



2024

Mini project

국내 여행지 추천

배누리, 안준영, 정기현, 박현아

2024.10.07

CONTENTS



01

배경 및 목표

02

데이터 및 코드 설명

03

주요 아키텍처

04

결과



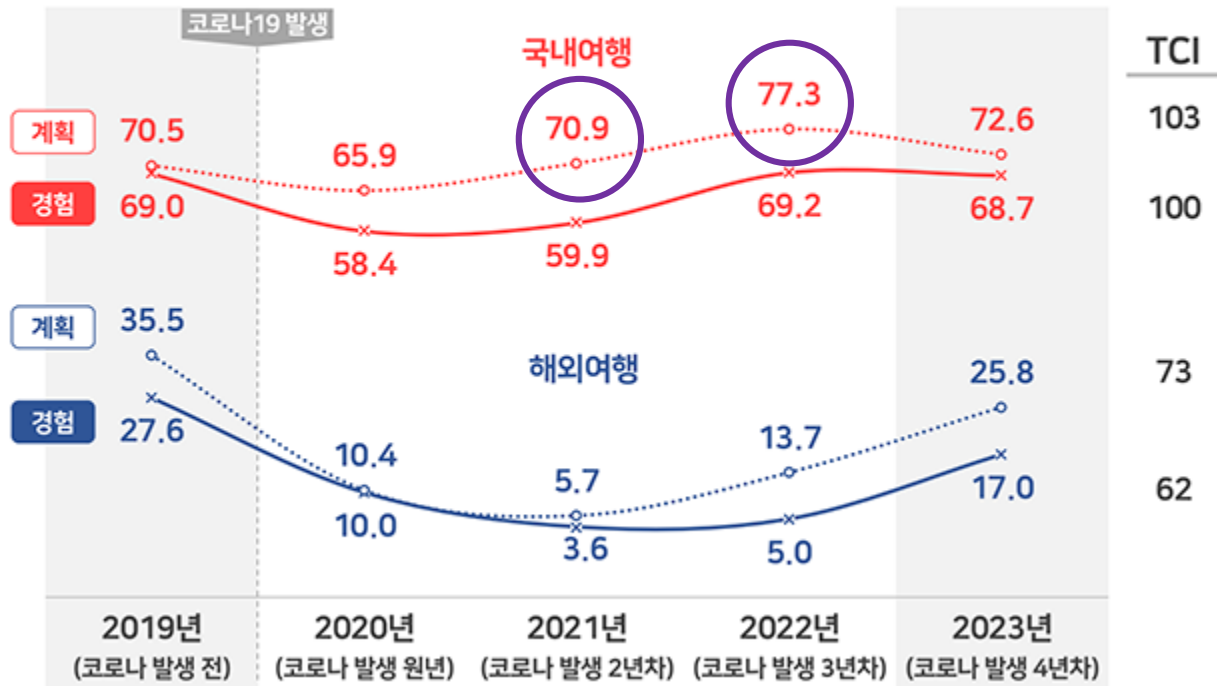
01

PART ONE .

배경 및 목표

주제 선정 배경

[그림1] 2019~2023 국내·해외여행 경험·계획률 추이



출처: 컨슈머인사이트

- 코로나 2년차인 '21년의 여행계획률은 코로나 전 수준을 넘었고, '22년에는 신기록을 세우며 피크에 달함.

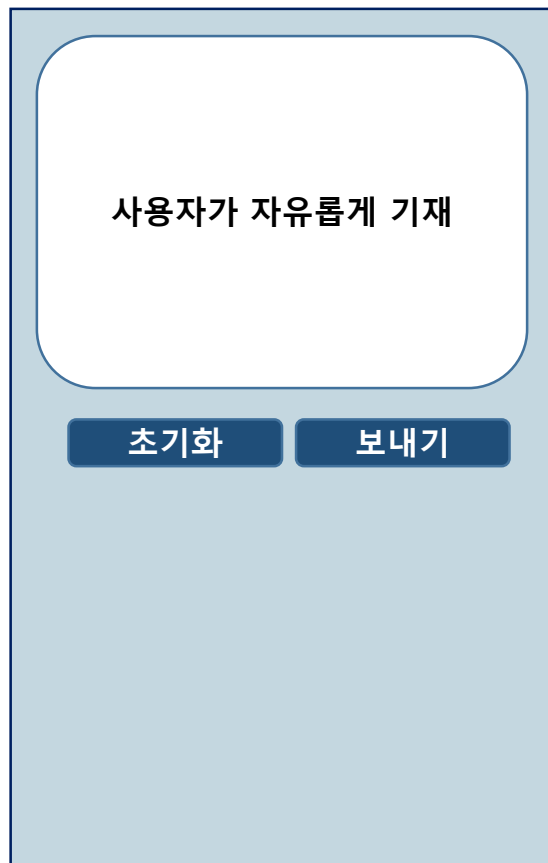
- '23년에는 다소 감소했으나 여행경험률은 3년차인 '22년에 '19년 수준에 이르러 TCI 100 수준을 유지하고 있음.

➡ 국내 여행지 추천

목표

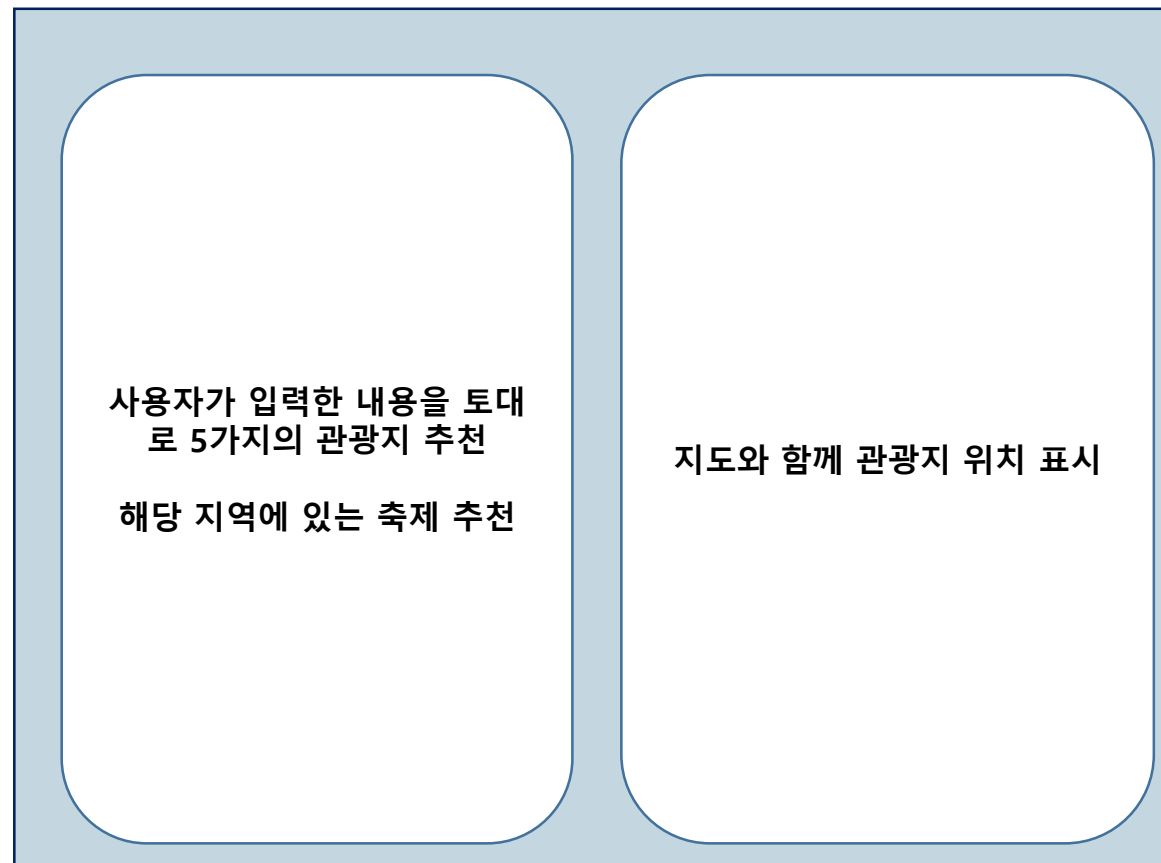
- 사용자의 입력에 따라 국내 여행지를 추천하는 모델 만들기
- 랭체인 및 streamlit을 이용한 챗봇 구현

(예시)



A diagram of the input interface. It consists of a light blue rounded rectangle containing a white rounded rectangle for text input. Below the input area are two dark blue buttons with white text: '초기화' (Reset) and '보내기' (Send).

input



A diagram of the output interface. It is a light blue rounded rectangle divided into two vertical white rounded rectangles. The left rectangle contains the text: '사용자가 입력한 내용을 토대로 5가지의 관광지 추천' and '해당 지역에 있는 축제 추천'. The right rectangle contains the text: '지도와 함께 관광지 위치 표시'.

