

# 테디노트의 랭체인을 활용한 RAG 비법노트 (기본편)

## 챕터8 메모리

## 메모리란?

- GPT 모델의 대화 기억 한계 해결
- 이전 대화 내용 참조 기능
- 멀티턴 대화 지원

## 학습 목표

1. 메모리 vs 챗 히스토리 차이점
2. 7가지 메모리 유형 이해
3. 상황별 최적 메모리 선택
4. 실무 구현 방법

# 대화 유형

## 싱글턴 대화

- 1회성 질문과 답변
- 이전 맥락 참조 없음

## 멀티턴 대화

- 연속된 질문과 답변
- 이전 대화 맥락 유지 필요

# 메모리 유형 개요

## 7가지 주요 메모리 타입



1. Buffer Memory
2. Buffer Window Memory
3. Token Buffer Memory
4. Entity Memory
5. Knowledge Graph Memory
6. Summary Memory
7. Vector Store Memory

# 1 ConversationBufferMemory

## 특징

- 모든 대화 내용 저장
- 가장 기본적인 형태
- 순차적 대화 기록

## 장단점

-  완전한 대화 보존
-  토큰 한계 위험

## 2 ConversationBufferWindowMemory

### 특징

- 최근 k개 대화만 유지
- 오래된 대화 자동 삭제
- FIFO 방식

### 핵심 파라미터

- `k`: 윈도우 크기
- 메모리 사용량 일정 유지

### 3 ConversationTokenBufferMemory

#### 특징

- 토큰 단위 대화 제한
- 비용 효율적 관리
- `max_token_limit` 설정

#### 장점

- 정확한 비용 제어
- 모델별 토큰 최적화



## 4 ConversationEntityMemory

### 특징

- 엔티티 기반 정보 추출
- 핵심 정보만 저장
- LLM 자동 분류

### 엔티티 예시

- 사람: "테디는 개발자"
- 관계: "테디와 셸리는 동료"

## 5 ConversationKGMemory

### 특징

- 지식 그래프 저장
- 객체 간 관계 파악
- 복잡한 연결망 관리

### 출력 형태

김설리 씨 is a 신입 디자이너  
김설리 씨 is in 우리 회사



## 6 ConversationSummaryMemory

### 두 가지 방식

#### 1. SummaryMemory

- 실시간 요약 저장

#### 2. SummaryBufferMemory

- 최근: 원본 유지
- 이전: 요약본 저장

## 7 VectorStoreRetrieverMemory

### 특징

- 벡터 기반 유사도 검색
- 시간순 무관 정보 검색
- FAISS 벡터 스토어 활용

### 장점

- 관련성 높은 정보 추출
- 긴 대화에서도 효율적

## 구현 방법

### LCEL 체인 연동

- RunnablePassthrough 활용
- MessagesPlaceholder 설정
- 자동화된 대화 흐름

### 데이터베이스 연동

- SQLite 영구 저장
- 세션 기반 관리
- RunnableWithMessageHistory

## 메모리 선택 가이드

상황	추천 메모리
짧은 대화	BufferMemory
긴 대화	BufferWindowMemory
비용 고려	TokenBufferMemory
핵심 정보	EntityMemory
관계 파악	KGMemory
매우 긴 대화	SummaryMemory
검색 기반	VectorStoreMemory

# 성능 최적화

## 1. 토큰 모니터링

- 사용량 추적
- 모델별 효율성

## 2. 파라미터 조정

- k값 최적화
- 서비스 특성 반영

## 3. 하이브리드 방식

- 여러 메모리 조합
- 상황별 전환

## 핵심 포인트

1. 메모리는 멀티턴 대화의 핵심
2. 상황별 적절한 선택 중요
3. 효율적인 구현으로 통합
4. 지속적인 최적화 필요



## 다음 단계

- 실제 프로젝트 적용
- 사용자 피드백 수집
- 고급 메모리 기법 탐구
- 커스터마이징 구현

# Thank You!

질문이 있으시면 언제든지 말씀해 주세요!