
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

ConversationTokenBufferMemory

- 저장할 대화 내용을 토큰(단어 조각)의 개수를 기준으로 제한하는 메모리
- 최근 대화의 히스토리를 버퍼를 메모리에 보관 → 대화의 개수가 아닌 **토큰 길이**를 사용하여 대화내용을 플러시(`flush`)할 시기를 결정
- **역할**:
  - 메모리 사용량을 효율적으로 관리하여, 지정된 토큰 수만큼만 대화를 기억
  - **📝 글자 수 제한이 있는 짧은 메모지**

```
# 환경변수 처리 및 클라이언트 생성
from langsmith import Client
from langchain.prompts import PromptTemplate
from langchain.prompts import ChatPromptTemplate

from dotenv import load_dotenv

import os
import json

# 클라이언트 생성
api_key = os.getenv("LANGSMITH_API_KEY")
client = Client(api_key=api_key)
```

"@traceable" 주석은 허용되지 않습니다. 허용되는 값은 다음과 같습니다. [@param, @title, @markdown]

```
# LangSmith 추적 설정하기 (https:smith.langchin.com)
# LangSmith 추적을 위한 라이브러리 임포트
from langsmith import traceable

# LangSmith 환경 변수 확인

print("\n--- LangSmith 환경 변수 확인 ---")
langchain_tracing_v2 = os.getenv('LANGCHAIN_TRACING_'
langchain_project = os.getenv('LANGCHAIN_PROJECT')
langchain_api_key_status = "설정됨" if os.getenv('LANG
org = "설정됨" if os.getenv('LANGCHAIN_ORGANIZATION')

if langchain_tracing_v2 == "true" and os.getenv('LAN
    print(f"✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACIN
    print(f"✅ LangSmith 프로젝트: '{langchain_project
    print(f"✅ LangSmith API Key: {langchain_api_key
    print("  -> 이제 LangSmith 대시보드에서 이 프로젝트를 확인
else:
    print("❌ LangSmith 추적이 완전히 활성화되지 않았습니다.
    if langchain_tracing_v2 != "true":
        print(f"  - LANGCHAIN_TRACING_V2가 'true'로 설
    if not os.getenv('LANGCHAIN_API_KEY'):
        print("  - LANGCHAIN_API_KEY가 설정되어 있지 않습
    if not langchain_project:
        print("  - LANGCHAIN_PROJECT가 설정되어 있지 않습
```

- 셀 출력

```
--- LangSmith 환경 변수 확인 ---
✅ LangSmith 추적 활성화됨 (LANGCHAIN_TRACING_V2='true')
✅ LangSmith 프로젝트: 'LangChain-prantice'
✅ LangSmith API Key: 설정됨
-> 이제 LangSmith 대시보드에서 이 프로젝트를 확인해 보세요.
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

# LLM 생성
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0.7,
    max_output_tokens=4096,
    )
```

- **max_token_limit : 대화 내용을 저장할 최대 토큰의 길이 설정**

```
from langchain.memory import ConversationTokenBufferMemory

# 메모리 설정
memory = ConversationTokenBufferMemory(
    llm=gemini_lc,
    max_token_limit=150,                        # 최대 토큰 길이를 150개로 제한
    return_messages=True
)
```

- 셀 출력 = 오류 메시 출력됨

```
/var/folders/h3/l7wnkv352kqftv0t8ctl2ld40000gn/T/ipykernel_7470/3077724278.py:4: LangChainDeprecationWarning:
  memory = ConversationTokenBufferMemory(
```

- 의미: 랭체인에서 더 이상 사용하지 않을 예정인 기능에 대한 경고
    - 첫번째 줄의 의미

```
/var/folders/h3/l7wnkv352kqftv0t8ctl2ld40000gn/T/ipykernel_7470/3077724278.py:4: LangChainDeprecationWarni
```

