생성형 AI 기초 마스터

LLM 핵심 용어와 개념

01

토큰(Token)과 토큰화

1 토큰과 토큰화

LLM이 자연어를 바로 이해하지 못하는 이유

인간은 자연어를 의미 기반으로 이해하지만, LLM은 계산 가능한 형태로 변환하는 전처리 단계가 필요

LLM이 자연어를 이해하기 위해 필요한 과정: 토큰화

- 토큰으로 나누는 이유? LLM이 자연어를 이해할 수 있게 하기 위해서!
- LLM이 자연어를 이해하고 구분하기 위해서는 말뭉치(Corpus)를 일정한 단위인 토큰(Token)으로 나뉘어야 함

단계	핵심 개념
말뭉치 (Corpus)	모델을 학습시키기 위해 수집된 대규모 텍스트 데이터(뉴스, 대화, 리뷰, 문서 등)
토큰(Token)	텍스트를 모델이 계산 가능한 최소 단위로 나눈 형태(단어 · 부분단어 · 문장부호 등)
토큰화(Tokenization)	자연어를 토큰 단위로 분리해, 모델이 처리 가능한 형태로 변환하는 과정

2 토큰의 정의 및 역할

토큰(Token)

토큰

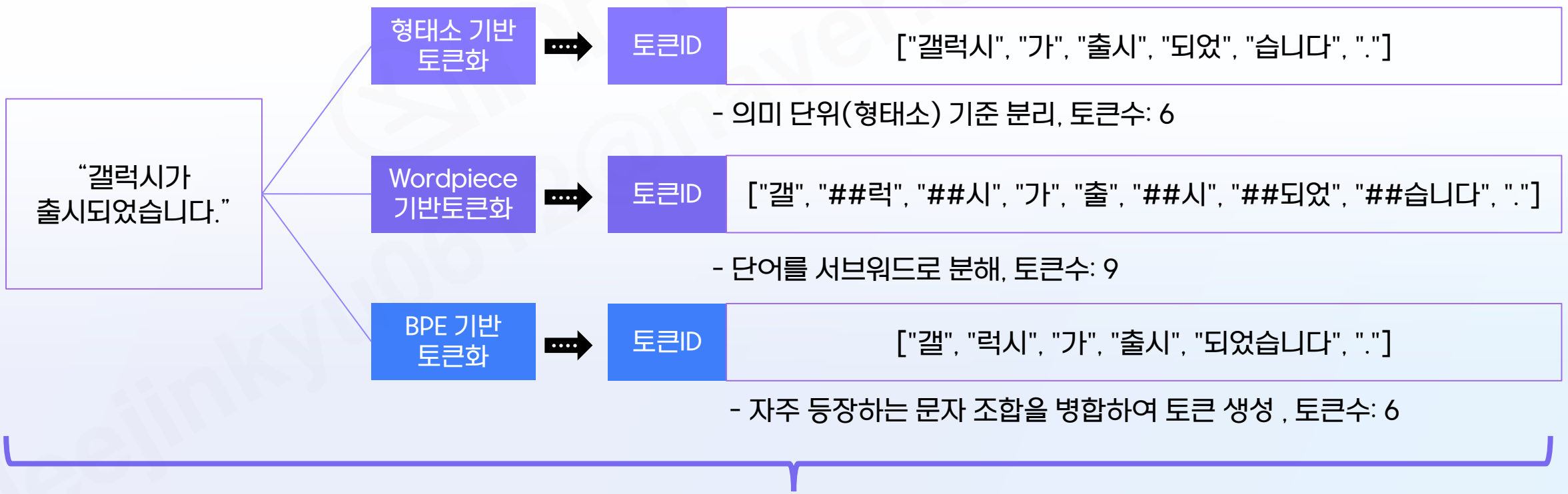
- LLM이 문장을 이해하고 생성할 때 사용하는 가장 작은 단위
- 일반적으로 단어, 문장부호, 또는 단어의 일부분이 하나의 토큰이 됨
- 토큰화 방식에 따라 다양하게 나뉨
  - ➡ 형태소와 같은 의미기반, 패턴 기반 등

