포머 기반이다.

[5]인경신공망의 경우와 마찬가지로 학습 알고리즘에 의해 결정된 의미나 특징의 기준들이 우리의 직관으로는 이해할 수 없을 가능성이 높다.

> The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.[6].
>
> — Attention Is All You Need (2017), Ashish Vaswani, et al.

이처럼 트랜스포머는 번역 작업 이외에도 음성인식, TTS, 그림 인식, 그림 생성, 일반적인 LLM 등의 역할을 범용적으로 수행할 수 있다. 트랜스포머의 작동원리를 좀 더 깊게 알아보자.

트랜스포머는 먼저 입력을 일정 단위로 쪼개는데, 이때 쪼개진 것들을 토큰이라고 한다. 토큰은 입력이 문자열일 때는 보통 단어이거나 일반적으로 자주 등장하는 문자 조합 따위이다. 사진이나 소리라면 입력을 일정 단위로 쪼갠 샘플이 토큰이 된다. 이 토큰은 각각 (여기서는 숫자의 배열의 의미에 가까운) 벡터로 할당되는데, 이 벡터들은 숫자로서

[6]현재 주류인 시퀀스 변환(sequence transduction) 모델들은 인코더-디코더 구조에서 복잡한 순환 신경망(RNN) 또는 합성곱 신경망(CNN)에 기반하고 있다. 현재 가장 우수한 성능을 내는 모델들은 또한 어텐션을 통해 인코더와 디코더를 연결한다. 우리는 '트랜스포머'라고 이름붙인 새로운 단순한 신경망 구조를 제안한다. 이는 전적으로 어텐션에만 기반하며, 순환(recurrence)과 합성곱(convolution)을 완전히 없앴다. 기계 번역 과제에 대한 두 개의 실험에서, 이 모델이 질적으로 더 우수할 뿐 아니라, 병렬화가 더 잘 되고 학습에 드는 시간도 현저히 적게 든다는 것을 보였다. 우리 모델은 WMT 2014 영어-독일어 번역 과제에서 BLEU 점수 28.4를 달성하였으며, 이는 (앙상블 모델들을 포함한) 기존의 최고 성능 결과보다 2 BLEU 이상 향상된 수치이다. 또한 WMT 2014 영어-프랑스어 번역 과제에서는 단일 모델 기준으로 BLEU 점수 41.8을 달성하며 최고 성능 신기록을 갱신했다. 이 모델은 8개의 GPU로 3.5일 동안 학습됐으며, 이는 현존 문헌에 보고된 최고 모델들의 학습 비용의 극히 일부에 불과하다. 영어 구문 분석 과제에 적용했을 때 대규모 학습 데이터와 제한된 학습 데이터 모두에서 우수한 결과를 얻음을 근거로, 트랜스포머는 다른 일반적인 작업도 잘 수행할 수 있음을 밝힌다 (필자 의역).

토큰과 관련된 모종의 의미를 나타내게 된다. 이 의미의 기준을 우리가 알 수는 없겠지만 이 벡터가 아주 고차원의 공간에 있다고 상상하면, 비슷한 의미를 가진 단어들은 공간 상에서의 위치가 비슷할 것이라고 이해할 수 있다.

다음으로는 트랜스포머의 핵심인 어텐션 과정을 거친다. 어텐션은 문장 속 각 단어들 간 상호작용을 통해 문장의 문맥 속에서 단어가 어떤 의미를 가지는지를 더욱 특정하고 조정한다. 어텐션 이후에는 의미의 벡터를 순전파 신경망(다중계층 퍼셉트론이라고도 한다)에 보내 의미를 구체화하고 고차원적으로 만드는데, 이 과정에 대해서는 나중에 더 자세히 알아보겠다. 이 모든 작업은 거대한 행렬과 벡터 간의 행렬곱의 형태로 나타난다.