- 메모리 기능이 새로운 방식으로 변경되었으니, 위 주소의 마이그레이션 가이드를 참고하여 코드를 수정하라는 의미

- 두번째 줄의 의미

```
memory = ConversationTokenBufferMemory
```

- 현재 사용하고 있는 ConversationTokenBufferMemory() 코드가 곧 지원 중단될 예정이라는 의미

- 임의의 대화 추가하기

```
memory.save_context(
    inputs={
        "human": "안녕하세요, 저는 최근에 여러분 회사의 공작 기계를 구매했습니다. 설치 방법을 알려주실 수 있나요?"
    },
    outputs={
        "ai": "안녕하세요! 구매해 주셔서 감사합니다. 해당 기계 모델 번호를 알려주시겠어요?"
```

```
    },
)
memory.save_context(
    inputs={"human": "네, 모델 번호는 XG-200입니다."},
    outputs={
        "ai": "감사합니다. XG-200 모델의 설치 안내를 도와드리겠습니다. 먼저, 설치할 장소의 전원 공급 상태를 확인해주세요. 기계는 220V 전원
    },
)
memory.save_context(
    inputs={"human": "전원은 확인했습니다. 다음 단계는 무엇인가요?"},
    outputs={
        "ai": "좋습니다. 다음으로, 기계를 평평하고 안정된 바닥에 배치해 주세요. 이후, 제공된 사용자 매뉴얼에 따라 케이블 연결을 진행해 주시기
    },
)
memory.save_context(
    inputs={"human": "연결은 어떻게 하나요?"},
    outputs={
        "ai": "매뉴얼의 5페이지를 참조해 주세요. 케이블 연결에 관한 상세한 지침이 있습니다. 이 과정에서 어려움이 있으시면 추가적으로 도와드리겠
    },
)
memory.save_context(
    inputs={"human": "설치가 완료되면 어떻게 해야 하나요?"},
    outputs={
        "ai": "설치가 완료되면, 전원을 켜고 초기 구동 테스트를 진행해 주시기 바랍니다. 테스트 절차는 매뉴얼의 10페이지에 설명되어 있습니다. 만
    },
)
memory.save_context(
    inputs={"human": "감사합니다, 도움이 많이 되었어요!"},
    outputs={
        "ai": "언제든지 도와드릴 준비가 되어 있습니다. 추가적인 질문이나 지원이 필요하시면 언제든지 문의해 주세요. 좋은 하루 되세요!"
    },
)
```

- 셀 처리에 26.6s 소요

- 최대 토큰의 길이를 150 으로 설정하고 대화를 저장했을 때 어떻게 동작하는지 확인해보기

```
# 대화내용 확인해보기

memory.load_memory_variables({})["history"]
```

```
[HumanMessage(content='설치가 완료되면 어떻게 해야 하나요?', additional_kwargs={}, response_metadata={}),
 AIMessage(content='설치가 완료되면, 전원을 켜고 초기 구동 테스트를 진행해 주시기 바랍니다. 테스트 절차는 매뉴얼의 10페이지에 설명되어
있습니다. 만약 기계에 이상이 있거나 추가적인 지원이 필요하시면 언제든지 연락 주시기 바랍니다.', additional_kwargs={},
response_metadata={}),
 HumanMessage(content='감사합니다, 도움이 많이 되었어요!', additional_kwargs={}, response_metadata={}),
 AIMessage(content='언제든지 도와드릴 준비가 되어 있습니다. 추가적인 질문이나 지원이 필요하시면 언제든지 문의해 주세요. 좋은 하루 되세
요!', additional_kwargs={}, response_metadata={})]
```

```
print(type(memory.load_memory_variables({})["history"]))                # <class 'list'>
```

- 셀 출력

```
[HumanMessage(content='설치가 완료되면 어떻게 해야 하나요?', additional_kwargs={}, response_metadata={}),
AIMessage(content='설치가 완료되면, 전원을 켜고 초기 구동 테스트를 진행해 주시기 바랍니다. 테스트 절차는 매뉴얼의 10페이지에 설명되어
HumanMessage(content='감사합니다, 도움이 많이 되었어요!', additional_kwargs={}, response_metadata={}),
AIMessage(content='언제든지 도와드릴 준비가 되어 있습니다. 추가적인 질문이나 지원이 필요하시면 언제든지 문의해 주세요. 좋은 하루 되세
```

```
# 가독성 좋게 출력해보기

from langchain.schema import HumanMessage, AIMessage

def pretty_print_history_list(history):
```

```
    """
    List[HumanMessage|AIMessage] 형태의 history를
    가독성 좋게 포맷팅해 출력합니다.
    """
    print("\n📜 최근 대화(최대 k턴)\n")
    for idx, msg in enumerate(history, start=1):
        if isinstance(msg, HumanMessage):
            role, emoji = "사용자", "👤"
        elif isinstance(msg, AIMessage):
            role, emoji = "어시스턴트", "🤖"
        else:
            role, emoji = msg.__class__.__name__, ""

        print(f"{idx:02d}. {emoji} {role}:")
        for line in msg.content.split("\n"):
            print("    " + line)
        print()

#── 호출 예시 ──#
history_list = memory.load_memory_variables({})["history"]
pretty_print_history_list(history_list)
```

