생성형 AI 기반 RAG 개발자 과정

Transformer 이해
Attention 의 모든 것

Session 2 / 33 — Day 1

토큰이 서로를 참조하는 방식 — 수식 없이 직관으로

박수현 · (사)한국정보공학기술사회 · 2026

LLM 이론 · LangChain · ChromaDB CRUD · Agent — 4일 핸즈온

이번 시간을 마치면

🧩 입력이 숫자가 되는 길

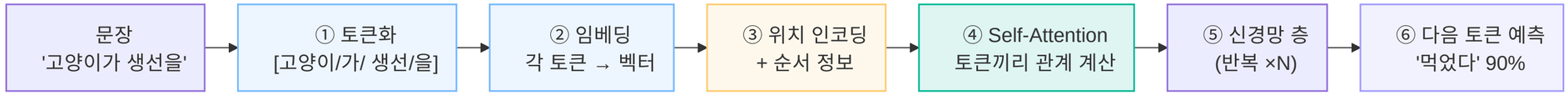
- **토큰화(Tokenization)** — 문장을 조각으로
- **임베딩(Embedding)** — 조각을 벡터로
- **위치 인코딩** — 순서 정보 주입

👁️ Attention 의 직관

- **Self-Attention** — 단어끼리 관계 보기
- **Query · Key · Value** — 도서관 비유
- **멀티헤드 · Transformer 구조** · 다음 토큰 예측

🎯 목표: "GPT 내부에서 무슨 일이 일어나는가"를 **5개 그림**으로 끝낸다.

입력 문장 → 다음 토큰, 한 그림



> 💡 오늘 이 6단계를 차례로 분해합니다. **④ Attention** 이 하이라이트.

PART A

PART A
입력을 숫자로 —
Token · Embedding

> 문장이 벡터가 되기까지 — 약 12분

토큰화 — 문장을 "조각"으로 쪼갠다

토큰이란

- LLM 은 단어가 아니라 **토큰** 단위로 처리
- 자주 쓰는 덩어리는 한 토큰, 드문 단어는 여러 조각 — **서브워드(BPE)**
- 각 토큰에 고유 **정수 ID** → 모델은 ID 나열을 입력으로 받음
- 영어 1 토큰 ≈ 4글자 ≈ 0.75 단어 · 한글은 더 잘게 쪼개짐 (토큰 ↑)