어텐션과 순전파를 충분히 반복한다면 문장의 마지막 벡터에 문장 전체의 의미가 충분히 녹아들었다고 할 수 있고, 이때 그 마지막 벡터에 특정 연산을 수행해 가능한 모든 토큰 중 문장 다음에 올 것을 예측하는 확률분포를 도출할 수 있다. 여기서 가장 확률이 높은 것(또는 조금의 무작위성을 주어 두세번째로 높은 것)을 골라 문장에 넣고 이 전체 문장을 다시 입력하면 트랜스포머 모델이 재귀적으로 돌아가며 출력을 뽑아낸다.

이런 시스템을 챗봇으로 사용하는 가장 간단한 방법은, 미리 다음과 같은 프롬프트를 주고

> 아래는 사용자와 유능하고 매우 지능적인 AI 조수의 대화이다.
> 사용자: (a)

사용자의 입력을 (a) 부분에 넣은 뒤 LLM에게 저 글 전체를 던져주면 그 다음에 오는 출력이 곧 챗봇의 응답이 되는 것이다. 이 방식은 지금도 쓰이고 있다. 이 원리를 안다면 요즘 챗봇 서비스에서 제공하는 개인 설정 프롬프트를 작성할 때 어떻게 작성하면 효과적인지도 알 수 있을 것이다.

II.3. 단어 임베딩

토큰은 꼭 단어 단위로 나뉘는 것이 아니지만, 이해를 위해 단어 단위로 나눈다고 하겠다[7]. 트랜스포머 모델은 자기가 학습한 모든 단어의 의미[8]를 행에 저장한 거대한 행렬을

[7] 문장부호, 어근과 접사, 어간과 어미, `<|endoftext|>` (GPT-3의 경우 응답을 끝낼 지점)도 모두 토큰이다.

하나 갖는데, 각 토큰에는 고유한 인덱스 ID가 붙어있어 이 행렬의 몇번째 행에서 단어의 의미를 찾아야하는지를 지시한다.

$$W_E \in \mathbb{R}^{V \times d} \tag{2.1}$$

이 행렬을 임베디드 행렬(embedded matrix)라고 하고, 보통 W_E로 표시한다. V는 학습한 어휘의 총 개수, d는 의미 벡터의 차원 수이다. 이 행렬의 값들은 기계학습의 모든 다른 값들처럼 처음에는 무작위로 시작하지만 학습 과정에서 조율된다. 토큰을 입력받으면 임베디드 행렬에서 그 의미를 받아와 토큰에 초기값으로 할당하게 된다.

그런데, 단어가 숫자로 표현되는 원리는 뭔가? 이 방법은 사실 기계학습에서는 트랜스포머나 LLM의 등장 이전에도 존재했는데, 이것을 임베딩이라고 한다. 단어의 의미 벡터는 아주 많은 기저로 이루어진 고차원 공간 상에 있다. 이 공간 상에서 단어들은 반복된 학습을 통해 비슷한 의미를 가진 것끼리 비슷한 경향성을 띄게 되고, 특정 의미 기준이 특정한 방향을 의미하게 된다.

충분히 학습되었다고 할 때, 이 벡터들 간에는 흥미로운 관계가 있다. Em(단어)를 "단어"에 대한 의미 벡터라고 하자. 이때 예를 들자면 아래와 같은 관계가 성립한다.

$$\begin{aligned} &\mathrm{Em}(\text{아버지}) - \mathrm{Em}(\text{어머니}) \approx \mathrm{Em}(\text{남자}) - \mathrm{Em}(\text{여자}) \\ \approx\ &\mathrm{Em}(\text{이모부}) - \mathrm{Em}(\text{이모}) \approx \mathrm{Em}(\text{고모부}) - \mathrm{Em}(\text{고모}) \approx \cdots\cdots \end{aligned} \tag{2.2}$$

즉, "남자 부모"를 뜻하는 단어를 모를 때, 아래와 같이 그 단어(또는 비슷한 단어)를 알아낼 수 있다.

$$\mathrm{Em}(\text{아버지}) \approx \mathrm{Em}(\text{어머니}) + \mathrm{Em}(\text{남자}) - \mathrm{Em}(\text{여자}) \tag{2.3}$$

3Blue1Brown의 예시를 빌리자면, 이 관계는 단수-복수 관계에서도 성립한다. 예를 들어 복수를 뜻하는 plural의 p를 따

$$\mathbf{p} := \mathrm{Em}(\text{cats}) - \mathrm{Em}(\text{cat}) \tag{2.4}$$

라고 한다면 다음과 같은 관계가 성립한다.