- 셀 출력

```
📜 최근 대화(최대 k턴)

01. 👤 사용자:
    설치가 완료되면 어떻게 해야 하나요?

02. 🤖 어시스턴트:
    설치가 완료되면, 전원을 켜고 초기 구동 테스트를 진행해 주시기 바랍니다. 테스트 절차는 매뉴얼의 10페이지에 설명되어 있습니다. 만약 기

03. 👤 사용자:
    감사합니다, 도움이 많이 되었어요!

04. 🤖 어시스턴트:
    언제든지 도와드릴 준비가 되어 있습니다. 추가적인 질문이나 지원이 필요하시면 언제든지 문의해 주세요. 좋은 하루 되세요!
```

`ConversationTokenBufferMemory` 사용 중단 예정

- `LangChain`의 최신 버전(`0.3.1` 이후)에서 `deprecated`(사용 중단 예정)
- `langchain==1.0.0`에서 **완전히 제거될 예정**
  - 이는 메모리 관리를 더 유연하고 효율적으로 처리하기 위한 변경
  - 대신 **`trim_messages`** 함수를 사용해 대화 히스토리를 토큰 제한에 맞게 잘라내는 방식으로 대체

왜 deprecated되었나?

- 기존 `ConversationTokenBufferMemory`는 대화의 최근 메시지를 토큰 제한에 맞게 유지했지만, **이제는 LangGraph나 LCEL (LangChain Expression Language)을 활용해 더 세밀한 제어가 가능**
- `trim_messages`를 사용하면 시스템 메시지를 유지하면서 최근 메시지만 토큰 제한 내로 유지

```
from langchain_core.messages import AIMessage, HumanMessage, SystemMessage, trim_messages
from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_core.runnables import RunnablePassthrough, RunnableLambda
```

```
from langchain_google_genai import ChatGoogleGenerativeAI
# ㄴ gpt(OpenAI) 사용할 경우 from langchain_openai import ChatOpenAI
from langgraph.checkpoint.memory import MemorySaver
from langgraph.graph import START, MessagesState, StateGraph
import uuid

# LLM 생성
gemini_lc = ChatGoogleGenerativeAI(
    model="gemini-2.5-flash-lite",
    temperature=0.7,
    max_output_tokens=4096,
    )

# LangGraph 그래프 정의
workflow = StateGraph(state_schema=MessagesState)

# API KEY를 환경변수로 관리하기 위한 설정 파일
from dotenv import load_dotenv

# API KEY 정보 로드
load_dotenv()
from langchain.memory import ConversationTokenBufferMemory

# 모델 호출 함수 (trim_messages로 토큰 제한 적용)
def call_model(state: MessagesState):
    # 대화 히스토리에서 토큰 제한 적용 (max_tokens=150)
    selected_messages = trim_messages(
        state["messages"],
        token_counter=gemini_lc,            # LLM을 사용해 토큰 수 계산
        max_tokens=150,                     # 최대 토큰 제한
        strategy="last",                    # 최근 메시지부터 유지
        start_on="human",                   # 유효한 대화 형식 유지
        include_system=True                 # 시스템 메시지 유지
    )

    # 프롬프트 템플릿
    prompt = ChatPromptTemplate.from_messages([
        # 시스템 메시지
        SystemMessage(content="You are a movie recommendation expert. Your job is to recommend movies to the use
        MessagesPlaceholder(variable_name="messages")
    ])

    # 체인 구성
    chain = prompt | gemini_lc
    response = chain.invoke(selected_messages)
    return {"messages": response}

# 노드와 엣지 추가
workflow.add_edge(START, "model")
workflow.add_node("model", call_model)

# 메모리 설정 (대화 지속성을 위해)
memory = MemorySaver()
app = workflow.compile(checkpointer=memory)

# 대화 실행 예시
thread_id = str(uuid.uuid4())  # 고유 스레드 ID
config = {"configurable": {"thread_id": thread_id}}

# 첫 번째 메시지
input_message = HumanMessage(content="주말에 볼 만한 로맨스 영화를 추천해 주세요.")
for event in app.stream({"messages": [input_message]}, config, stream_mode="values"):
    event["messages"][-1].pretty_print()
```