방식	기술적 성격	방식
형태소 기반 토큰화	언어 규칙 기반	문장을 단어의 의미 단위(형태소) 로 나눔 예) 명사, 동사 어간, 조사, 어미
WordPiece	통계 기반 서브워드 방식	단어를 서브워드(작은 단어 조각)로 분해한 뒤, 학습 데이터의 사용 패턴을 기반으로 가장 적합한 조합을 선택해 토큰을 구성하는 방식
BPE (Byte Pair Encoding)	빈도 기반 서브워드 방식	단어를 서브워드로 나눈 뒤, 자주 함께 등장하는 문자 쌍을 반복적으로 병합해 토큰을 생성하는 방식 의미 기준이 아닌 사용 빈도(패턴)에 따라 토큰을 합치는 것이 특징

3 토큰화의 실제 예시

문장이 모델 안에서 쪼개지는 방식 이해하기

- 토큰화: 문장을 모델이 이해할 수 있는 작은 조각으로 분리하는 과정
- 토큰ID: 토큰(단어/서브워드/기호 등)을 모델이 계산할 수 있도록 숫자로 변환한 값



같은 문장도 토크나이저에 따라 토큰 개수가 달라짐

01

프롬프트 엔지니어링

1 LLM과 프롬프트의 기본 개념

LLM을 똑똑하게 쓰는 방법: 프롬프트란 무엇인가?

프롬프트 엔지니어링의 개념과 중요성

프롬프트	프롬프트 엔지니어링
사용자가 LLM에게 '무엇을, 어떻게' 물어볼지 정의하는 입력문	모델이 최적의 결과를 내도록 프롬프트를 구조화·설계하는 기술

- LLM의 응답 품질은 프롬프트의 구성과 표현 방식에 따라 크게 달라짐
- 단순한 질문보다 맥락, 역할, 의도 등을 명확히 제시할수록 더 정확한 결과를 얻을 수 있음

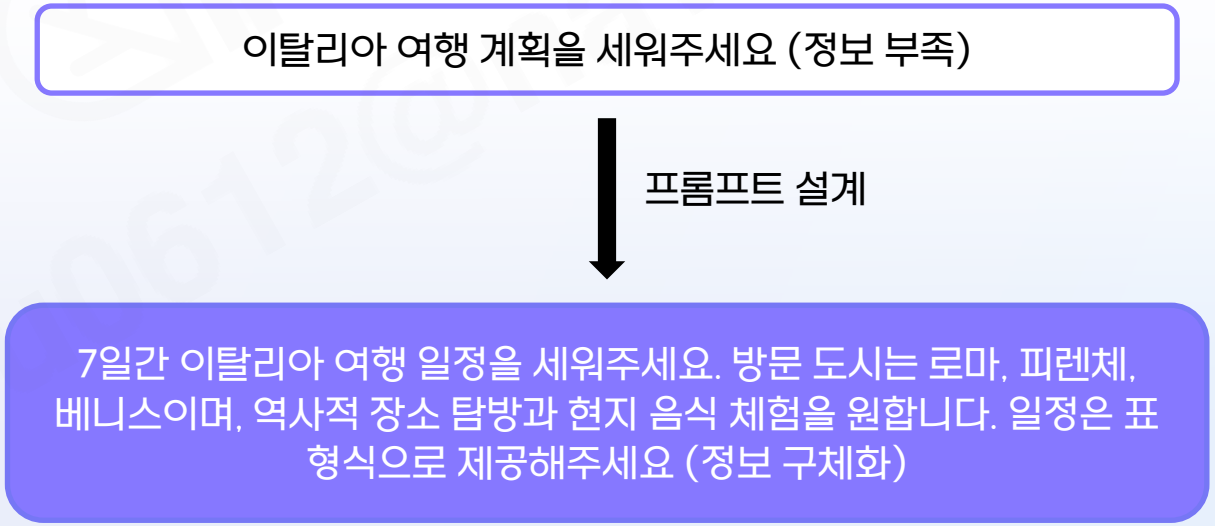
leejinkyu@naver.com 846896

2 프롬프트 엔지니어링의 이해

프롬프트 엔지니어링의 역할과 가치

프롬프트가 LLM의 성능을 좌우한다

- 동일한 질문이라도 프롬프트 구조에 따라 결과의 정확도와 깊이가 달라짐
- 명확한 역할 지정과 단계적 요청은 모델의 논리적 사고를 유도함
- 프롬프트는 단순한 지시문이 아니라, 모델의 '사고 과정'을 설계하는 도구임