[8] 이제부터 '의미'라고 하면 우리가 이해하는 음성적, 연상적 의미가 아니라 기계학습에서 벡터로 표시되는 수치적인 의미를 말한다.

$$\mathbf{p} \cdot \mathrm{Em}(\text{students}) \gg \mathbf{p} \cdot \mathrm{Em}(\text{student}) \tag{2.5}$$

흥미로운 점은, 이것이 꼭 단수-복수 단어의 관계에서만 보이는 경향이 아니라는 것이다.

$$\mathbf{p} \cdot \mathrm{Em}(\text{one}) \ll \mathbf{p} \cdot \mathrm{Em}(\text{two}) < \mathbf{p} \cdot \mathrm{Em}(\text{three})... \tag{2.6}$$

이런 식으로, 비슷한 의미 기준은 벡터공간 상에서 같은 방향으로 정렬되어 있는 경향을 보인다. 사실 이 모든 것이 선형회귀의 원리의 연장선 상에 있다고 생각하면 이해가 편한 것 같다.

어텐션 같은 연산을 받는 경우, 임베딩은 꼭 단어의 의미에만 의존하는 것이 아니다. 예를 들어 문장에서 단어의 위치도 벡터를 조작할 수 있으며, 그 단어를 수식하는 단어에 의해 의미가 조작될 수도 있다. 이런 식으로 의미는 더욱 더 특정화, 구체화되고 풍부하게 변하게 된다.

모델이 글을 읽게 되면 먼저 임베딩 벡터에서 단순히 단어의 의미를 가져온다. 하지만 트랜스포머 모델의 목표는 각 단어가 개별적으로 의미하는 것보다 더 구체적이고 풍부한 의미를 얻는 것이므로 이를 위해 어텐션을 수행한다. 인공지능 모델은 (당연하게도) 정해진 수의 벡터로만 연산할 수 있는데, 이런걸 context size 또는 token size라고 한다. 이 연산에 대해서는 후술하겠다.

마지막에는 문장의 끝에 있는 의미 벡터를 전체 단어의 행렬로 투영하는 행렬 연산을 적용한다. 이렇게 출력된 벡터에 softmax라고 하는 정규화 연산을 적용해 합이 1인 확률분포를 만든다. 이때 행렬 연산으로 작용하는 행렬을 역임베딩 행렬(unembedding matrix)라고 하고 W_U로 표기한다. 이 행렬의 값 또한 처음에는 무작위 값이다가 학습을 통해 조정된다.

softmax 연산의 일반식은 아래와 같다.

$$x_k \leftarrow \frac{\exp(x_k)}{\sum_i \exp(x_i)} \tag{2.7}$$

즉 이것이 벡터에 작용하면 이런 모양이다.

$$\text{softmax}\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \frac{\exp(x_1)}{\sum_{i=1}^{N}\exp(x_i)} \\ \vdots \\ \frac{\exp(x_N)}{\sum_{i=1}^{N}\exp(x_i)} \end{bmatrix} \tag{2.8}$$

하지만 항상 가장 확률이 높은 것을 선택하는 것보다는 적당히 높은 것도 선택하는 것이 창의성, 다양성 같은 걸 증대할 수 있으므로, 온도[9]라고 불리는 상수 T를 추가해 무작위성을 줄 수 있다.

$$x_k \leftarrow \frac{\exp\left(\frac{x_k}{T}\right)}{\sum_i \exp\left(\frac{x_i}{T}\right)} \tag{2.9}$$

T가 클수록 더 작은 값에도 더 큰 확률을 부여해 확률 값을 평준화시킨다. 이때 softmax 연산을 적용하기 전의 언임베딩 연산을 거친 마지막 토큰의 의미벡터를 logit이라고 하기도 한다.