예시

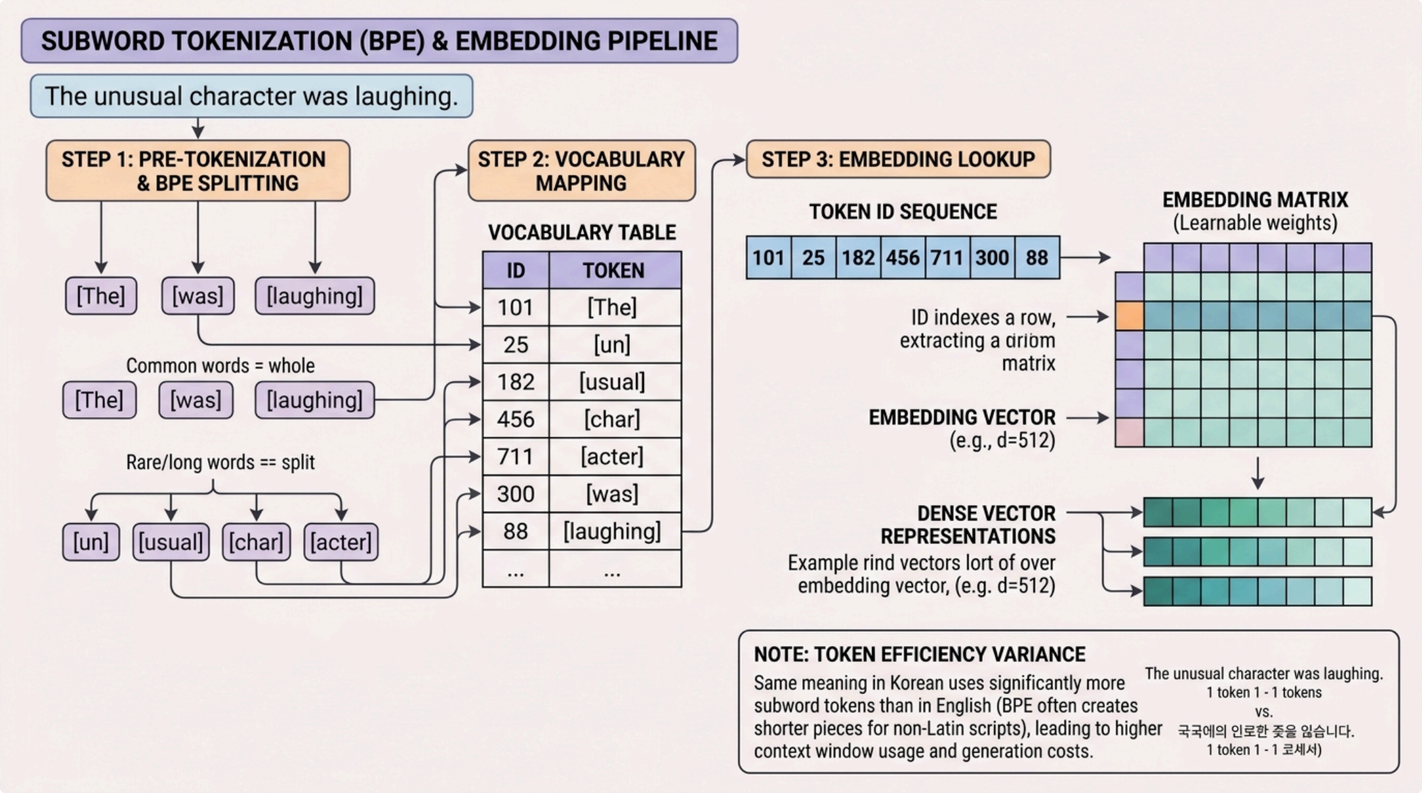
```
"ChatGPT는 똑똑하다"
→ [Chat][G][PT][는][ 똑][똑][하다]
→ 정수 ID [13319, 38, 11571, ...]
```

개발자가 토큰을 알아야 하는 이유

- **비용**이 토큰 수로 과금됨 (글자 수 ✕)
- **context window** = 토큰 한도 (예: 128K)
- 입력+출력 토큰이 한도를 넘으면 잘림
- 같은 의미라도 **한국어는 토큰 2~4배** → 비용·한도에 직결

> 🔑 Session 4 에서 `tiktoken` 으로 **실제 토큰 수·비용**을 직접 계산합니다.

문장 → 토큰 → ID → 임베딩



문장 → 서브워드 토큰 → 정수 ID → 임베딩 벡터 · 토큰이 입력·요금·컨텍스트의 기본 단위

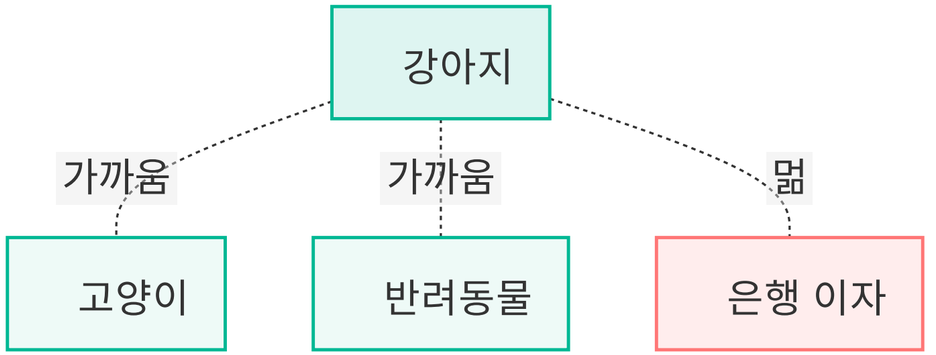
임베딩 — 각 토큰을 "숫자 좌표"로

무엇을 하나

- 정수 ID 는 사전의 일련번호일 뿐 — **의미가 없다**(ID 100 ↮ 101)
- 그래서 토큰 ID → **고차원 벡터**(예: 1536차원)로 변환
- 의미가 비슷하면 **벡터도 가까움**
- "강아지" ↔ "개" 는 가깝고, "강아지" ↔ "은행"은 멂