02

효과적인 프롬프트 설계

1 프롬프트의 구성

프롬프트의 기본 구성요소

- 좋은 프롬프트는 구조에서 시작된다
- 역할, 맥락, 지시, 출력 형식을 포함해 구조화된 프롬프트를 만들면 효과적인 결과를 얻을 수 있음

역할 (Role)	모델에게 어떤 인물이나 전문가로서 행동할지를 지정함
맥락 (Context)	문제의 배경, 조건, 목적 등을 구체적으로 제시함
지시 (Instruction)	모델이 수행해야 할 작업이나 질문을 명확히 전달함
출력 형식 (Output format)	응답의 형식, 구조, 길이 등을 구체적으로 요청함

2 프롬프트 작성 원칙

프롬프트 작성의 기본 원칙

- 명확하고 구체적인 프롬프트일수록 더 정확한 결과를 얻을 수 있음

명확성	불필요한 모호함을 줄이고, 질문의 목적을 분명히 제시함
구체성	세부 조건(형식, 길이, 단계 등)을 구체적으로 명시함
맥락성	모델이 이해할 수 있도록 배경 정보와 의도를 함께 제공함
단계성	복잡한 문제는 단계별로 나누어 처리하도록 안내함

03

RAG 시스템 구현 단계별 워크플로우