- 셀 출력 (4.6s)

```
================================ **Human Message** ================================

주말에 볼 만한 로맨스 영화를 추천해 주세요.
================================== **Ai Message** ==================================

주말에 볼 만한 로맨스 영화를 찾으시는군요! 어떤 분위기의 로맨스를 좋아하시나요? 몇 가지 질문에 답해주시면 더 정확한 추천을 해드릴 수 있습니

혹시 이런 종류의 로맨스를 좋아하시나요?

*   **달달하고 설레는 로맨스:** 처음 만난 두 사람이 사랑에 빠지는 과정이 예쁜 영화
*   **현실적이고 공감 가는 로맨스:** 현실적인 연애의 어려움과 극복 과정을 그린 영화
*   **코믹하고 유쾌한 로맨스:** 웃음과 함께 달콤함을 느낄 수 있는 영화
*   **애틋하고 감동적인 로맨스:** 가슴 뭉클한 이야기와 여운이 남는 영화
*   **판타지/SF적인 요소가 가미된 로맨스:** 현실과는 다른 특별한 설정 속에서 피어나는 사랑 이야기

혹은, 특정 배우나 감독을 좋아하시나요? 아니면 최근에 재미있게 본 로맨스 영화가 있으신가요?

우선, 몇 가지 인기 있고 많은 분들이 좋아하는 로맨스 영화를 장르별로 추천해 드릴게요!

**1. 달달하고 설레는 로맨스:**

*   **어바웃 타임 (About Time, 2013):** 시간을 되돌릴 수 있는 능력을 가진 남자가 사랑하는 여자를 만나면서 벌어지는 이야기. 따뜻하
*   **라라랜드 (La La Land, 2016):** 꿈을 좇는 두 젊은 남녀의 로맨스와 성장을 그린 뮤지컬 영화. 아름다운 영상미와 음악, 그리고 현
*   **노트북 (The Notebook, 2004):** 오랜 시간이 흘러도 변치 않는 운명적인 사랑 이야기. 가슴 시린 로맨스를 좋아하신다면 강력 추천

**2. 현실적이고 공감 가는 로맨스:**

*   **그녀 (Her, 2013):** 외로운 남자가 인공지능 운영체제와 사랑에 빠지는 이야기. 독특한 설정이지만, 현대 사회의 외로움과 소통에 대한
*   **싱글 맨 (A Single Man, 2009):** 사랑하는 사람을 잃은 교수가 하루 동안 겪는 이야기. 절제된 감정선과 아름다운 영상미로 잔잔하
*   **비포 선라이즈 (Before Sunrise, 1995) / 비포 선셋 (Before Sunset, 2004) / 비포 미드나잇 (Before Midnight, 201

**3. 코믹하고 유쾌한 로맨스:**

*   **러브 액츄얼리 (Love Actually, 2003):** 여러 커플의 다양한 사랑 이야기를 옴니버스 형식으로 보여주는 영화. 크리스마스 시즌에
*   **500일의 썸머 (500 Days of Summer, 2009):** 사랑에 대한 환상을 가진 남자와 현실적인 여자의 500일간의 이야기. 달콤함과 씁

**4. 애틋하고 감동적인 로맨스:**

*   **타이타닉 (Titanic, 1997):** 역사적인 사건을 배경으로 펼쳐지는 비극적이면서도 아름다운 사랑 이야기. 클래식 로맨스의 대명사죠.
*   **냉정과 열정 사이 (冷静と情熱のあいだ, 2001):** 이탈리아를 배경으로 펼쳐지는 두 남녀의 애틋한 사랑 이야기. 아름다운 풍경과 함께

어떤 영화가 가장 끌리시나요? 더 구체적인 취향을 알려주시면 더욱 만족스러운 영화를 추천해 드릴 수 있습니다! 😊


# 두 번째 메시지 (이전 대화 기억 확인)

input_message = HumanMessage(content="그중에서 2010년대 영화를 알려줘.")
for event in app.stream({"messages": [input_message]}, config, stream_mode="values"):
    event["messages"][-1].pretty_print()
```

- 셀 출력 (4.3s)

```
================================ **Human Message** ================================

그중에서 2010년대 영화를 알려줘.
================================== **Ai Message** ==================================

2010년대 영화를 찾으시는군요! 어떤 장르나 분위기의 영화를 좋아하시나요? 아니면 혹시 특정 배우나 감독을 염두에 두고 계신가요?
```

혹시 특별히 원하시는 것이 없다면, 제가 생각하는 2010년대의 명작 또는 화제작 몇 편을 추천해 드릴게요.