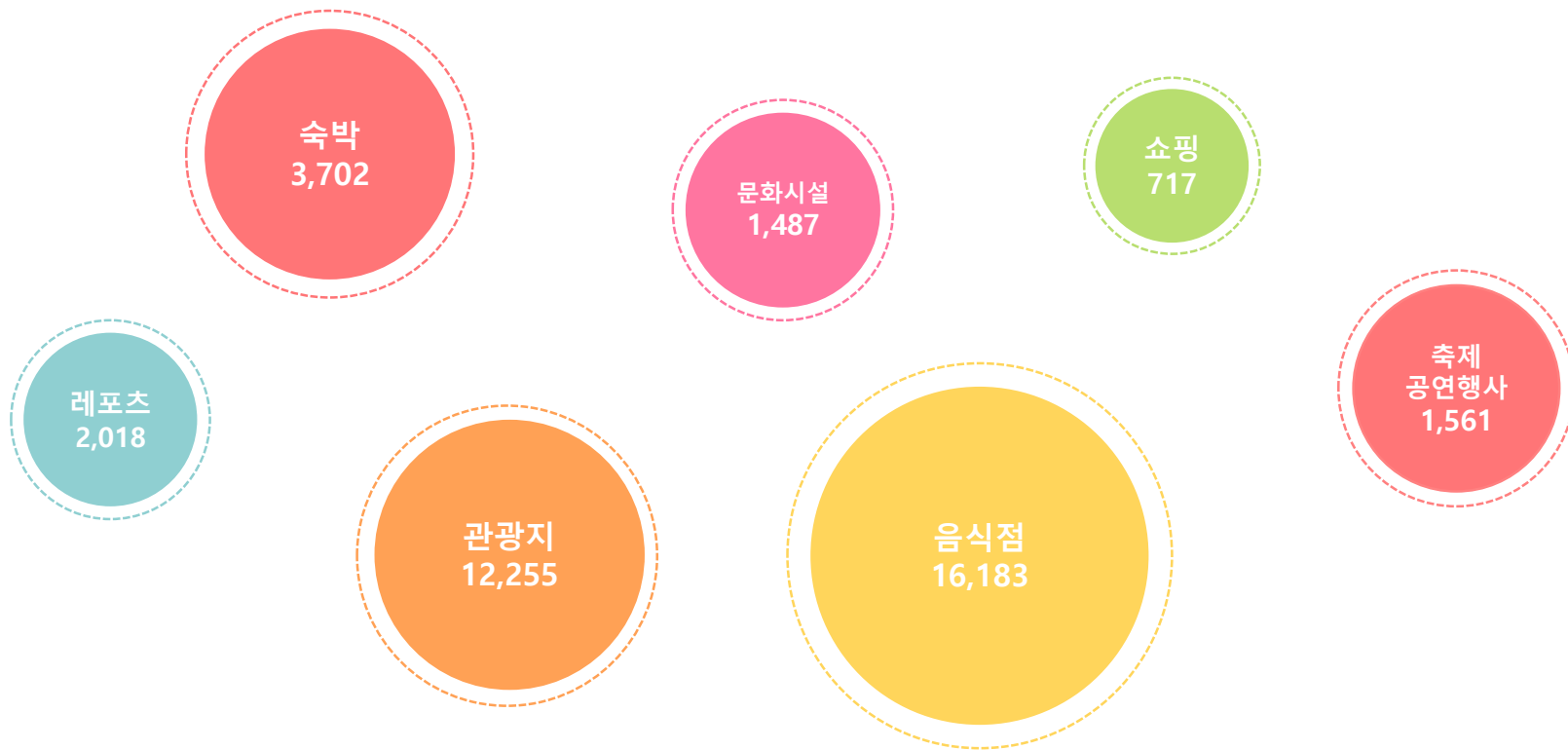
output

02

PART TWO .

데이터 및 코드 설명

데이터 – Tour API 4.0



관광지 데이터 : 36,435개의 행, 7개의 열

축제 데이터 : 1,561개의 행, 7개의 열

데이터 전처리

```
# '명칭', '주소', '위도', '경도', '개요', '대분류', '중분류'
import pandas as pd
import os

# 데이터 로드
df1 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\축제공연행사_축제.csv', encoding='cp949')
df2 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_전통공연.csv', encoding='cp949')
df3 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_전시회.csv', encoding='cp949')
df4 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_오페라.csv', encoding='cp949')
df5 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_연극.csv', encoding='cp949')
df6 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_영화제.csv', encoding='cp949')
df7 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_박람회.csv', encoding='cp949')
df8 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_뮤지컬.csv', encoding='cp949')
df9 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_무용.csv', encoding='cp949')
df10 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_대중콘서트.csv', encoding='cp949')
df11 = pd.read_csv('C:\\ai5\\미니프로젝트\\_data\\TourAPI\\공연행사_기타행사.csv', encoding='cp949')
```

```
#필요한 열만 추출
df1 = df1[['명칭', '주소', '위도', '경도', '개요']]
df2 = df2[['명칭', '주소', '위도', '경도', '개요']]
df3 = df3[['명칭', '주소', '위도', '경도', '개요']]
df4 = df4[['명칭', '주소', '위도', '경도', '개요']]
df5 = df5[['명칭', '주소', '위도', '경도', '개요']]
df6 = df6[['명칭', '주소', '위도', '경도', '개요']]
df7 = df7[['명칭', '주소', '위도', '경도', '개요']]
df8 = df8[['명칭', '주소', '위도', '경도', '개요']]
df9 = df9[['명칭', '주소', '위도', '경도', '개요']]
df10 = df10[['명칭', '주소', '위도', '경도', '개요']]
df11 = df11[['명칭', '주소', '위도', '경도', '개요']]
```

```
# 열 생성
df1['대분류'] = '축제'
df1['중분류'] = '축제'
df2['중분류'] = '전통공연'
df3['중분류'] = '전시회'
df4['중분류'] = '오페라'
df5['중분류'] = '연극'
df6['중분류'] = '영화제'
df7['중분류'] = '박람회'
df8['중분류'] = '뮤지컬'
df9['중분류'] = '무용'
df10['중분류'] = '대중콘서트'
df11['중분류'] = '기타행사'
```

```
#공연행사 데이터 병합
perform = pd.concat([df2, df3, df4, df5, df6, df7, df8, df9, df10, df11], ignore_index=True)
perform['대분류'] = '공연행사'

#데이터 중복값 제거
perform = perform.drop_duplicates(subset=['명칭', '주소', '위도', '경도', '개요'])
```

- 세분화 된 데이터 로드
- 각 데이터들에서 필요한 열만 추출
- 분류하기 위하여 새로운 열 생성
- 관련 데이터 병합

-> 숙박, 문화시설, 레포츠, 관광지, 쇼핑, 음식점에서도
동일하게 수행 후 모든 데이터 병합 (축제 제외)

핵심 코드 설명

```
# 모델 로드
model = SentenceTransformer('jhgan/ko-sroberta-multitask')

path = 'C:/ai5/미니프로젝트/_data/TourAPI/'

df1 = pd.read_csv(path + '문화시설_쇼핑데이터_.csv')
df2 = pd.read_csv(path + '음식점데이터.csv')
df3 = pd.read_csv(path + '관광지데이터.csv')
df4 = pd.read_csv(path + '레포츠타데이터.csv')
df5 = pd.read_csv(path + '숙박.csv')
festival_df = pd.read_csv(path + '축제.csv', on_bad_lines='skip')

## 결측치 처리 ##
df = pd.concat([df1, df2, df3, df4, df5], axis=0)
print(df.isnull().sum())

df['광역시/도'] = df['주소'].apply(lambda x: ' '.join(x.split()[:2]))
festival_df['광역시/도'] = festival_df['주소'].apply(lambda x: ' '.join(x.split()[:2]))

# 임베딩 생성
df['embedding'] = df['대분류'].map(lambda x: list(model.encode(x)))
df['embedding1'] = df['중분류'].map(lambda x: list(model.encode(x)))
df['embedding2'] = df['광역시/도'].map(lambda x: list(model.encode(x)))
df['embedding3'] = df['개요'].map(lambda x: list(model.encode(x)))
```

```
명칭      0
주소      0
위도      0
경도      0
개요      0
대분류    0
중분류    0
```

- 한국어 자연어 처리를 위해 설계된 사전 훈련된 transformer 모델 로드
- 데이터 로드 후 결측치 처리 (결측치 없음)
- 상세 주소를 간단한 주소로만 표시

예) 대구광역시 동구 팔공산로185길 11 -> 대구광역시 동구

- 각 열 별로 임베딩 생성
- 임베딩 생성한 열의 shape는 각각 (36435,)로 데이터 개수와 동일

model.encode : 문장을 벡터로 변환하는 함수

→ 결측치 출력 화면

핵심 코드 설명

```
# 가중치를 적용한 유사성 계산 함수
def chatbot(user):
    # 입력된 텍스트의 임베딩
    embedding = model.encode(user)

    # 각 임베딩별 유사도 계산
    df['similarity'] = df['embedding'].map(lambda x: cosine_similarity([embedding], [x]).squeeze())
    df['similarity_other_embedding_1'] = df['embedding1'].map(lambda x: cosine_similarity([embedding], [x]).squeeze())
    df['similarity_other_embedding_2'] = df['embedding2'].map(lambda x: cosine_similarity([embedding], [x]).squeeze())
    df['similarity_other_embedding_3'] = df['embedding3'].map(lambda x: cosine_similarity([embedding], [x]).squeeze())

    # 가중치를 설정 (원하는 비율로 조정 가능)
    weight1 = 0.05 # 대분류 가중치
    weight2 = 0.5 # 중분류 가중치
    weight3 = 0.2 # 광역시/도 가중치
    weight4 = 0.15 # 개요 가중치

    # 총합 유사도 계산
    df['total_similarity'] = (df['similarity'] * weight1 +
                             df['similarity_other_embedding_1'] * weight2 +
                             df['similarity_other_embedding_2'] * weight3 +
                             df['similarity_other_embedding_3'] * weight4)

    # 총합 유사도가 높은 상위 5개 관광지명 추출
    top_places = df.nlargest(5, 'total_similarity')[['명칭', '광역시/도', '개요']]

    # 요약된 설명 생성 (각 개요에 대해 요약을 적용)
    top_places['요약'] = top_places['개요'].apply(lambda x: remove_html_tags(summarizer.summarize(x, words=50)) if len(x.split()) > 50 else remove_html_tags(x))
```

- 사용자가 입력한 텍스트를 임베딩 벡터로 변환함(사용자의 입력 수치화)
- 각 임베딩 열에 대한 코사인 유사도 계산
- 각 유사도에 가중치 부여 -> 총합 유사도 계산
- 상위 관광지 5개 추출 및 각 관광지에 대한 요약 출력

유사도 방식

1. Cosine_similarity

벡터 간의 각도와 두 벡터의 크기를 사용해 각도 기반의 유사도를 구하는 방식

1에 가까울수록 두 벡터가 유사함 (즉, 두 벡터의 방향이 거의 동일).

0에 가까울수록 두 벡터는 서로 무관함.

-1에 가까울수록 두 벡터가 반대 방향을 가리킴.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

2. 유클리드 거리

두 점 사이의 직선 거리를 측정하는 방법, 하지만 차원이 고차원이 될수록 의미가 희미해진다는 한계가 있음.

3. 자카드 유사도

벡터간의 교집합을 합집합으로 나눈 것, 0~1의 값을 가짐.