1 RAG 구현: 데이터 로드 및 전처리

RAG 구현 과정

01
데이터 로드 및 전처리

02
임베딩 및 벡터스토어 생성

03
프롬프트 및 리트리버 생성

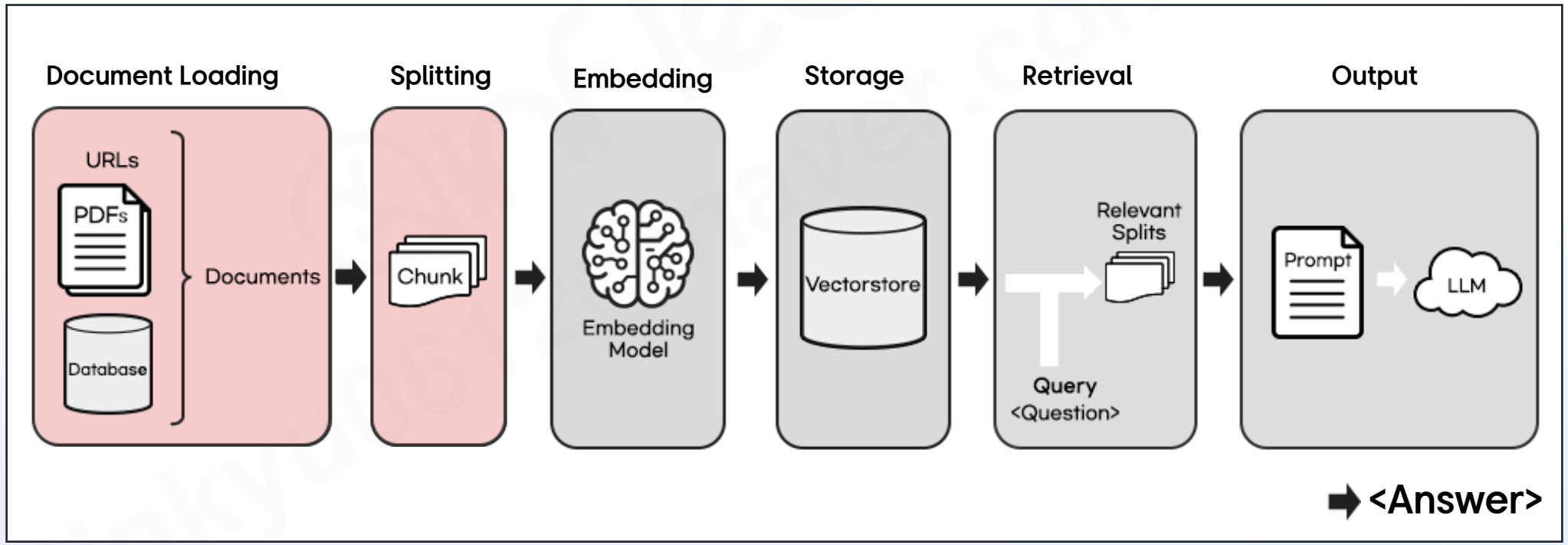
04
언어 모델 생성 및 체인 실행

leejinkyu0612@gmail.com 846896

1 RAG 구현: 데이터 로드 및 전처리

RAG 구현 과정

1. 데이터 로드 및 전처리



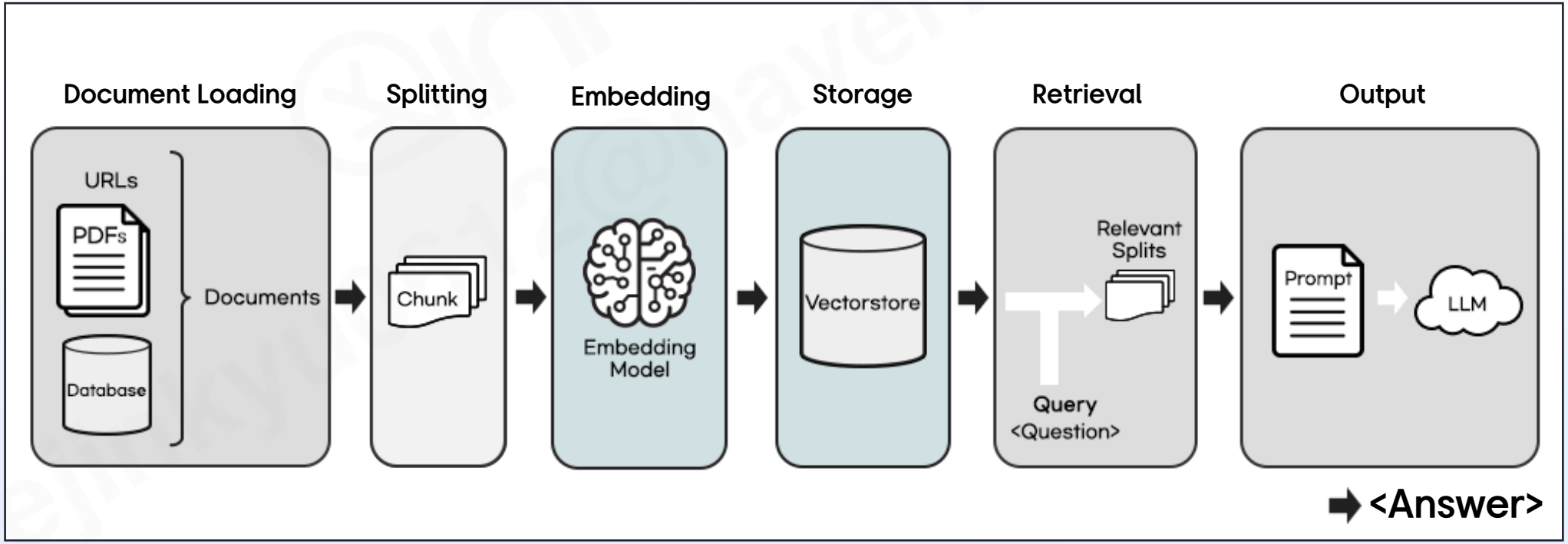
- 문서 업로드: PDF, Word 등 다양한 형식의 문서를 시스템에 불러오는 단계
- Parsing(파싱): 문서에서 텍스트를 추출하고, 불필요한 내용,형식을 정리해 모델이 사용할 수 있는 형태로 만드는 과정
- Chunking(청킹): Parsing(파싱)된 텍스트를 의미가 유지되는 범위에서 나누는 과정이며,
  청크 간 일부 내용을 공유하는 오버랩(Overlap)을 적용해 검색과 임베딩에 활용할 수 있도록 만드는 과정