II.4. 어텐션

앞서 보았던 '배'의 예시로 계속하자면, 아래 문장들에서 '배'의 의미는 모두 다름을 우리는 알 수 있다.

- 배(腹)가 아프다.
- 배(船)가 항구를 떠난다.
- 얘, 겨울 배(梨)가 맛있단다!
- 두 배(倍) 증가했다.

우리는 이 '배'의 의미를 문장 주변의 단어들로 알 수 있다. 이 예시에서는 주로 서술어로 그 의미를 유추할 수 있다. 하지만 언어 모델이 이런 문장들을 처음 입력받을 때, 토큰 '배'는 모두 같은 의미 Em(배)로 표현된다. 어텐션은 이 배의 의미를 주변 토큰들과 상호연산 시킴으로써 구체화시키는 역할을 한다.

또, 주어진 프롬프트로 그림을 생성하는 AI에게 아래 구를 입력했다고 해보자.

- 경비행기 프라모델

[9] 열역학에서 따온 비유이다. 열역학에서 온도 T는 입자의 평균 운동에너지 $K = \frac{f}{2}k_B T$, 엔트로피(무질서도) $\mathrm{d}S = \frac{\delta Q}{T}$ 등을 나타낸다.

‘비행기’라는 말을 들으면 보통 대형 민항 여객기나 작은 구식 프롭기를 떠올리는 것이 일반적이다. 하지만 앞에 접사 ‘경(輕)-’이 붙으므로서 소형 비행기(세스나 社의 경비행기가 시장에서 압도적이므로 보통 세스나-172 같은 모델이 생성된다)라는 의미가 특정된다. 뒤에 오는 프라모델이 이 모든 걸 수식받음으로써 실제 비행기가 아니라 작고 장난감같아 보여야한다는 정보가 추가된다. 보통 프라모델, 미니어쳐, 장난감 등의 말을 이런 거대한 물체에 대한 말에 추가로 붙이게 되면 AI는 물체에다가 냅다 키링을 달아버리고 밝은 빛과 바닥이 있는 환경에 던져놓는다(이번 예시에서는 경비행기라는 물체가 너무 구체적이었는지 키링을 달지는 않았다.).

Figure 2.1 — 비행기, 경비행기, 세스나 경비행기, 경비행기 프라모델 이미지 생성 결과 (Google Imagen 4 via Gemini 2.5 Flash assistant)

II.5. 임베딩과 어텐션의 작동 원리

이제 수학적으로 들어가 보자. 어텐션을 설명할 예시 문장으로 “얘, 겨울 배가 맛있단다!”를 채택하도록 하겠다.

앞서 토큰은 단어라고 했지만 그건 영어 화자들에게 잘 먹혀들어가는 설명이다. 한국어, 일본어 같은 교착어나 독일어, 러시아어 같은 교착어는 어미와 접사의 역할이 중요하므로 단어 또는 어절 단위로 나누어지지 않는다. 굳이 말하면 (토크나이저 알고리즘에 따라 다르겠지만) 토큰은 형태소에 가깝다고 하는게 좋을 것 같다. SentencePiece나 BPE 같은 토크나이저를 사용한다면 아래와 같이 쪼갤 수 있다.

```
["얘", ",", "겨울", "배", "가", "맛", "있", "단", "다", "!"]
```

이제 이 단어들은 W_E에 의해 의미 벡터로 대치된다. 예를 들어 ‘겨울’의 ID가 666이라고 하면

$$\mathrm{Em}(\text{겨울}) = W_E[666] = \mathbf{e}_{\text{겨울}} \in \mathbb{R}^d \tag{2.10}$$

또, 트랜스포머는 입력을 병렬로 처리하기 때문에 토큰의 순서를 제대로 알 수 없는데, 어텐션을 위해서는 순서를 알아야 하므로 이를 위해 임베딩된 의미 벡터에 추가적으로 위치 정보를 추가하게 된다. 이 과정을 positional encoding이라고 한다. 위치 p(입력 상에서 토큰의 위치 인덱스)와 차원 $i(d/2-1$과 같은 꼴의 값)에 대해 이런 식으로 삼각함수를 이용하여 위치 정보를 표현한다.