핵심 코드 설명

```
# 관광지명과 요약된 설명 리스트 생성
recommended_info = []
for idx, row in enumerate(top_places.itertuples(), start=1):
    recommended_info.append(f"{idx}. {row.명칭}\n {row.요약}")

recommended_places_str = '\n\n'.join(recommended_info)

# 상위 관광지 중 첫 번째 관광지의 광역시/도에서 열리는 축제 추천
top_tourist_region = top_places.iloc[0]['광역시/도']
related_festivals = festival_df[festival_df['광역시/도'] == top_tourist_region][:5]

festivals_list = related_festivals['명칭'].tolist()
festivals_str = ', '.join(festivals_list)
```

```
##### 지도 생성 #####
xy_coord = []
for place in top_places['명칭']:
    index = df[df['명칭'] == place].index[0]
    latitude = df.loc[index, '위도']
    longitude = df.loc[index, '경도']
    xy_coord.append({'lon': longitude, 'lat': latitude, 'name': place})

# 지도 객체 생성 (구글 맵 타일 설정)
m = folium.Map(location=[xy_coord[0]['lat'], xy_coord[0]['lon']], zoom_start=10, tiles='OpenStreetMap')

# 구글 맵 타일 추가
folium.TileLayer(tiles='http://mt1.google.com/vt/lyrs=r&x={x}&y={y}&z={z}',
                attr='Google', name='Google Map', overlay=True, control=True).add_to(m)

# 마커 추가
for loc in xy_coord:
    folium.Marker([loc['lat'], loc['lon']], popup=loc['name']).add_to(m)

# 지도 HTML 반환
map_html = m._repr_html_()
#####
```

```
# Gradio 인터페이스 생성
demo = gr.Interface(fn=chatbot, inputs='text', outputs=['text', 'html'])
demo.launch(debug=True, share=True)
```

- 관광지명과 각 관광지에 대한 요약을 문자열로 만들어 출력
출력 예) {번호}{관광지명}{설명}
- 관광지 중 첫 번째 관광지에서 열리는 축제 추천 출력
- 추천된 관광지들을 지도에 마커로 표시
- 구글 맵 타일을 적용한 지도를 생성하여 HTML 형식으로 반환

랭체인 코드 설명

```
# Streamlit 페이지 설정 (제목 및 아이콘)
st.set_page_config(page_title="국내 여행지 추천 🗺️", page_icon="🗺️")
st.title("국내 여행지 추천 🗺️") # 페이지 타이틀 표시

# 세션 상태에서 대화 내용이 없을 경우 빈 리스트로 초기화
if "messages" not in st.session_state:
    st.session_state["messages"] = []

# 세션 상태에 저장된 대화 내역을 출력하는 함수
def print_history():
    for msg in st.session_state["messages"]:
        st.chat_message(msg.role).write(msg.content) # 사용자와 AI의 메시지를 출력

# 대화 내역을 세션 상태에 추가하는 함수
def add_history(role, content):
    st.session_state["messages"].append(ChatMessage(role=role, content=content)) # role에 따라 'user' 또는 'ai'로 구분

# 체인을 생성하는 함수 (프롬프트, 모델 연결)
def create_chain(prompt, model):
    chain = prompt | ChatOpenAI(model_name=model, api_key="api_key") | StrOutputParser() # OpenAI API와 파싱 체인 생성
    return chain
```

- 처음 실행될 때 messages라는 변수를 빈 리스트로 초기화하여 사용자와 AI 간의 대화 기록을 저장
- 세션 상태에 저장된 메시지를 사용자(user)와 AI(ai)의 역할에 따라 구분해서 표시
- ChatOpenAI: OpenAI의 GPT 모델을 사용하여 AI 응답을 생성함
- StrOutputParser: 모델의 출력 결과를 문자열로 파싱함

랭체인 코드 설명

```
# 사이드바에 대화 초기화 버튼과 프롬프트 설정 인터페이스를 추가
with st.sidebar:
    clear_btn = st.button("대화내용 초기화")

    # 프롬프트 설정 UI
    prompt = """당신은 친절한 AI 어시스턴트입니다. 사용자의 질문에 간결하게 답변해 주세요."""
    user_text_prompt = st.text_area("프롬프트", value=prompt)
    user_text_apply_btn = st.button("프롬프트 적용", key="apply1")

    # 프롬프트 적용 로직
    if user_text_apply_btn:
        st.markdown(f"✅ 프롬프트가 적용되었습니다")
        prompt_template = user_text_prompt + "\n\n#Question:\n{question}\n\n#Answer:"
        prompt = PromptTemplate.from_template(prompt_template)
        st.session_state["chain"] = create_chain(prompt, "gpt-4") # 수정된 프롬프트는 PromptTemplate을 통해 템플릿화되고, AI 모델과의 체인이 새로 생성

# 대화내용 초기화 버튼이 눌리면 세션 상태의 대화 내용 초기화
if clear_btn:
    retriever = st.session_state["messages"].clear()

# 이전 대화 내역 출력
print_history()
```

대화내용 초기화

프롬프트

당신은 친절한 국내 여행 추천 AI입니다. 사용자의 질문에 자세하게 답변해주세요.

프롬프트 적용

✅ 프롬프트가 적용되었습니다

- 사이드바에 대화 내용 초기화 버튼과 프롬프트 수정 기능을 제공함
- 초기화 버튼을 누르면 세션 상태에 저장된 대화 내용이 모두 삭제됨

랭체인 코드 설명

```
# 세션 상태에 체인이 없다면 기본 체인을 생성
if "chain" not in st.session_state:
    prompt_template = user_text_prompt + "\n\n#Question:\n{question}\n\n#Answer:" # 기본 프롬프트 템플릿
    prompt = PromptTemplate.from_template(prompt_template) # 템플릿 생성
    st.session_state["chain"] = create_chain(prompt, "gpt-4") # 체인 생성 후 세션에 저장

# 사용자가 입력창에 질문을 입력했을 때
if user_input := st.chat_input():
    add_history("user", user_input) # 입력된 질문을 세션 상태에 저장
    st.chat_message("user").write(user_input) # 사용자 메시지 출력

    # AI의 답변 처리
    with st.chat_message("assistant"):
        chat_container = st.empty() # 응답을 실시간으로 표시할 공간 생성

        # AI 모델의 스트리밍 응답 처리
        stream_response = st.session_state["chain"].stream({"question": user_input}) # 질문에 대한 답변 스트리밍
        ai_answer = ""

        # 스트리밍된 AI 응답을 차례로 화면에 출력
        for chunk in stream_response:
            ai_answer += chunk # 스트리밍된 답변을 하나의 문자열로 결합
            chat_container.markdown(ai_answer) # 실시간으로 화면에 표시

        # AI 응답을 세션 상태에 저장
        add_history("ai", ai_answer)
```

- 기본 프롬프트를 사용해 GPT-4 모델과 연결된 체인을 생성



03

PART THREE .

주요 아키텍처

자연어 처리 딥러닝 모델 변천 과정

RNN (1986)

- Recurrent Neural Network
- 고정된 길이의 문맥벡터 사용
- 마지막 은닉층의 값이 문맥벡터

Seq2Seq (2014)

- 시퀀스 입력을 통해 다른 시퀀스 출력을 얻도록 고안된 모델
- 기계번역에 주로 사용되는 모델

Transformer (2017)

- Attention만을 활용하여 구성한 Encoder-Decoder 모델
- 연산효율성과 기계 번역 품질에서 높은 성능을 낸 모델
- BERT, GPT의 기반 모델

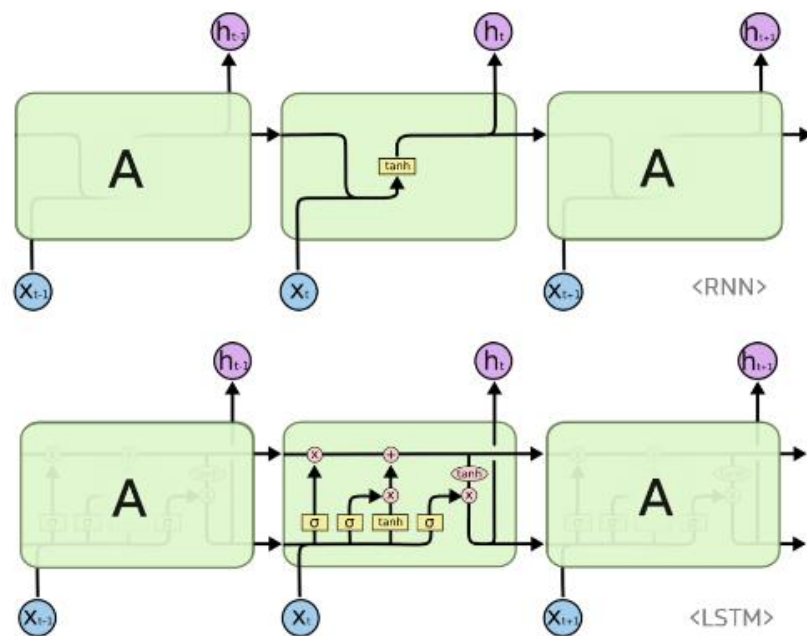
LSTM (1997)

- Long Short-Term Memory
- RNN에서 장기적 기억을 더 잘하도록 변환된 신경망

Attention (2015)

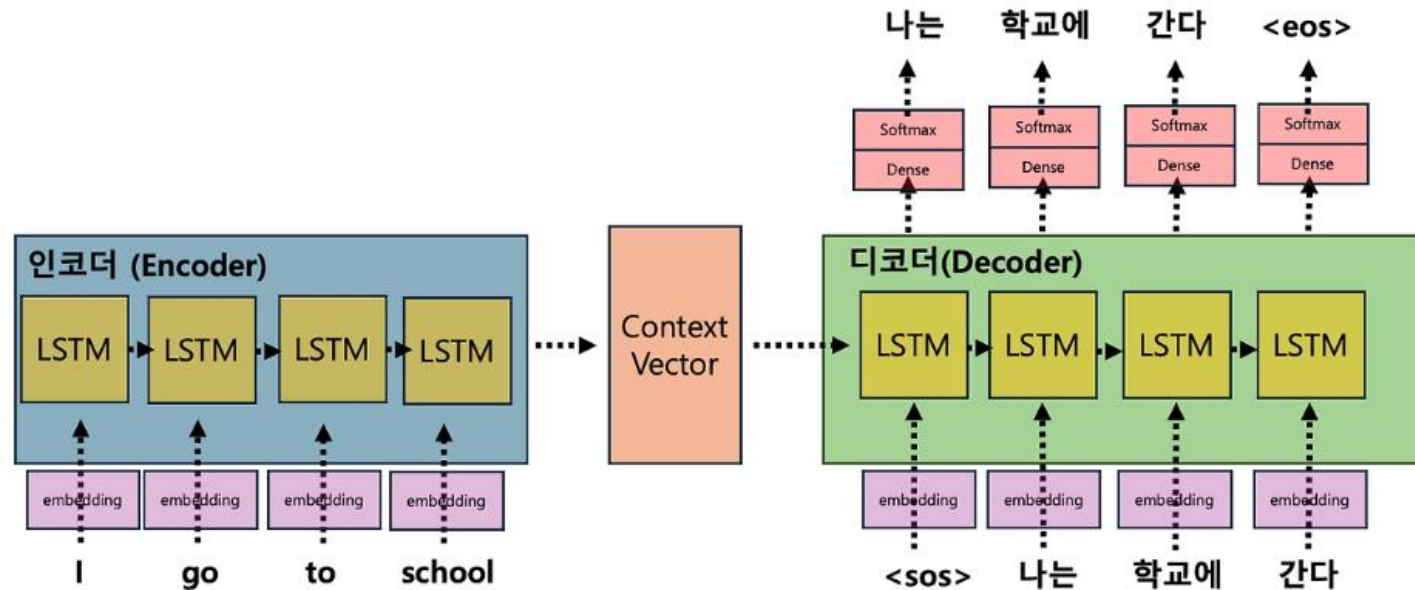
- 고정길이 문맥벡터를 사용하지 않아 Bottleneck 문제를 보완
- 전체 입력 시퀀스에서 '주의'를 기울여야하는 부분에 초점을 맞춰 학습

LSTM의 한계



- 단일 시퀀스를 입력받고, 해당 시퀀스를 처리하여 결과를 내는 구조임.
- 기본적인 LSTM 구조에서는 입력 시퀀스와 출력 시퀀스가 동일한 길이를 가져야 처리할 수 있으며 한 번에 하나의 값을 출력하는 방식.
- 시퀀스 데이터를 처리하는 데 매우 효과적이지만, **입력과 출력 길이가 크게 다른 경우에는 모델이 충분히 모든 정보를 반영하지 못할 수 있음.**
- 입력 시퀀스가 길거나 복잡할 경우, 각 타임스텝의 정보를 잊어버리거나 학습에 어려움을 겪을 수 있음.