2 RAG 구현: 임베딩 및 벡터스토어

RAG 구현 과정

2. 임베딩 및 벡터스토어 생성

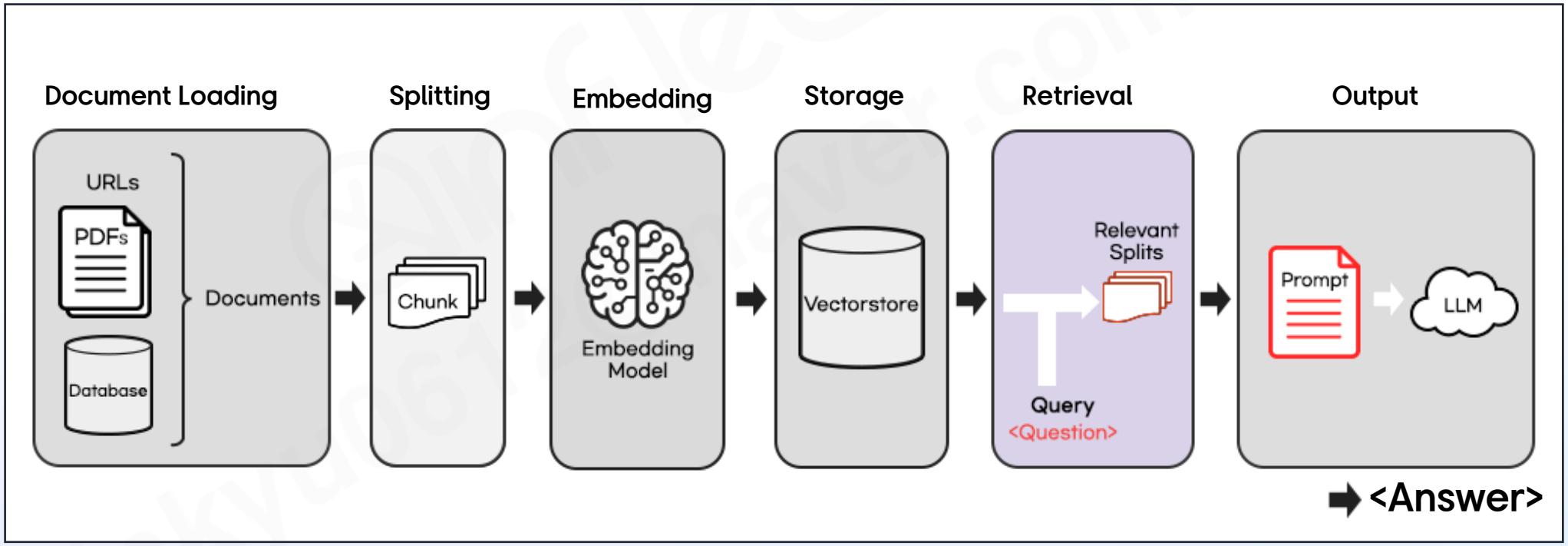
- 임베딩: Chunking과 Overlap으로 나뉜 텍스트 조각들을 임베딩 모델을 통해 벡터로 변환
- 벡터 데이터베이스 생성: 변환된 벡터를 벡터 데이터베이스에 저장



3 RAG 구현: 프롬프트 및 리트리버

RAG 구현 과정

3. 프롬프트 및 리트리버 생성



- 리트리버(Retriever): 사용자의 질문을 벡터로 변환하고 벡터 DB와 비교해 의미적으로 유사한 문서를 찾은 뒤, 상위 K개의 관련 문서를 반환하는 모듈
- 재순위화(Rerank): 반환된 문서 중 질문과 더 높은 관련성이 있는 문서를 우선순위로 다시 정렬하는 단계
- 프롬프트 생성: 사용자 질문과 검색된 관련 문서를 합쳐 하나의 입력으로 만들어 LLM에 전달