$$\begin{aligned} \mathrm{PE}(p, 2i) &= \sin\left(\frac{p}{10000^{2i/d}}\right) \\ \mathrm{PE}(p, 2i+1) &= \cos\left(\frac{p}{10000^{2i/d}}\right) \end{aligned} \tag{2.11}$$

이것은 원 논문에서 제시된 방법으로, PE에 삼각함수를 이용하는 이유에는 몇 가지가 있다. 첫번째로, 이 PE 정보 자체가 의미 벡터에 들어가야하기 때문에 미분이 가능해야 해서 단순 정수를 사용할 수 없다. 사인곡선은 $\mathbb{R}$에서 연속이고 미분가능할 뿐 아니라 파동으로서의 주기성과 위상 또한 가지고 있으며, 삼각함수의 덧셈정리를 이용해 비교적 용이하게 $\sin(a+b)$와 같은 선형결합을 계산할 수 있다. 즉, 토큰의 위치(p)가 변하면 삼각함수 값에 의해 벡터가 부드럽게 회전하므로 내재적으로 연속이면서 순서 정보를 함의한 값을 만들 수 있다. 또, 순서에 대한 별도의 추가적인 학습 없이 모델이 순서 "감각"을 갖출 수 있으므로 유용하다. 차원 i는 최종 결과에서 삼각함수의 주기를 달리하여, 입력 내 토큰 간 거리 스케일에 따른 의미관계를 다양하게 표현할 수 있도록 한다. 즉, 하위 차원은 국소적인 간격에서의 세밀한 위치변화를, 상위 차원은 전체 입력 문장 구조에서의 긴 거리 간 위치변화를 담당한다.

설명이 난해한데, 정리하자면 PE를 삼각함수로 작성한 것은 토큰 간 순서, 상대적 간격, 문맥적 구조를 파동의 위상 회전이라는 내재적인 수학적 특성으로 표현하여 합, 내적 등 간단한 연산만으로 어순과 맥락을 쉽게 알아낼 수 있도록 하기 위한 것이다.

우리의 문장에서 '겨울'이 $p=2$이므로[10] 예시 계산식으로 계산해보면

$$\mathrm{PE}_{\text{겨울}} = \begin{bmatrix} 0.002 \\ 0.998 \\ \vdots \end{bmatrix} \tag{2.12}$$

같이 된다.

[10] 인덱스는 0부터 센다.

이제 최종 입력 벡터는 의미와 PE를 합친

$$\mathbf{x}_p = \mathbf{e}_{\text{token}} + \text{PE}_p \tag{2.13}$$

가 된다.

이런 식으로 의미벡터가 모인 전체 입력은 이런 식으로 생겼을 것이다.

$$X = [x_0, x_1, \ldots, x_{n-1}] \tag{2.14}$$

각각의 입력 벡터들은 이제 세 개의 서로 다른 연산자와 결합해 선형변환을 받는다.

$$\begin{aligned} Q_i &= W_Q x_i \\ K_i &= W_K x_i \\ V_i &= W_V x_i \end{aligned} \tag{2.15}$$

W_Q, W_K, W_V는 학습으로 값, 즉 구체적 역할이 정해지는 행렬들이고, Q_i는 query로 어떤 정보를 찾고자 하는지에 대한 것이고, K_i는 key로 어떤 정보를 가지고 있는가, V_i는 value로 정보의 내용이다. 여기서 '정보'란 의미에 관한 정보로, 결론적으로 어떤 토큰이 의미 벡터 공간에서 어떤 식으로 다른 토큰에 변환을 주어야 하는가에 대한 것이다. query의 질문을 key가 답하는 것이라고 보면 된다.

