📌 Positional Encoding 정리 (Transformer 구조 이해)

---

1. 핵심 개념

- Transformer는 RNN과 달리 입력 순서를 고려하지 않고 병렬로 처리하기 때문에, **입력 시퀀스의 위치 정보(positional information)**를 모델이 스스로 알 수 없음.
- 이를 해결하기 위해, 각 입력 토큰에 대해 해당 위치를 나타내는 **Positional Encoding** 벡터를 더하여 순서를 반영함.
- Transformer의 성능에 핵심적으로 기여하는 구성 요소 중 하나임.

---

2. 공식 및 방식 요약

🔹 고정형 **Positional Encoding (논문 방식)**

- 주어진 위치 pos와 임베딩 차원 `i`에 대해 다음과 같이 정의:

PE(pos, 2i)   = sin(pos / 10000^(2i / d))
PE(pos, 2i+1) = cos(pos / 10000^(2i / d))

- pos: 위치 (0부터 시작)
- `i`: 차원 인덱스
- d: 전체 임베딩 차원 수 (예: 512)

👉 최종 입력:

z_pos = x_pos + PE_pos

- 학습형 **Positional Encoding (Learned Positional Embedding)**
  - 위치마다 임베딩 벡터를 직접 학습함
  - 성능상 큰 차이는 없으며, 많은 최신 모델(GPT, BERT)은 이 방식을 사용

---

3. 직관적 예시 표

위치 **(pos)**	차원 **0 (sin)**	차원 **1 (cos)**	차원 **2 (sin)**	차원 **3 (cos)**
0	0.0000	1.0000	0.0000	1.0000
1	0.8415	0.5403	0.0090	0.9999
2	0.9093	-0.4161	0.0180	0.9998

---

4. 스스로 생각해볼 질문

질문 **1:** 단순히 위치 정보를 더하는 것만으로 순서가 잘 반영될까**?**

- 위치 벡터를 덧셈만 하는 방식이 충분한가?

  → 아닌거 같다. 왜냐하면, 위치는 상대적인 것이므로 뒤바뀔수도 있기 때문이다.

질문 **2:** 왜 하필 **Sine**과 **Cosine**을 사용하는가**?**

- 연속된 사인과 코사인을 Orthonormal Basis로 표현가능하며, 고유한 위치 표현을 만들 수 있음
- 수학적으로 미분 가능, 상대적 위치 관계가 선형적으로 표현 가능
- 새로운 위치의 벡터도 두 점 사이의 값을 추정**(interpolation)** 가능

질문 **3: Positional Encoding**이 없는 경우**, Transformer**는 어떻게 동작할까**?**

- 순서 정보가 없으면 "I love you"와 "You love I"가 같은 의미로 처리될 수 있음
- 기계번역, 요약, 질의응답 같은 순서 기반 작업에서 성능이 크게 저하