3 RAG 구현: 프롬프트 및 리트리버

RAG에서의 Top-K — 필요한 정보만 똑똑하게 고르기

- RAG는 질문에 맞는 문서를 찾아서 답을 만드는 방식
- Top-K는 리트리버(Retriever)에서 검색된 문서 중 가장 관련성이 높은 상위 K개만 선택

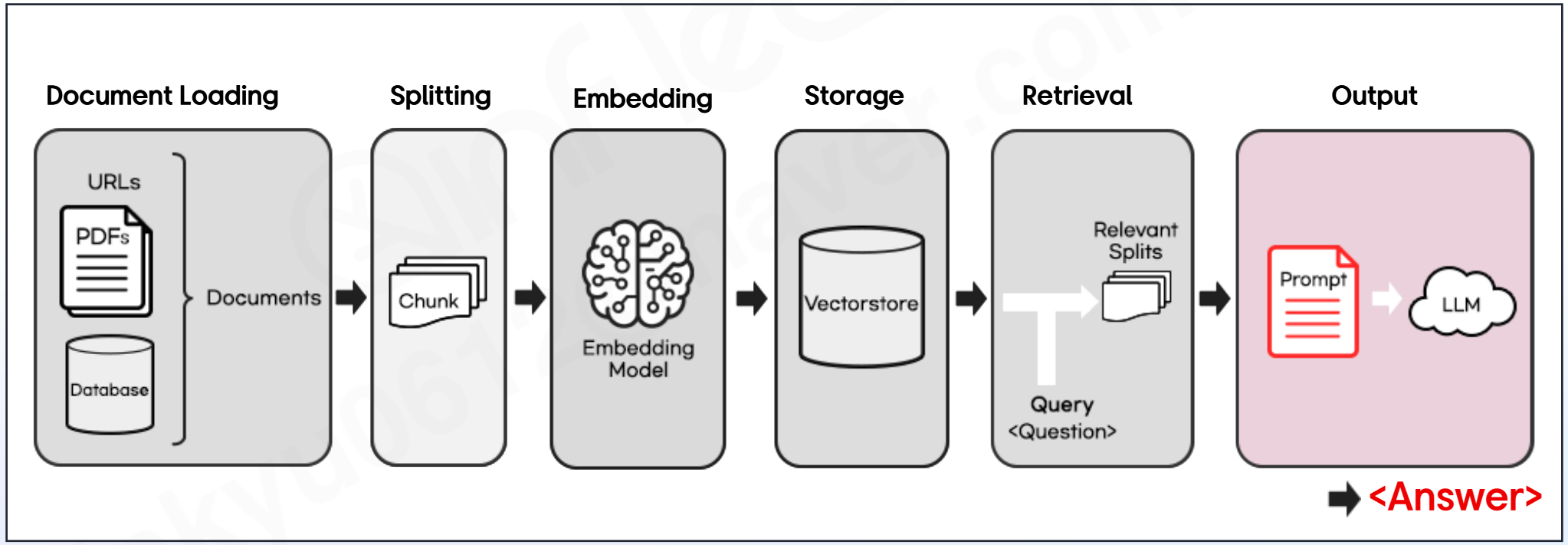
Top-K가 작은 경우	Top-K가 큰 경우
정확하지만 정보가 제한됨	불필요한 내용이 섞일 수 있음

leejinkyu6120@naver.com 848898

4 RAG 구현: 언어 모델 생성 및 체인 실행

RAG 구현 과정

4. 언어 모델 생성 및 체인 실행



- 프롬프트를 전달받은 언어 모델은 문맥을 이해한 뒤, 이를 기반으로 최종 답변을 생성하는 단계

▶참고: 임베딩(Embedding) 모델은 텍스트를 벡터로 변환해 의미 기반으로 문서를 찾아주는 역할이고, LLM은 검색된 내용을 바탕으로 최종 답변을 생성하는 역할로 구분됨