의미 공간(Semantic Space)

- 단어·문장이 **좌표**로 배치된 공간
- **거리 = 의미 유사도**



> 🔑 이 임베딩이 곧 **RAG 의 검색 엔진**. Session 7 에서 직접 cosine 으로 계산합니다.

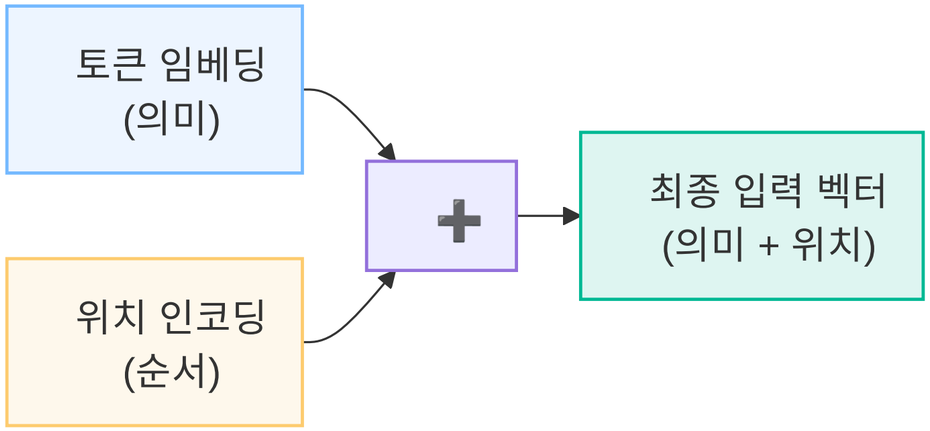
"나는 너를" ≠ "너는 나를"

문제

- Transformer 는 문장을 **한 번에** 봄(병렬)
- → RNN 과 달리 **순서 정보가 없다**
- 하지만 언어는 순서가 의미를 바꿈

해결: Positional Encoding

- 각 토큰 임베딩에 **"몇 번째인지"** 신호를 더함
- 같은 단어라도 위치마다 다른 벡터



> 💡 핵심 한 줄: **의미(임베딩) + 순서(위치) = 모델이 받는 입력**.

PART B

PART B
Attention —
문장이 서로를 참조한다

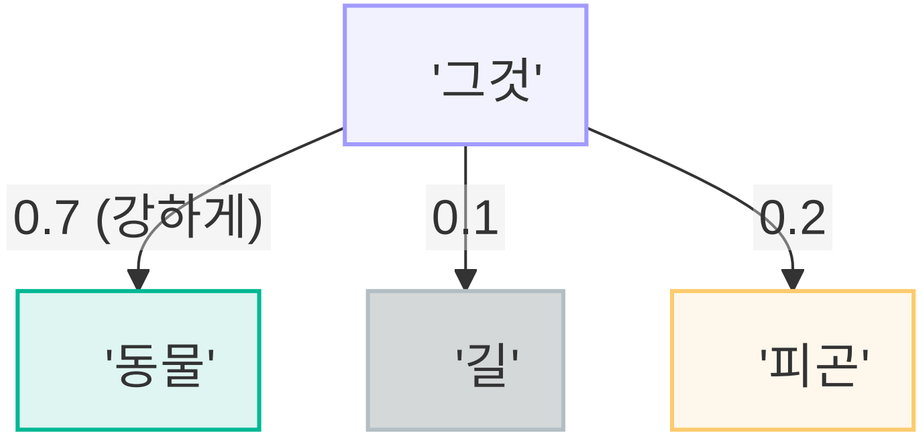
Self-Attention · Q·K·V — 약 14분

"이 단어를 이해하려면, 어디를 봐야 하나?"