Seq2Seq



- Seq2Seq는 두 개의 LSTM 네트워크로 구성되며, 인코더(Encoder)와 디코더(Decoder)로 나누어 시퀀스 변환 문제를 처리함.
- 인코더(**Encoder**): 입력 문장의 모든 단어들을 순차적으로 입력받은 후 모든 단어 정보들을 압축해서 하나의 벡터로 만들어 디코더로 전송함.
- 디코더(**Decoder**): 디코더는 컨텍스트 벡터를 받아서 번역된 단어를 한 개씩 순차적으로 출력함.
- 한계 : 컨텍스트 벡터의 고정적인 크기로 Bottleneck(병목현상)이 일어날 수 있음.

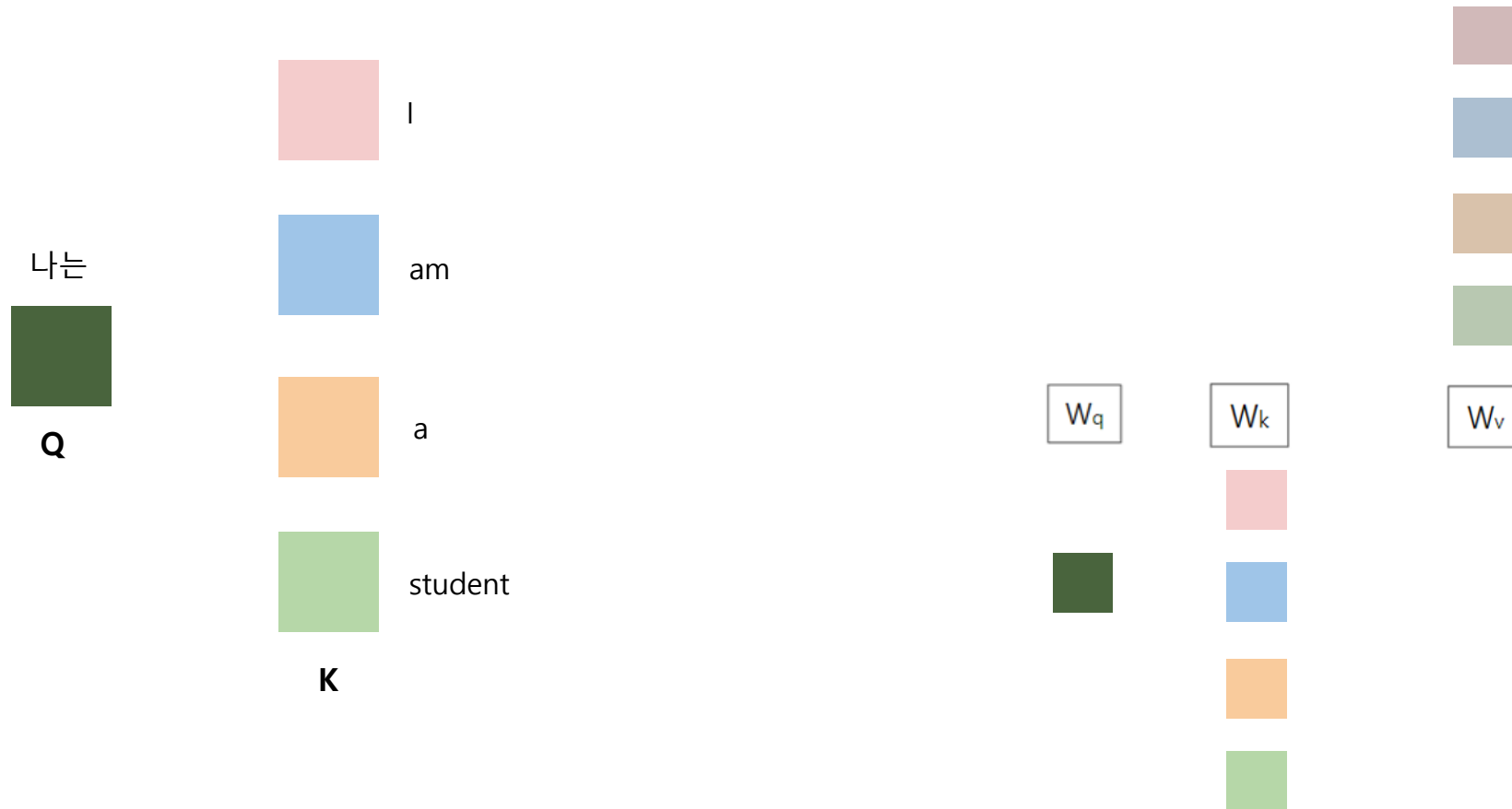
Attention 메커니즘

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

- Query : 특정 단어(토큰)의 정보를 요청하는 역할.
- Key : 각 단어의 특징을 나타냄. 쿼리가 다른 단어와 관련성이 있는지를 판단할 때, 키를 이용함.
- Value: 그 단어의 실제 정보를 담고 있으며, 어텐션 점수에 따라 최종 출력에 반영되는 정보.
- Query와 Key 사이의 유사도를 측정하여 각 입력이 얼마나 중요한지를 계산함.
- 이 유사도는 주로 **내적**으로 계산됨. 즉, Query와 Key의 내적 결과가 높을수록 해당 입력이 더 중요한 것으로 간주됨.

Query, Key, Value

예시 : 나는 학생 입니다



Query, Key, Value

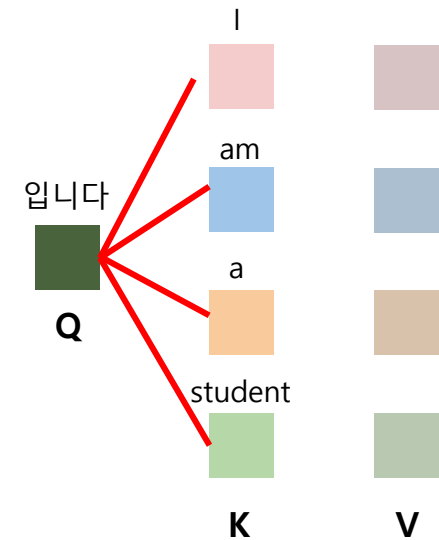
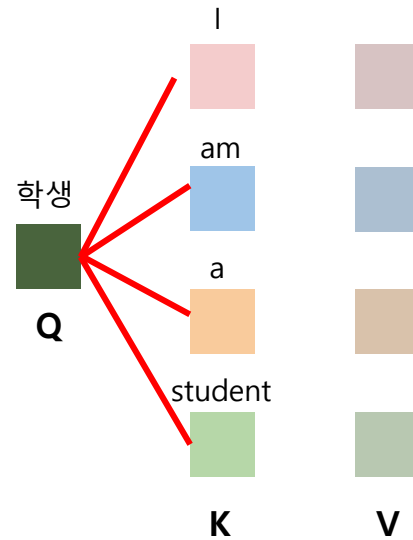
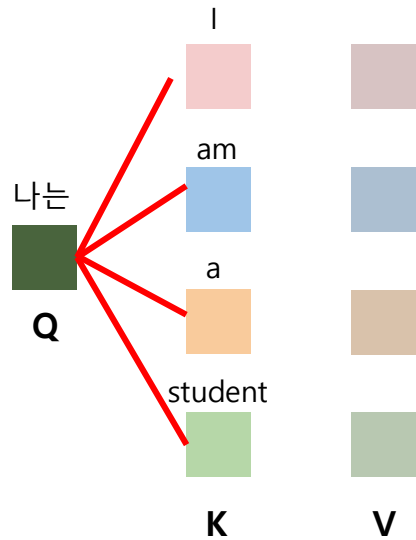
예시 : 나는 학생 입니다



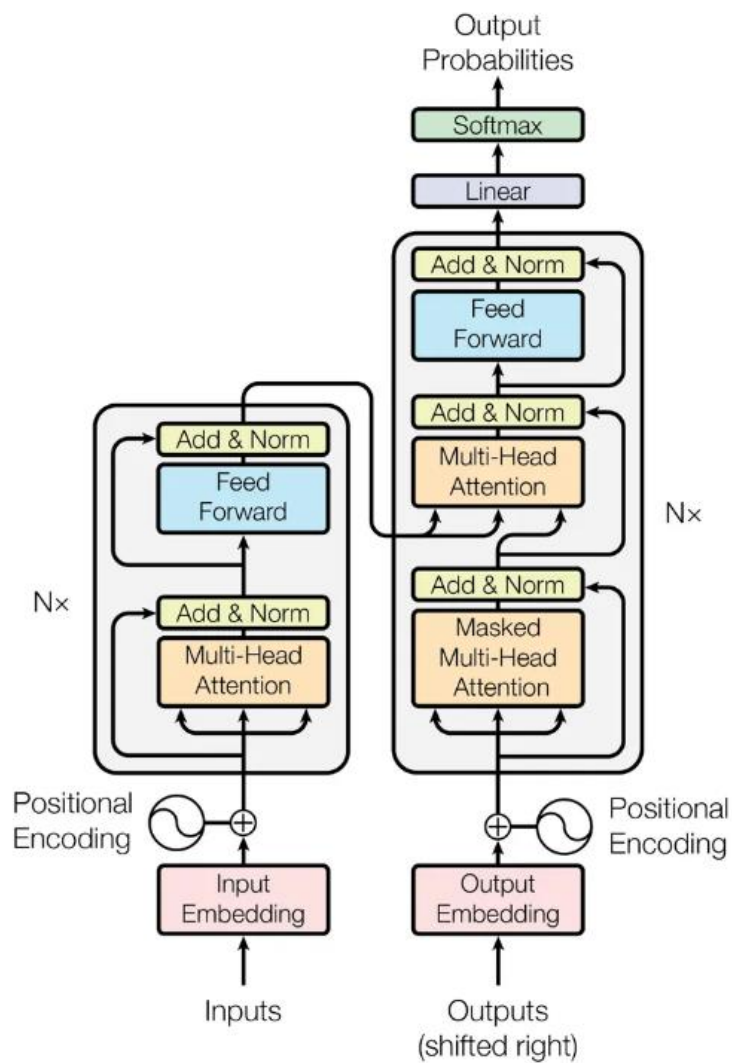
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Query, Key, Value