*   **봉준호 감독의 <기생충> (Parasite, 2019)**: 칸 영화제 황금종려상과 아카데미 작품상을 수상하며 세계적인 신드롬을 일으킨 영화입니
*   **크리스토퍼 놀란 감독의 <인셉션> (Inception, 2010)**: 꿈속에서 아이디어를 심는다는 독창적인 설정과 압도적인 스케일, 그리고 복
*   **알폰소 쿠아론 감독의 <그래비티> (Gravity, 2013)**: 우주 공간에서의 생존을 그린 영화로, 극강의 몰입감과 시각적인 아름다움으로
*   **쿠엔틴 타란티노 감독의 <원스 어폰 어 타임... 인 할리우드> (Once Upon a Time in Hollywood, 2019)**: 1960년대 할리우드
*   **조나단 글레이저 감독의 <언더 더 스킨> (Under the Skin, 2013)**: 외계에서 온 존재가 인간을 유혹하며 벌어지는 기이하고 몽환적

이 외에도 2010년대에는 정말 훌륭한 영화들이 많습니다. 혹시 더 구체적인 취향을 알려주시면 더 맞춤화된 추천을 해드릴 수 있습니다!

```
# 두 번째 대화 (이전 대화 기억 확인)

input_message = HumanMessage(content="감독이 '크리스토퍼 놀란'인 영화도 추천해 줄 수 있나요?")
for event in app.stream({"messages": [input_message]}, config, stream_mode="values"):
    event["messages"][-1].pretty_print()
```

- 셀 출력 (4.9s)

```
================================ **Human Message** =================================

감독이 '크리스토퍼 놀란'인 영화도 추천해 줄 수 있나요?
================================== **Ai Message** ==================================

네, 당연히 크리스토퍼 놀란 감독의 영화도 추천해 드릴 수 있습니다!

크리스토퍼 놀란 감독은 복잡한 스토리텔링, 독창적인 비주얼, 그리고 깊이 있는 주제를 다루는 것으로 유명하죠. 그의 영화들은 많은 팬들에게 사랑

어떤 종류의 영화를 좋아하시는지, 혹은 최근에 재미있게 본 놀란 감독의 영화가 있다면 알려주시면 더 맞춤화된 추천을 해드릴 수 있습니다.

혹시 특별히 찾으시는 분위기나 장르가 있으신가요? 예를 들어:

*   **SF/액션:** 시간, 공간, 현실 등을 다루는 영화를 좋아하시나요?
*   **스릴러/미스터리:** 반전이 있거나 긴장감 넘치는 영화를 찾으시나요?
*   **드라마/인간 심리:** 캐릭터의 내면이나 복잡한 관계를 다루는 영화를 선호하시나요?

일단, 크리스토퍼 놀란 감독의 대표작 몇 편을 먼저 소개해 드릴게요. 아직 보지 못하신 영화가 있다면 이 중에서 골라보시는 것도 좋을 것 같습니다

*   **인셉션 (Inception, 2010):** 꿈속에서 생각을 심는다는 독특한 소재와 압도적인 비주얼이 돋보이는 SF 액션 스릴러입니다.
*   **인터스텔라 (Interstellar, 2014):** 인류의 생존을 위해 우주를 탐험하는 과정을 그린 감동적인 SF 영화입니다. 과학적인 고증과
*   **다크 나이트 (The Dark Knight, 2008):** 슈퍼히어로 영화의 새로운 지평을 열었다고 평가받는 배트맨 시리즈의 최고작 중 하나입니
*   **메멘토 (Memento, 2000):** 단기 기억상실증에 걸린 주인공이 아내를 죽인 범인을 찾는 과정을 역순으로 보여주는 독창적인 스릴러입니
*   **덩케르크 (Dunkirk, 2017):** 제2차 세계대전 당시 덩케르크 철수 작전을 생생하게 그린 전쟁 영화입니다. 시간의 흐름을 교차하며 긴
*   **테넷 (Tenet, 2020):** 시간을 거스르는 '인버전'이라는 독특한 설정을 가진 SF 액션 영화입니다. 복잡한 플롯과 스타일리시한 액션을

이 외에도 **프레스티지 (The Prestige, 2006)**, **배트맨 비긴즈 (Batman Begins, 2005)**, **다크 나이트 라이즈 (The Dark

어떤 영화에 가장 관심이 가시나요? 아니면 혹시 이미 보신 영화가 있다면 알려주세요!
```

설명

- **`trim_messages`**: 대화 메시지를 **토큰 제한(150)** 에 맞게 최근 메시지만 유지
- **`strategy="last"`** 로 최근 메시지를 우선으로 함

- `LangGraph` : 대화 상태를 관리
  - **메모리**(예: `MemorySaver`)를 통해 세션 간 대화가 지속
  - 기존 `ConversationTokenBufferMemory`의 기능을 대체
- 실행 결과 예시: 첫 메시지 두번째 대화를 이어가면 `AI`가 이전 대화를 기억해 대답
- **커스터마이징** :
  - `max_tokens`를 조정
  - `token_counter`를 다른 LLM으로 변경
  - 더 복잡한 로직(예: 요약 추가)이 필요하면 **LangGraph의 how-to 가이드** 를 참고

- *next: 대화 엔티티 메모리(ConversationEntityMemory)*