직관

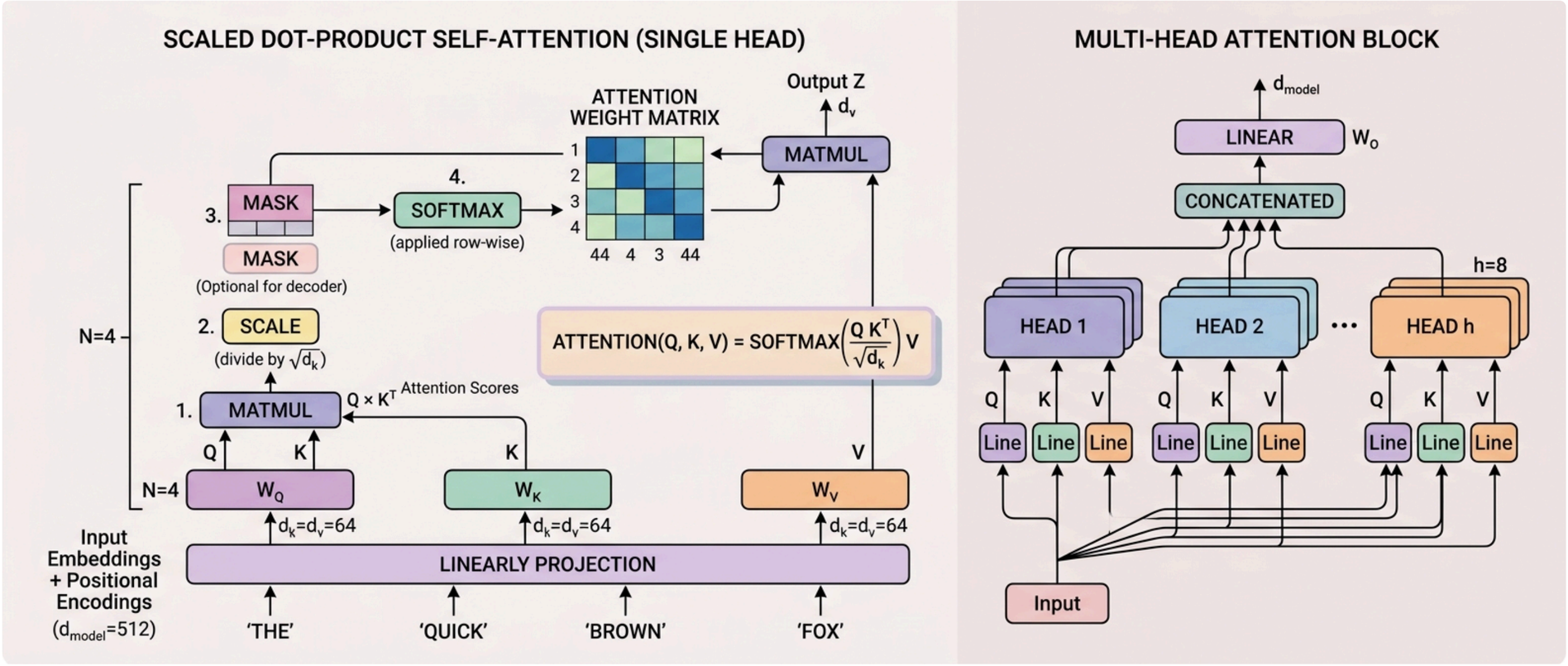
> "그 **동물**은 길을 건넜다. 왜냐하면 **그것**이 피곤했기 때문이다."

- "**그것**"을 이해하려면? → "**동물**"을 봐야 함
- Attention = 각 단어가 **다른 단어에 얼마나 주목할지** 가중치 계산



> 🔑 모든 단어가 모든 단어를 **동시에** 참조한다 = **Self-Attention**.