예시 : 나는 학생 입니다



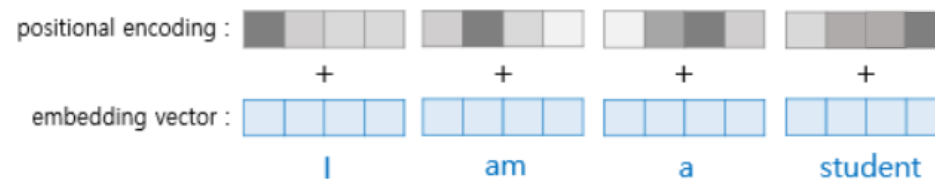
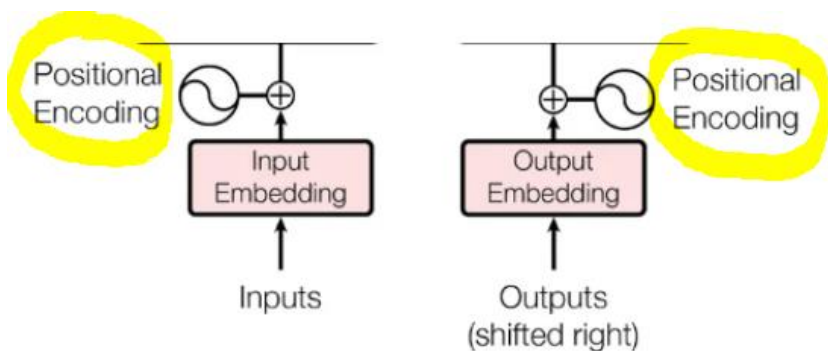
트랜스포머 아키텍처



Transformer 동작과정

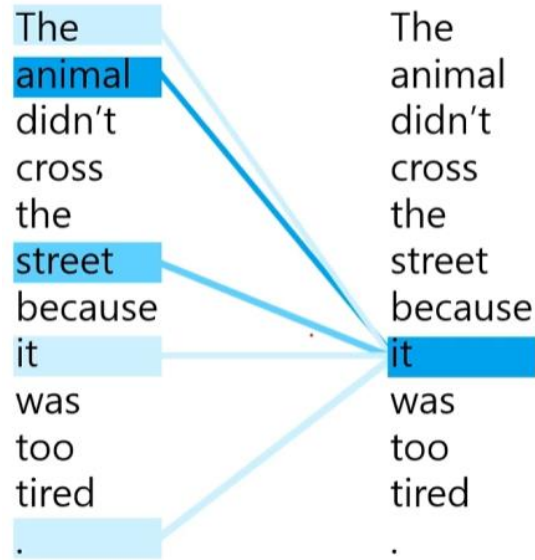
- 입력 시퀀스가 인코더로 들어가면 각 단어에 **포지셔널 인코딩**이 더해져 각 단어의 순서가 고려됨.
- 여러 층의 **Self-Attention**과 **Feed-Forward** 네트워크를 통과하며 변환됨.
- 변환된 인코더의 출력이 디코더로 전달되고, 디코더는 이 정보를 바탕으로 출력 시퀀스를 생성함.
- 디코더에서도 Self-Attention과 Encoder-Decoder Attention 레이어를 거쳐 최종 출력 값을 만들어냄

Positional Encoding

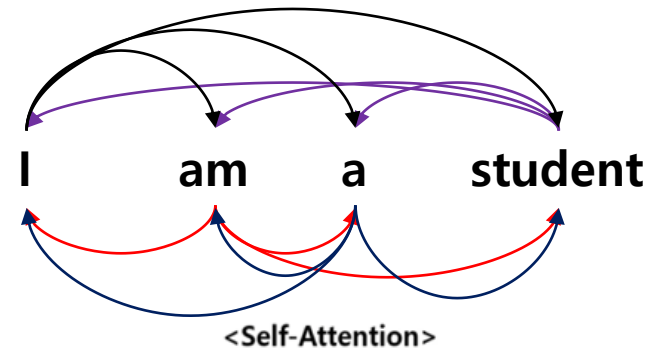
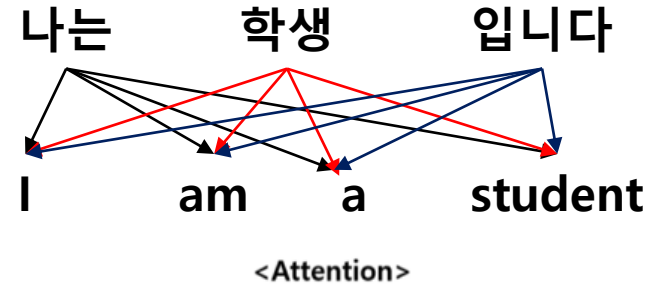


- 입력 시퀀스의 단어 순서를 모델이 이해할 수 있도록 돕는 메커니즘
- **Positional Encoding**을 사용하여 단어의 위치 정보를 입력 임베딩 벡터에 추가함.
- 트랜스포머는 순서 정보를 학습할 수 없기 때문에, **사인/코사인 함수**로 각 단어의 위치를 벡터로 변환함.
- 문장 내 단어들의 **상대적 위치**를 파악하고, 문장의 의미를 보다 잘 이해하도록 도움.

Self-Attention

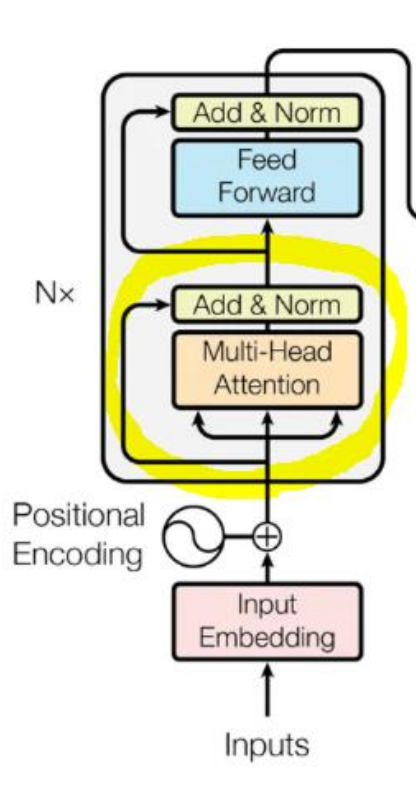


글의 문맥이해가 필요



- 한 문장 안에서 각 단어가 다른 단어와 어떻게 연관되어 있는지 파악함
- 모든 입력 단어를 동시에 참조하기 때문에, 단어 간의 장기 의존성을 잘 처리할 수 있으며 병렬로 계산이 가능하여 빠른 처리 속도를 제공함

Multi-Head Attention



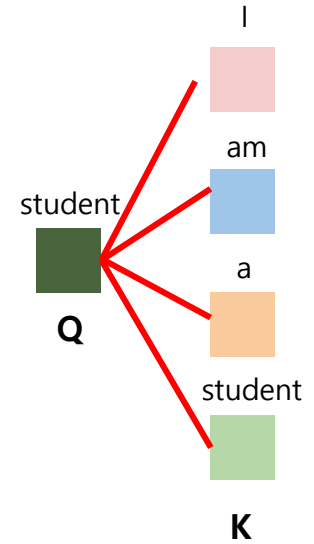
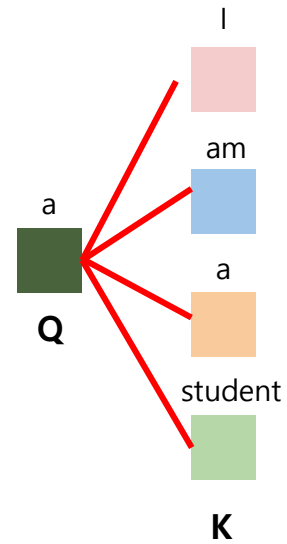
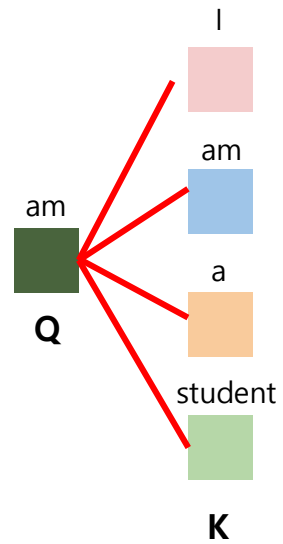
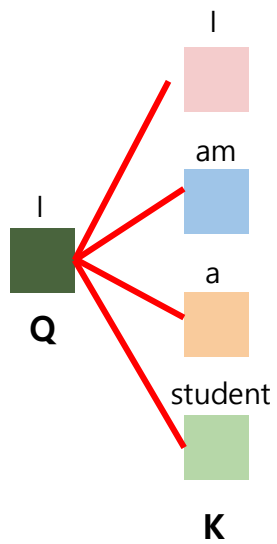
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