예를 들어 '배'라는 토큰을 입력 내의 다른 토큰 '겨울', '맛' 등과 비교해 '배'의 의미와 가장 관련이 있어서 그 의미를 구체화할 수 있는 토큰을 찾아낸다. 이때 의미가 관련된 정도는 attention score로 결정되는데, score(i, j)는 토큰 $p = i$가 $p = j$에게 얼마나 주의(attention)을 기울이는가를 나타낸다. 'attention'이라는 말이 여기서 나온 것이다.

$$\text{score}(i, j) = \frac{Q_i \cdot K_j}{\sqrt{d_k}} \tag{2.16}$$

여기서 $d_k = \dim(Q \cdot K)$로, 값이 폭주하는 것을 막는다. 이제 이 값에 softmax를 적용하여 합을 1로 정규화한다.

$$\alpha_{ij} = \text{softmax}(\text{score}(i, j)) \tag{2.17}$$

이제 각 토큰의 value 벡터인 V_j를 정규화되어 확률분포로 볼 수 있는 가중치 α_{ij}로 가중치합하여 선형결합하면 주변 토큰의 문맥이 반영된 배의 의미가 도출된다.

$$z_i = \sum_j \alpha(ij) V_j \tag{2.18}$$

우리의 문장에서 원래 '배'는 그냥 '배'라는 단어의 일반적 의미를 가지고 있었으나, '맛있단다' 쪽에 어텐션을 둔 후 그 값이 조정되어 최종적으로 과일로서의 배의 의미를 가지게 된다.

이런 과정들은 각 토큰에 대해 수행되는데, 이것을 행렬로 모아서 식을 간결히 할 수 있다.

$$\begin{aligned} Q &= XW_Q \\ K &= XW_K \\ V &= XW_V \end{aligned} \tag{2.19}$$

이제 전체 어텐션 연산이 한 줄로 표현된다.

$$\boxed{\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V} \tag{2.20}$$

이것이 원 논문에서 제시된 어텐션 연산의 식으로, 논문의 핵심 내용이자 트랜스포머를 대변하는 식이라고 할 수 있다.

트랜스포머는 이 과정을 병렬로 한꺼번에 많이 수행한다. 즉 서로 다른 W_Q, W_K, W_V를 가진 여러 개의 "헤드(head)"가 있는데[11], 각 헤드가 서로 다른 "관점"에서 각기 다른 기준으로 입력을 평가한다. 다시, 이 기준은 학습의 결과이기 때문에 우리가 상식적으로 이해하지 못할 수도 있다.

$$\text{head}_h = \text{Attention}(Q_h, K_h, V_h) \tag{2.21}$$

이제 각 헤드의 결과를 결합하면

$$Z = \begin{bmatrix} z_i^{(1)} \\ z_i^{(2)} \\ \vdots \\ z_i^{(H)} \end{bmatrix} \tag{2.22}$$

[11] 실제 구현에서는 각 헤드가 전체 행렬에서 헤드 수만큼 분할된 가중치를 공유한다고 한다.

처럼 되고, 이제 여러 헤드의 결과를 선형결합하면

$$\text{MultiHead}(Q,K,V) = \begin{bmatrix} \text{head}_1 \\ \vdots \\ \text{head}_H \end{bmatrix} W_O \tag{2.23}$$

여기서 W_O는 출력 행렬로, 최종 의미 결합을 위해 학습된 또 다른 행렬이다.

이제 이 어텐션을 거친 의미들을 다중계층 퍼셉트론, 즉 아래와 같은 순전파 신경망에 통과시킨다.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.24}$$

그 다음, 다시 어텐션을 하고 퍼셉트론에 보내는 것을 여러번 반복한다. 이때, 이미 지정된 문맥적 의미를 바탕으로 또 새로운 문맥적 의미가 추가되어 의미가 구체화되고 깊어지게 된다. 말하자면 단순히 문법적 구조를 파악하는 것을 벗어나, 문장의 의도가 무엇인지, 어떤 표현법이 사용되었는지, 정서는 어떤지 등등을 알아낼 수 있다고 기대하면서 여러번 이것을 반복하는 것이다.