$Attention(Q,K,V) = softmax(QK^T/\sqrt{d_k})V$

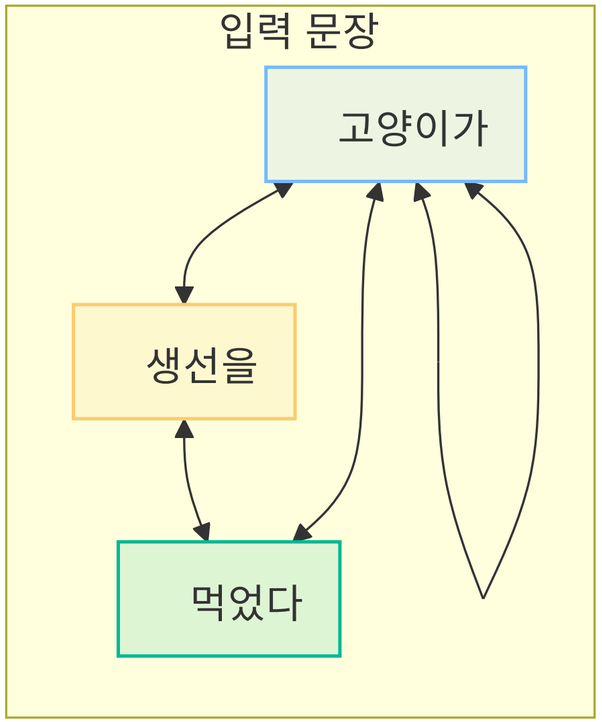


셀프 어텐션 $Attention(Q,K,V)=softmax(QK^T/\sqrt{d_k})V$ · Q·K 유사도로 V를 가중 합성(멀티헤드 병렬)

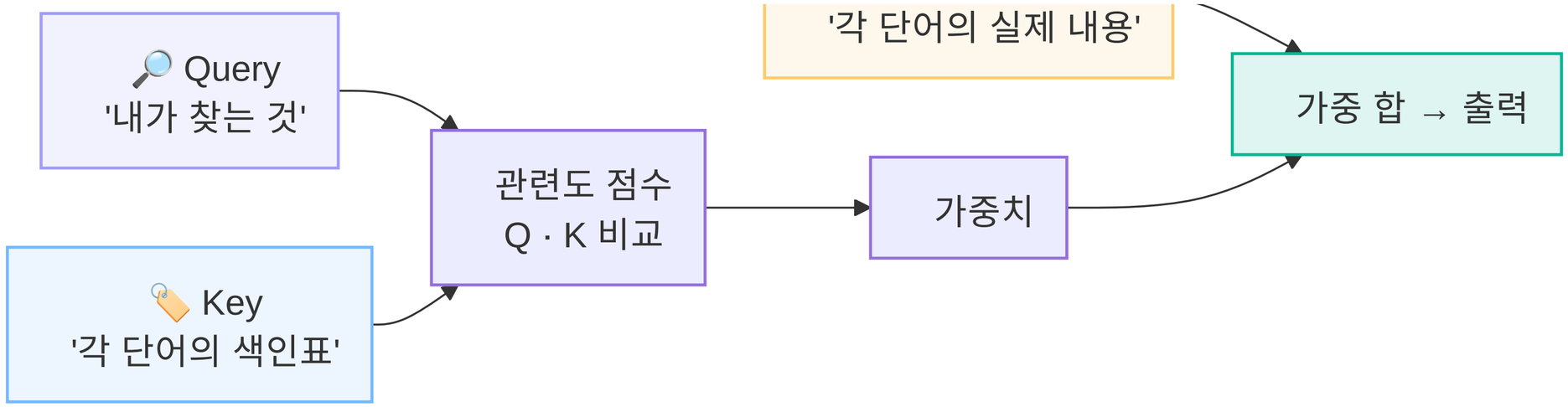
Self-Attention: 모두가 모두를 본다

- 각 토큰이 **자기 포함 모든 토큰**과의 관련도를 계산
- 관련도가 높은 토큰의 정보를 **더 많이 흡수**
- → "먹었다"는 "고양이가"(주어)와 "생선을"(목적어)를 함께 보고 의미 확정

> 💡 RNN(순차) 대비 혁명: 거리가 멀어도 **한 번에** 연결 → 장기 의존성 해결.