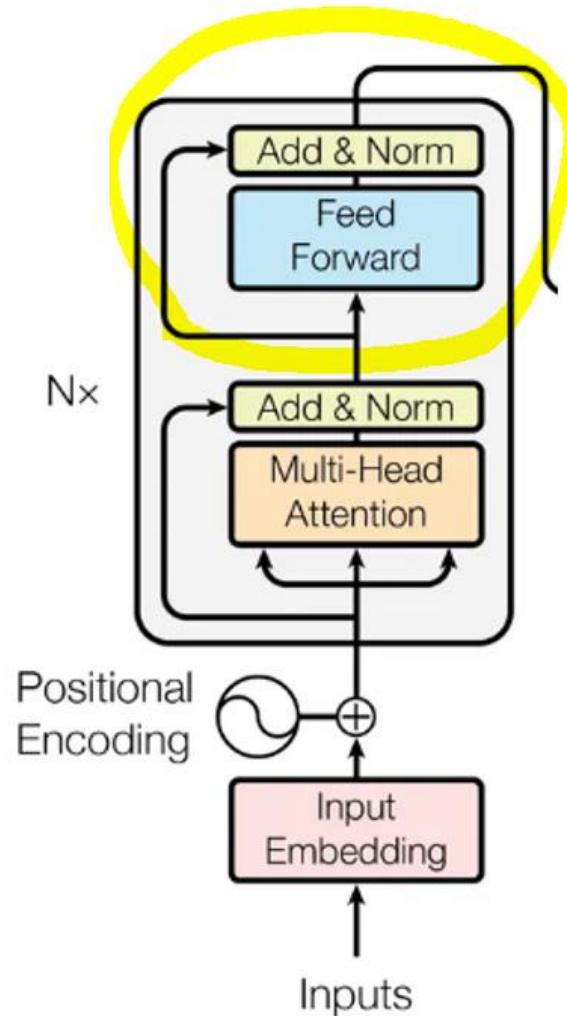
- **Self-Attention** 메커니즘을 병렬로 여러 번 수행하는 구조
- 각 Head는 자기만의 Query, Key, Value에 대해 Self-Attention을 수행하여 중요도를 계산함
- 각 Head에서 나온 Attention 결과는 모두 결합(concatenate)되어 하나의 벡터로 합쳐지고 선형 변환을 거쳐 최종 출력이 만들어짐

Multi-Head Attention

예시 : I am a student



Feed-forward



Feed-forward 네트워크 과정

- 입력: 각 어텐션(attention) 메커니즘 이후에 적용되며, 어텐션 결과를 입력으로 받음.
- 첫 번째 선형 변환 (Fully Connected Layer): 입력 벡터에 대해 고차원 공간에서의 선형 변환이 이루어짐. 입력 벡터를 곱하고 더하는 방식으로 차원을 확장함.
- 비선형 활성화 함수: 주로 ReLU(Rectified Linear Unit) 같은 비선형 활성화 함수가 적용됨.
- 두 번째 선형 변환: 첫 번째 레이어에서 확장된 차원을 다시 원래 차원으로 축소하는 역할을 함.

Masked Multihead Attention

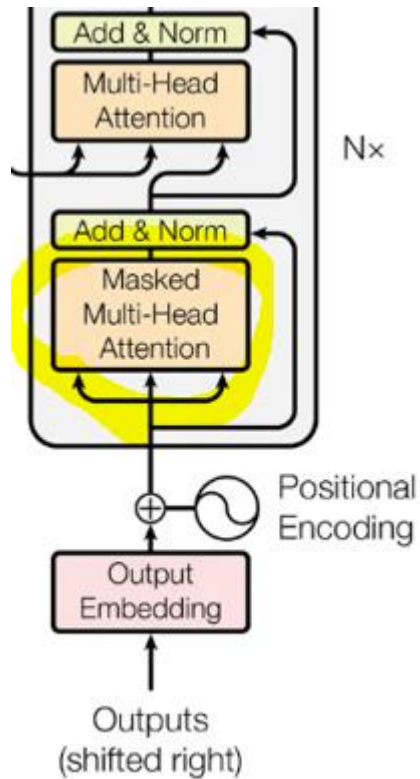


Diagram illustrating the Softmax operation on the attention weights for the word 'student'.

Input Matrix (Attention Weights):

	I	am	a	student
I	10	$-\infty$	$-\infty$	$-\infty$
am	0	10	$-\infty$	$-\infty$
a	-5	-0.1	10	$-\infty$
student	10	-7	-8	10

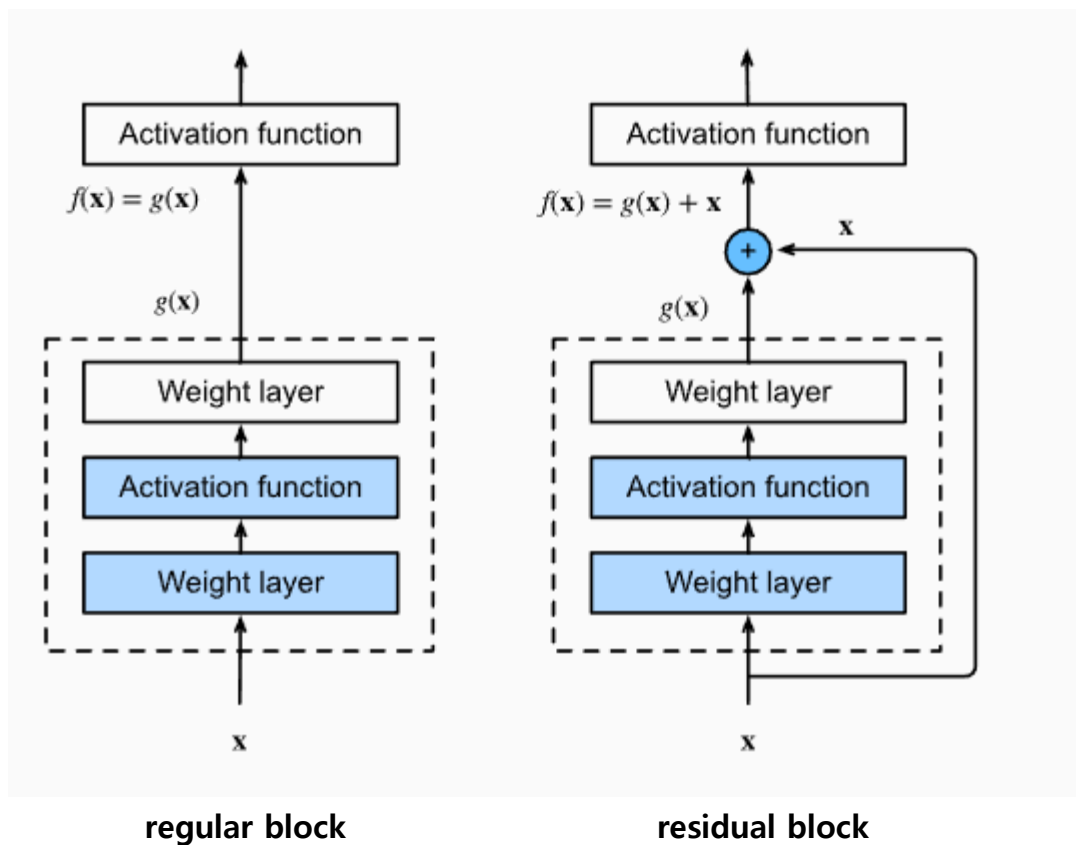
Softmax operation is applied to the input matrix.

Output Matrix (Softmax Results):

	I	am	a	student
I	1.0	0.0	0.0	0.0
am	0.5	1.0	0.0	0.0
a	0.3	0.4	1.0	0.0
student	1.0	0.2	0.1	1.0

- 디코더에서 사용되며, **미래 단어를 참조하지 못하도록** 마스킹을 적용함
- 마스크된 부분은 큰 음수 값을 더해 Softmax 결과가 0에 가까워지도록 함.

Residual Block



구분	Regular Block	Residual Block
구조	단순한 순차적 레이어 (가중치, 활성화 함수 통과)	입력을 레이어에 통과시키고, 입력을 출력에 더하는 잔차 연결 포함
출력 계산식	$f(x) = g(x)$	$f(x) = g(x) + x$
기울기 소실 문제	네트워크가 깊어질수록 기울기 소실 문제 발생 가능	잔차 연결 덕분에 기울기 소실 문제 완화
깊은 네트워크 학습	깊은 네트워크에서 성능 저하 가능성	매우 깊은 네트워크에서도 성능 유지 가능



04

PART FOUR .

결과

결과 – 특정 지역 언급 x

질문을 입력하세요

쇼핑하기 좋은 백화점 추천해줘

초기화

Submit

추천 관광지

1. 롯데백화점 인천점

매일 새로움과 즐거운 경험을 제공하는 롯데백화점 인천점은 유명 해외 명품과 국내외 다양한 패션 및 라이프스타일 매장, 트렌디한 푸드 에비뉴 매장이 입점되어 있다. 인천 지역 최고의 프리미엄 쇼핑 공간에서 즐거운 쇼핑 Life와 재미, 편리함을 모두 즐길 수 있다.

2. 롯데백화점 본점

롯데백화점 본점은 매일 새로움과 즐거운 경험을 제공한다. 롯데백화점 본점의 본관, 에비뉴엘관에는 더욱 다양한 해진 프리미엄 여성, 남성해외패션 브랜드들이 준비 되어있다. 갤러리에 온 듯한, 특별한 미술품들도 매장에서 감상할 수 있다.

3. 롯데백화점 본점 에비뉴엘

롯데백화점 본점과 연결되어 있으며, 해외 패션 브랜드 및 보석 브랜드의 매장만 입점해 있는 고품격 백화점이다. 따로 포토존이 만들어져 있으며 독특한 인테리어로 꾸며져 있다.

9층에는 다양한 테마의 전시회를 감상할 수 있는 갤러리와 레스토랑이 있으며, 10층에는 피부 클리닉 센터와 스파가 있다.

쇼핑을 즐기고 다양한 서비스도 받을 수 있는 신개념의 백화점이다.

4. 더현대 대구

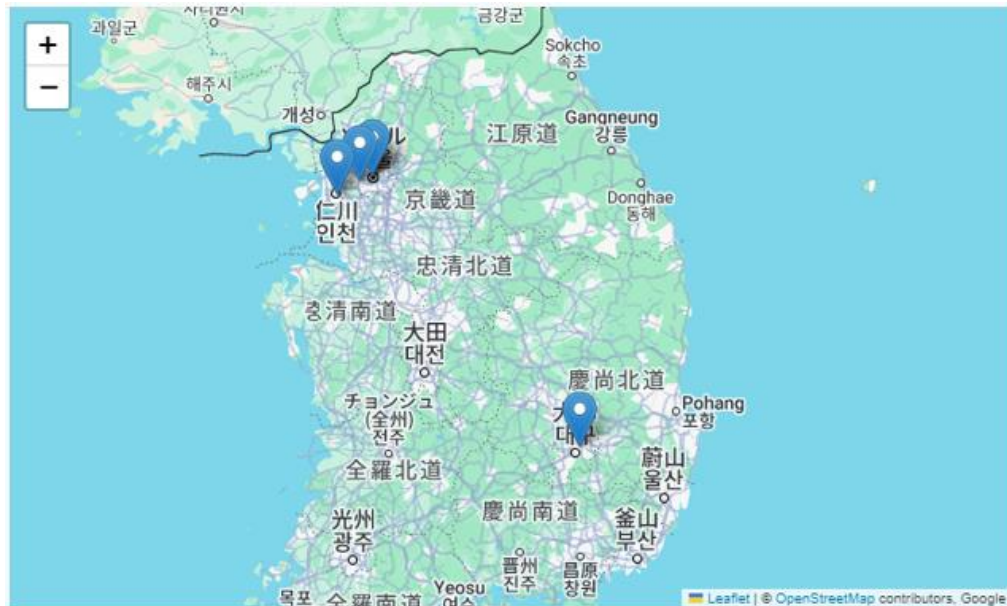
더현대 서울을 이은 더현대 두 번째 시리즈 더현대 대구는 2호선 반월당역에서 5분 거리에 위치해 있다. 이곳에서는 여러 하이엔드 브랜드와 더불어 라이프스타일, 리빙 컬렉션까지 다양한 쇼핑을 즐길 수 있다. 또한, 각종 맛집이 입점해 있어 식사하러 들리기에 편리하다.