II.6. 트랜스포머의 한계

트랜스포머가 작동하는 과정을 이해했다면 트랜스포머 모델의 한계도 체감할 수 있다. 첫번째로, 기존에 존재하던 RNN 등의 방식보다는 연산이 간결한 것은 맞지만 여전히 $O(n^2)$으로 엄청나게 복잡하다. GPT-3의 경우 학습 시 조정되어야할 어텐션 관련 가중치 파라미터는 자그마치 57,982,058,496 개[12][13]이다. 또, 토큰 사이즈를 늘리면 늘릴수록 행렬들의 차원 수는 제곱으로 늘어나기 때문에 연산의 복잡도가 제곱으로 늘어난다. 따라서 학습과 사용 모두에 엄청난 컴퓨팅 자원이 요구된다. 또, AI 챗봇의 상황에서 대화가 길어지고 내용이 많아지면 챗봇이 대화 내용을 까먹게 되거나 왜곡하면서 갈수록 대답의 질이 떨어진다.

매우 얕은 지식으로 조심스럽게 유추해보건대, AI 회사들은 현재 새로운 모델을 발표할 때 혁신이라고 대차게 광고하면서 내부적으로는 이미 성능 향상에 기여한다고 알려진 조건들, 예를 들어 컴퓨팅 자원, 많은 학습량, 효율적인 학습 및 정보처리 알고리즘, 어텐

[12] 출처: https://arxiv.org/pdf/2005.14165

[13] GPT-3의 어텐션을 제외한 전체 파라미터 수는 어텐션의 그것의 3 배 정도로 약 1750억 개이며, 이후 모델들의 사양은 정확히 공개되지 않고 있으나 확실한 것은 파라미터 수가 정말 엄청나게 많다는 것이다.

션을 반복하는 횟수 등만 늘리는 것이 아닌가 한다. 우리가 GPT-3보다 GPT-5가 압도적으로 똑똑하다고 느끼는 것은 주로 학습량이 비교가 안 될 정도로 차이가 나기 때문일 것이다. "추론" 또는 "이성" 모델[14]이라고 하는 것은 실제로 모델이 뭔가 다른 것을 하는 게 아니라, 사용자의 입력을 바탕으로 모델이 자기가 수행해야할 작업 체크리스트같은 것[15]을 먼저 만든 후 그 단계별로 출력을 수행하는 것이다. 즉, 상술이다. 트랜스포머는 분명 혁명적이었지만 분명 근본적인 한계가 있으며 현재는 그 한계를 단순히 컴퓨팅 자원을 늘려나가고 하드웨어를 개선하거나 FlashAttention, MQA, RoPE 같이 어텐션의 효율성을 증가시키는 것 등으로만 해결하고 있다. 그래서 어떤 사람[16]들은 강인공지능이나 범인공지능이 등장하기 위해서는 트랜스포머에서 탈피해야 한다고 주장하기도 한다.

(반 농담) 무엇보다, LLM은 겨울 배가 맛있다는 문장을, 유행을 잘 아는 사람이라면 당연히 뒤에 뭐가 이어져야 하는지 아는 인간과 달리 뒤에 아무 말이나 만들어서 붙인다[17].

- 인간: 얘, 겨울 배가 맛있단다! 배가 달아.
- LLM(예시): 얘, 겨울 배가 맛있단다! 달고 시원하지?

재치가 없다는 소리이다. AI는 아직은 농담도 상황에 따라 제대로 만들어서 하질 못한다.

[14] reasoning, thinking, think deeper 따위의 문구

[15] 이 과정에 chain of thought 이라는 번지르르한 이름이 붙었으며, GPT-4o, Claude 3 Opus 등에서 사용된다고 각 회사가 공식 발표했다.

[16] 일부 연구자(Yann LeCun, Marcus 등)는 '트랜스포머는 패턴 인식에는 탁월하나 추론과 일반화에는 한계가 있다'고 평한다. 다만 OpenAI나 DeepMind 등은 트랜스포머와 확장된 메모리, 멀티모달 입력 만 드응로 충분히 강인공지능에 근접할 수 있다고 주장한다.

[17] 혹시 이게 뭔지 모르는 독자는 인터넷에 '배 먹어 배'나 '겨울배' 따위를 검색해 보도록 한다.