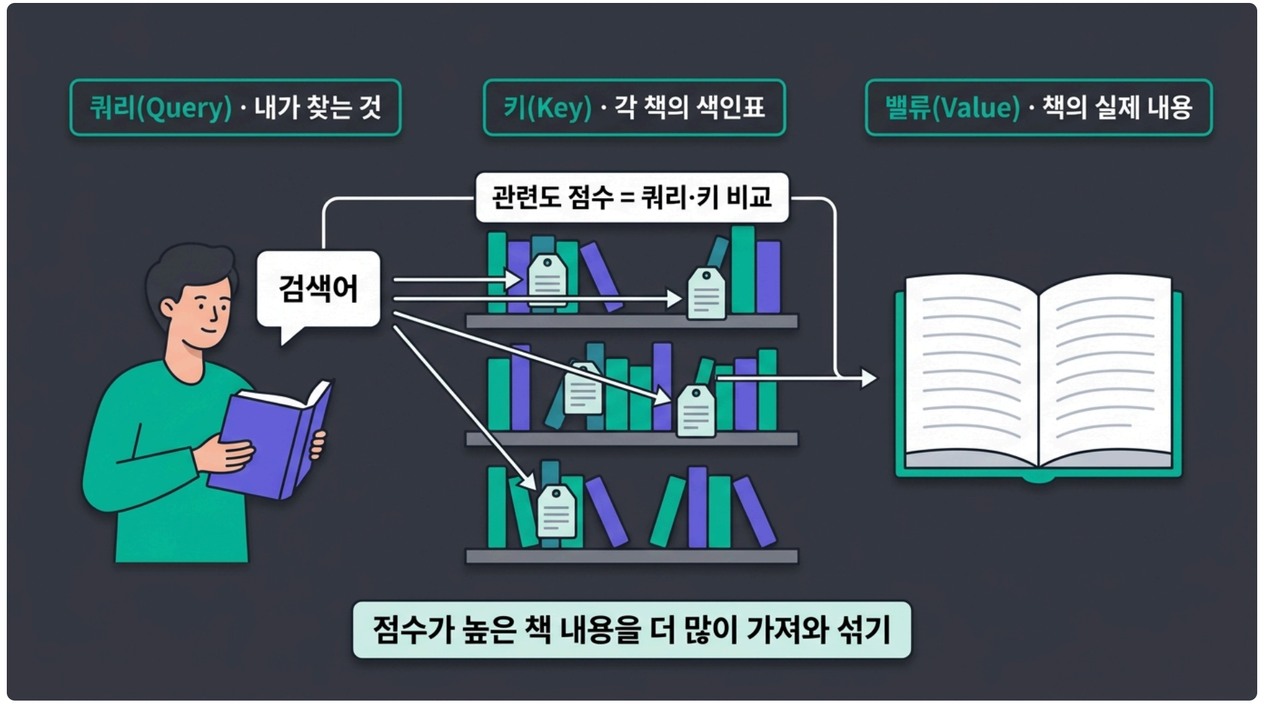
Q·K·V — 검색의 3요소



요소	도서관 비유	역할
Query	내가 찾는 검색어	"지금 이 단어가 원하는 정보"
Key	책마다 붙은 색인 태그	"각 단어가 가진 라벨"
Value	책의 실제 내용	"가져올 실제 정보"

🔑 한 줄 요약: **Query 와 Key 를 비교해 점수를 내고, 그 점수로 Value 를 섞는다.**

Q·K·V — 도서관 비유 한 장으로



💡 **Query**(찾는 것) 와 **Key**(색인표) 를 비교해 관련도를 매기고, 그 점수로 **Value**(실제 내용) 를 가중합해 출력합니다.