5. 행복한백화점

지하 3층, 지상 9층으로 1층 야외에는 행사장이 있고 내부로 들어서면 패션잡화를 판매하고 있으며, 2층에는 여성 의류, 3층에는 남성 패션, 4층에는 시계, 보석, 안경, 가구 등을 파는 상점들과 미용실, 고객센터, 라이브스튜디오가 있으며 5층에는 전문식당가가 자리 잡고 있다.

그 위로는 영화관과 키즈카페, 펫파크 등 다양한 문화시설도 갖추고 있어 고객들에게 편의를 제공하고 있다.

인천광역시 미추홀구에서 열리는 축제: 인천여성영화제 20주년 회고전, 시민창작예술축제 학산마당극놀이



결과 – 특정 지역 언급 x, 축제 x

질문을 입력하세요

맛집 추천해줘

초기화

Submit

추천 관광지

1. 능두네 세종고운점(능두네)

능두네 세종고운점은 세종시 고운동에 자리 잡고 있다. 태국 음식 전문점인 이곳의 대표 메뉴는 꽃게, 바지락, 홍합, 오징어 등이 어우러진 팻퐁커리 탈레다. 이 밖에 풀앰꿍, 소고기쌀국수, 뽕뽕아, 랏나탈레, 팻타이, 팻카파오브라목, 팻퐁파이딩, 팻카파오루쌈 등을 판다. 내부에는 단체석이 마련되어 있어 각종 모임을 하기에 좋다.

2. 카메식당

보령 대천 시내에 위치하고 있는 카메 식당은 낯지만 정겨운 외관과 따뜻한 분위기를 내는 내부 인테리어가 된 곳이다. 카레, 파스타, 크로켓 등 일본식 퓨전요리를 취급하는 곳인데 메뉴 하나하나 정성이 느껴지면서도 맛나다. 한번 방문한 손님들은 다음에 또 와야지 하고 다짐하게 하는 곳이다. 현지 사람들이 외부인과 식사할 때 종종 추천한다고 한다.

3. 비타포

성북동으로 들어서는 가장 초입, 한성대입구역 부근에 위치한 베트남쌀국수집 '비타포'는 소박한 느낌을 주는 맛집이다. 매장내부에 단체석이 있어 단체로 이용이 가능하며 1~2인 좌석도 있어 간단히 혼밥을 하기도 좋은 곳이다. 반찬은 셀프대에서 먹을 만큼 담아 먹을 수 있다. 한성대역 입구에 설치된 무인관광안내 시스템을 이용하면 성북동의 역사 문화를 탐방하는데 도움이 된다.

4. 더마살라

명동근처에 위치한 직장인들에게 유명한 인도음식 맛집이다. 서울특별시 중구에 있는 인도요리전문점이다. 대표 메뉴는 치킨커리이다.

5. 작동홍두깨칼국수

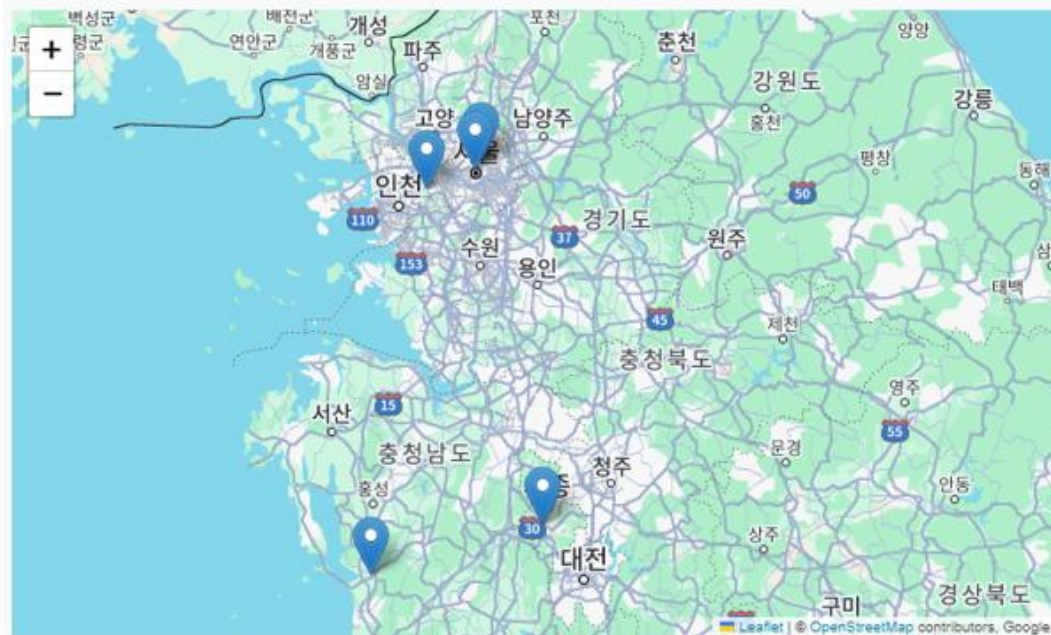
생방송투데이 2591회(20.06.10), 2TV생생정보 738회(19.01.21), 2TV 저녁생생정보 38회(15.12.14), VJ특공대 680회(13.08.30)에 방영된 부천시 지정 부천 맛집이다.

7호선 까치울역 2번 출구, 원미산 아래 자리 잡고 있다.

원미공원과 가까우며 특히 황톳빛 단독 건물은 멀리서도 눈에 띈다.

전용 주차장도 넓다.

메인 메뉴인 칼국수는 면을 직접 뽑아낼 뿐만 아니라 새우, 오징어, 미더덕 등 해물로 우려낸 육수 맛이 일품이다.



결과 – 특정 지역 언급

질문을 입력하세요

수원에 있는 카페 추천해줘

초기화

Submit

추천 관광지

1. 웨일

수원시 영통구 법조로에 있는 카페로 상현역 2번 출구에서 도보 3분 정도로 가깝고 카페 내부는 깔끔하고 곳곳에 식물을 놓아 내추럴한 분위기를 연출했고 천장 층고가 높아 개방감이 느껴진다. 바닥은 에폭시 코팅으로 넓어 보이며 테이블과 의자는 나무로 되어있다. 또한 매장의 앞과 뒤쪽 모두 통창으로 되어있어 개방감이 느껴진다. 또한 매장 내에 다양한 디저트들과 커피, 음료가 준비되어있다.

2. 와이엠엠디[YMMD:You made my day]

바로 앞에는 탄천이 있어 산책하다가 들르기 좋은 장소다. 입구에는 나무로 된 좌석이 있고 출입구와 창문 모두 따뜻한 느낌의 나무 소재로 되어 있다. 벽에 빔을 쏘아 애니메이션을 틀어주기도 한다. 반려견을 동반할 수 있으나 펫 캐리어나 품속 동반만 가능하며 펫티켓을 준수해야 한다. 데이트하기 좋은 잔잔한 분위기의 감성 카페다.

3. 콜링우드

콜링우드는 수원 화성 장안문 근처에 위치한 카페이다. 외관이 검은색 외벽으로 되어 있고 입구 앞에 화분으로 꾸며진 입구로 들어가면 내부는 층고가 높고 곳곳에 대형 화분이 놓여 있으며 테이블이 여유롭게 배치되어 있다. 이곳은 아메리카노, 바닐라빈 라떼, 라떼 등의 커피 음료와 크림 초코, 히비스커스 베리티 등의 음료를 맛볼 수 있다.

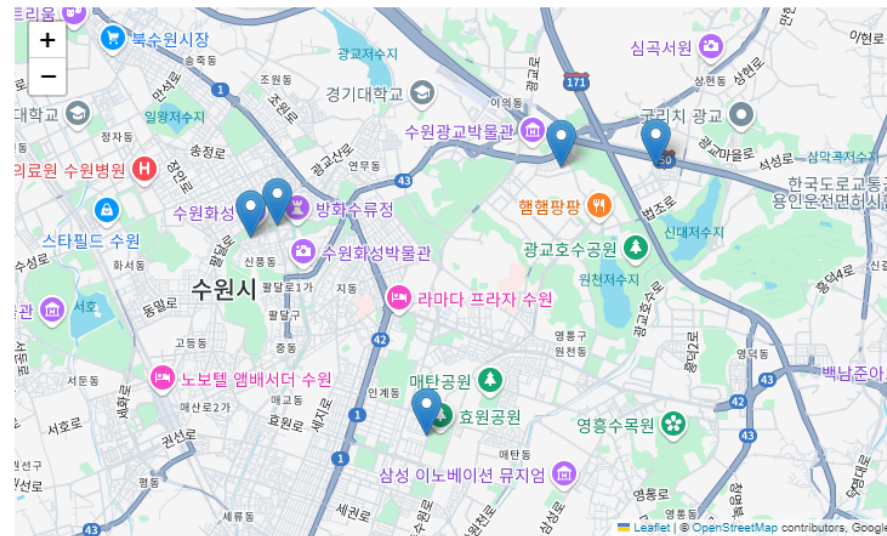
4. 킷넷 본점

킷넷 본점은 수원화성 화서문 근처에 있는 카페이다. 오래된 주택건물을 개조하여 만든 카페로 입구와 욕장에 야외 테이블이 골고루 배치되어 있고 회색 배경의 내부에는 우드톤 의자와 테이블이 깔끔하게 배치되어 있다. 킷넷 본점은 반려동물들을 동반할 수 있는 카페로 실내는 캐리어와 이동 가방이 있어야 이용할 수 있고 실외는 편하게 이용할 수 있다.

5. 카페 도안

각 원두에 맞는 추출 도구로 추출한 브루잉 커피와 원두를 고를 수 있는 아메리카노, 카페라떼 등의 커피 메뉴와 라떼, 에이드, 허브티 등의 다양한 음료가 있고, 복숭아 롤케이크, 얼그레이 파운드, 숙인절미 파운드 등의 디저트 메뉴를 맛볼 수 있다. 주변에 경기아트센터와 인계예술공원이 있고 수인분당선 수원시청역이 있어 대중교통으로도 이용할 수 있다.