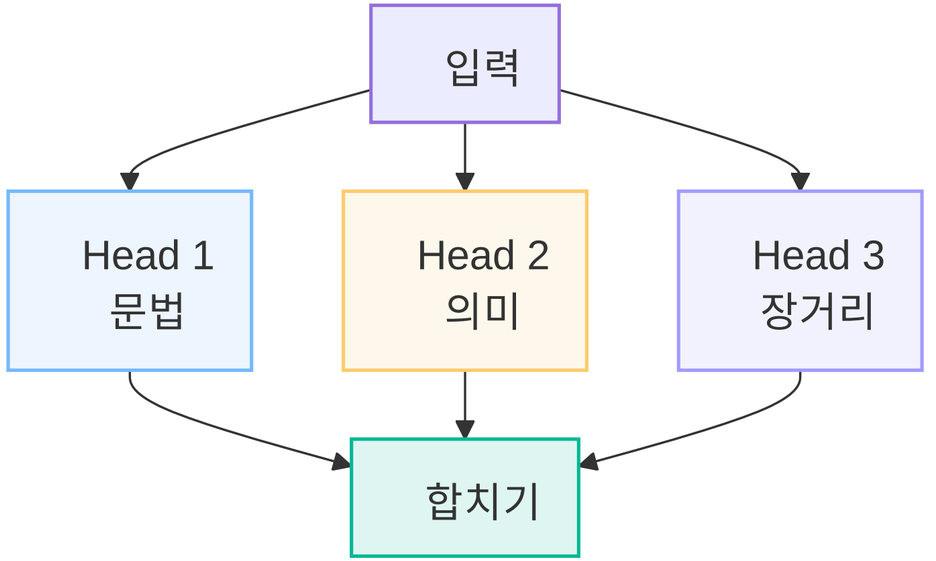
Multi-Head Attention — 한 번이 아니라 여러 번

왜 여러 "헤드"인가

- 한 번의 Attention = **한 가지 관점**만 포착
- 그런데 문장은 여러 관계를 **동시에** 가짐
- **문법** 관계 (주어-동사 일치)
- **의미** 관계 (대명사가 가리키는 대상)
- **장거리** 관계 (멀리 떨어진 단어 연결)

멀티헤드는 이렇게 동작

- Q·K·V 를 **헤드 수만큼 여러 벌**(예: 8·12벌) 만든다
- 각 헤드가 **독립적으로 병렬** Attention 수행
- 헤드마다 **서로 다른 관계**에 집중 (한 헤드는 문법, 다른 헤드는 지시 관계…)
- 각 헤드 결과를 **이어 붙여(concat)** 하나의 출력으로 합침 → 표현력 ↑



💡 여러 전문가가 같은 문장을 각자 관점으로 읽고 종합하는 셈.

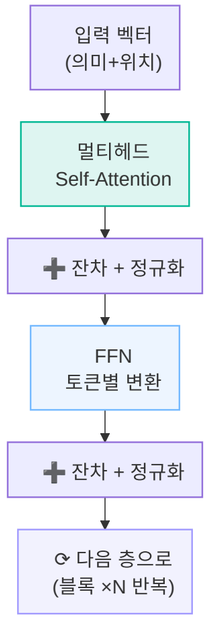
Attention 한 번이 아니라 — "층(layer)"으로 쌓는다

한 층(블록) 안에는

1. **멀티헤드 Self-Attention** — 단어끼리 관계 흡수
2. **FFN(Feed-Forward)** — 각 토큰을 개별 변환·정제 - 두 단계 모두 **잔차 연결 + 정규화**로 감쌈 - **잔차(residual)**: 입력을 출력에 더해 줌 → 깊어도 학습 안정 - **정규화(LayerNorm)**: 값의 크기를 고르게 → 수렴 빠름

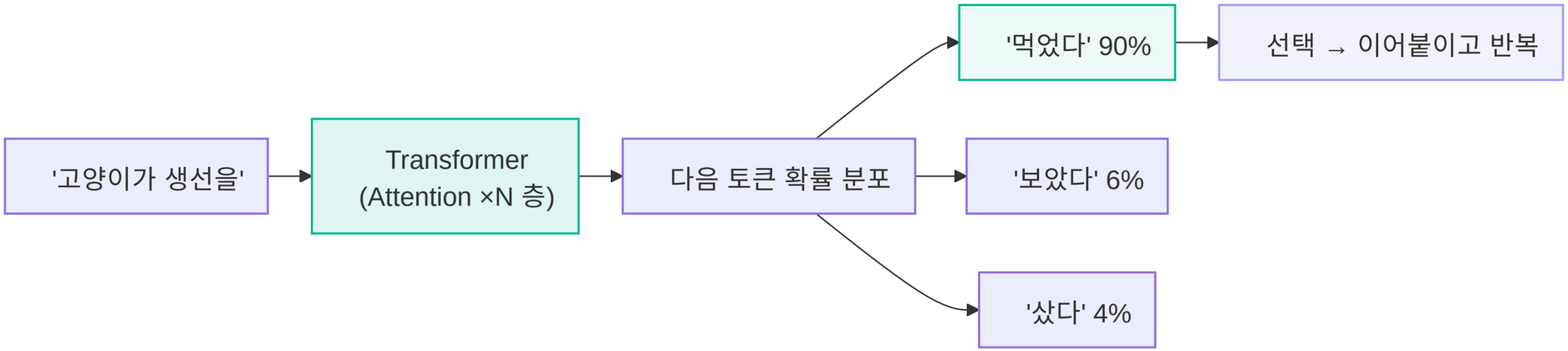
이 블록을 N번 반복

- 층이 깊어질수록 단순 패턴 → 복잡한 의미·문맥
- GPT 류는 수십~수백 층 (예: 수십 개)



> 💡 **인코더 / 디코더**: 이해(읽기)에 강한 인코더(BERT 계열)와 생성(쓰기)에 강한 디코더(GPT 계열)가 같은 블록을 다르게 조합한 것. **GPT 는 디코더만** 쌓아 "다음 토큰"을 생성.

GPT 는 결국 "다음 토큰 맞히기"



- 생성 = **다음 토큰 예측 → 이어붙이기 → 반복** (자기회귀, autoregressive)
- 확률 분포에서 어떻게 고르냐 = **temperature** 가 조절 (Session 5)
- 높은 temperature → 덜 확률적인 토큰도 선택 → 창의적/불안정

> 🔑 "환각(hallucination)"의 뿌리: 모델은 **사실이 아니라 '그럴듯한 다음 토큰'**을 고른다. → 그래서 **RAG** 가 필요 (Session 6).

이론이 실습으로 — 4가지 연결고리

Transformer 개념	코스에서 이렇게 만난다
토큰	Session 4 — `tiktoken` 으로 비용·context 한도 계산
임베딩 / 의미공간	Session 7 — cosine 으로 직접 검색, Day2 벡터 DB
다음 토큰 확률	Session 5 — `temperature` · `top_p` 파라미터
환각 (그럴듯함≠사실)	Session 6~ — RAG 로 근거 주입

핵심 7가지

1. ✅ 입력 처리: **토큰화 → 임베딩 → 위치 인코딩**
2. ✅ **임베딩 = 의미 좌표**, 가까울수록 비슷한 의미 → RAG 의 바탕
3. ✅ **Self-Attention** — 모든 단어가 모든 단어를 동시에 주목
4. ✅ **Q·K·V** — Query·Key 로 점수, Value 를 가중합 (도서관 비유)
5. ✅ **멀티헤드** — Q·K·V 를 여러 벌로, 여러 관계를 병렬 포착 후 합침
6. ✅ **Transformer 구조** — (멀티헤드 Attention + FFN), 잔차·정규화로 감싸 **N층 반복**, GPT 는 디코더 스택
7. ✅ 생성 = **다음 토큰 예측 반복**, '그럴듯함 ≠ 사실' → 환각·RAG 의 이유