경기도 수원시에서 열리는 축제: 수원재즈페스티벌, 리얼 트레저 페스티벌, 경기청년페스티벌, 경기미디어트페스타, 경기민족예술제



시연 영상1

필드를 입력하세요

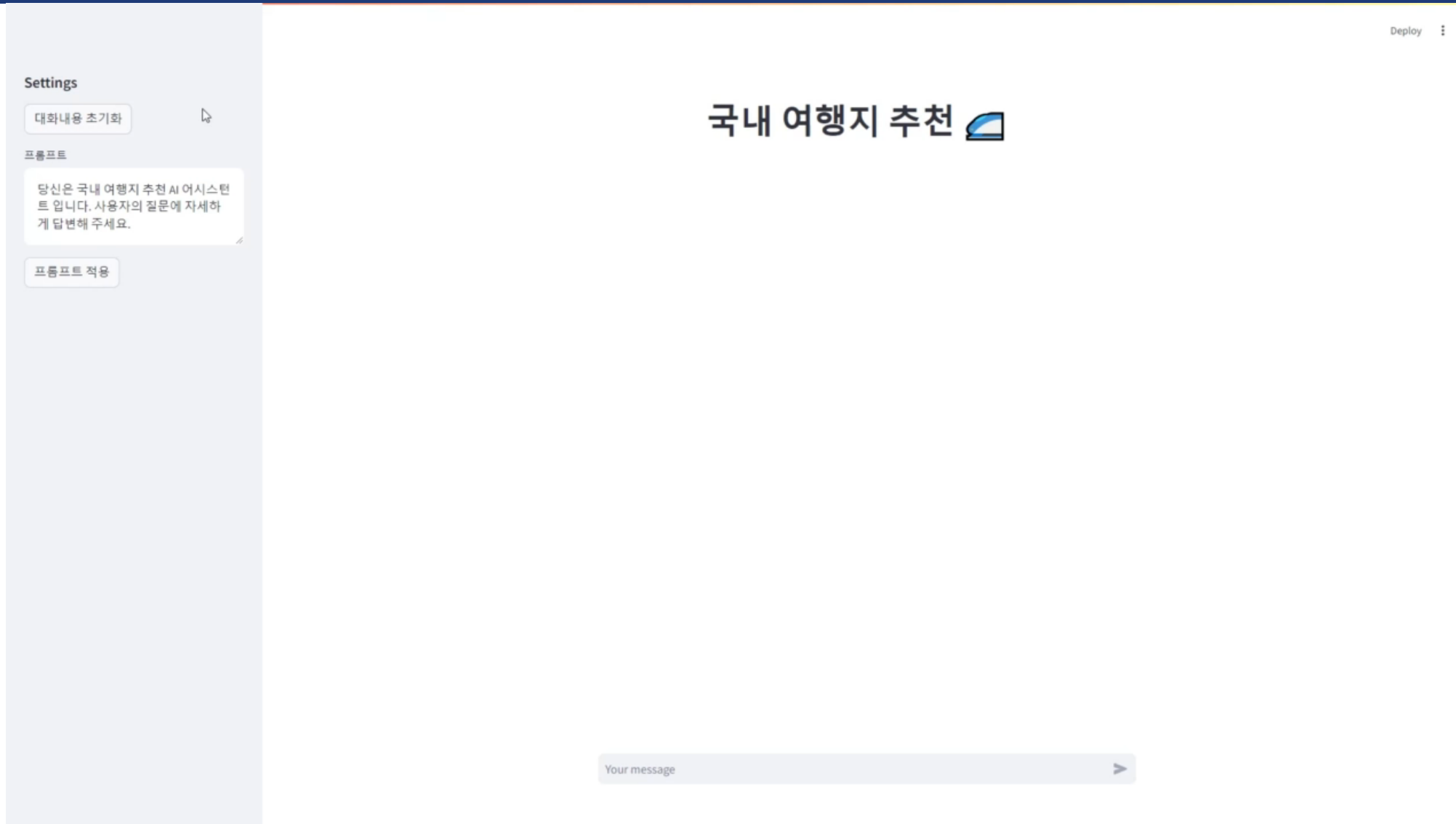
관람지 관련 정보를 입력해주세요

초기화

Submit

추천 관광지

시연 영상2



결과 – GPT모델과 성능비교

질문을 입력하세요

부산 호텔 추천해줘

초기화

Submit

추천 관광지

1. 힐 사이드 관광호텔

언덕 위의 작은 별장 힐사이드 호텔은 부산의 상징인 오륙도와 영도가 한 눈에 내려다 보이는 영주동 언덕 위에 작은 별장 같은 호텔이다. 차분하고 안락한 스타일의 객실로 편안함을 제공하고 있다. 부산의 주요관광지와의 접근성이 좋아 가족, 연인, 친구와 함께 방문하기 좋은 호텔이다.

2. 크라운 하버 호텔 부산

부산 중구 중앙동에 위치한 크라운하버 호텔 부산은 비즈니스 편의는 물론 특색있는 원도심 관광까지 즐길 수 있는 최적의 호텔이다. 8가지 타입의 다양한 객실로 구성되어 다양한 니즈에 맞춰 이용할 수 있고, 합리적인 가격으로 세심하고 온기있는 서비스를 제공해 가성비가 좋다. 부산여객선터미널, 부산역, 부산타워, 국제시장, 남포동, BIFF 광장을 도보로 이용할 수 있어 편리하게 여행할 수 있다.

3. 지엔비호텔(GNB호텔)[한국관광 품질인증/Korea Quality]

지엔비호텔은 부산지하철 자갈치역에서 도보 10분 거리에 있는 대형 호텔로, 스탠다드부터 디럭스, 스위트, 펜트하우스까지 다양한 타입의 객실을 제공한다. 특히 호텔의 자랑인 펜트하우스와 펜트하우스 플러스 객실에는 월풀스파가 있어 부산 도심 풍경을 바라보며 여행의 피로를 씻을 수 있고, 인덕션과 취사용품도 있어 요리도 할 수 있다. 조식은 지하 1층 레스토랑에서 제공하고, 비즈니스 출장객을 위해 1층 비즈니스센터에서 프린트, 스캔 서비스도 제공한다.

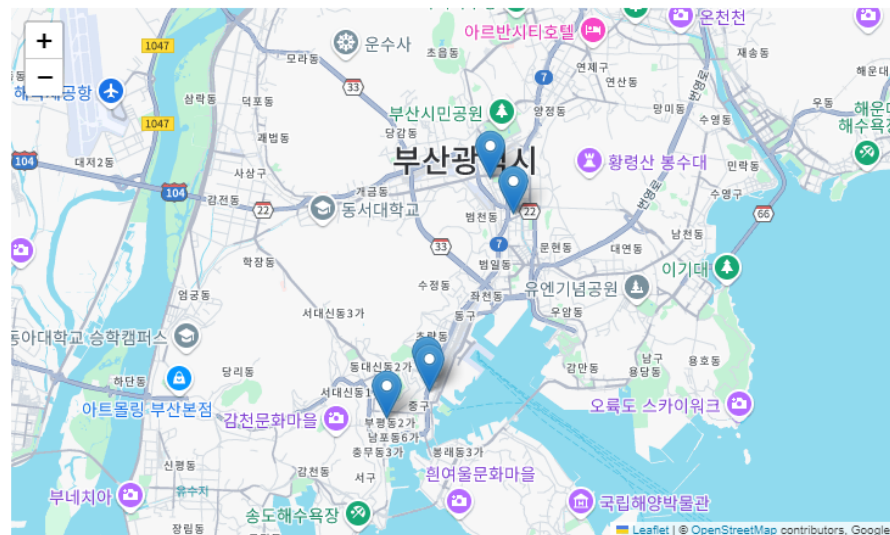
4. 브라운도트호텔 서면범천점

국제금융센터가 인근에 위치하고 있어 부산을 방문하는 금융인, 관광객 등 방문객이 부산 중심인 호텔인 브라운도트호텔 서면범천점에서 편안하고 여유로운 휴식을 취할 수 있다. 브라운도트호텔 서면범천점에서는 매일 지배인 주최 무료 리셉션, 시설 내 쇼핑몰, 드라이클리닝/세탁 서비스도 이용할 수 있으며, 객실 내 무료 WiFi에 연결해 최신 정보를 확인하고, 레스토랑, 짐 보관, 세탁기, 건조기 등 편의 시설과 서비스를 이용할 수 있다.

5. 브라운도트 호텔 서면역점 비즈니스

브라운도트 호텔 서면역점 비즈니스는 서면역 도보 3분 거리에 있는 비즈니스 호텔로, 넓은 객실, 화장실과 욕실을 분리한 디자인, 고급 린넨사양을 적용한 안락한 베딩이 인상적이다. 합리적 비용으로 서면을 중심으로 한 부산 여행시 가성비 좋게 이용이 가능하고, 롯데백화점, 젊음의 거리, 서면맛집 등 핫플레이스와 접근성이 좋아 편리하게 여행을 즐길 수 있다.

부산광역시 중구에서 열리는 축제: 유라리 건맥축제, 중구 부산항 펀&핑 페스티벌, 피란수도 부산 문화유산 야행, 용골 댄스 페스타, 부산자갈치축제



결과 – GPT모델과 성능비교

Settings

대화내용 초기화

프롬프트

당신은 국내 여행지 추천 AI 어시스턴트입니다. 사용자의 질문에 자세하게 답변해 주세요.

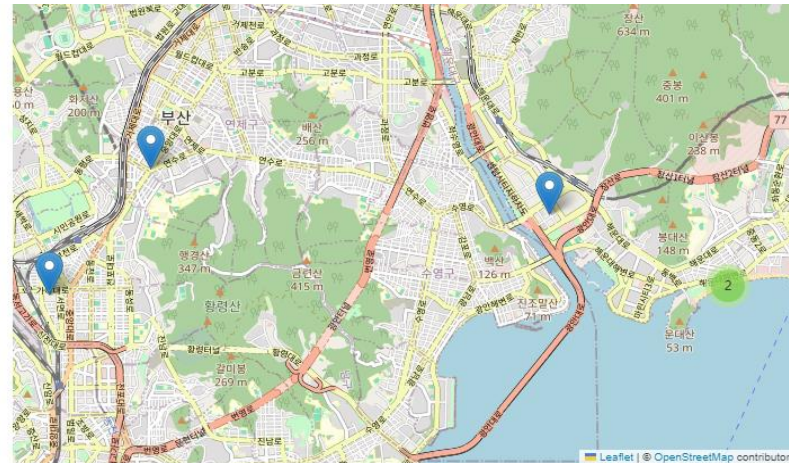
프롬프트 적용

국내 여행지 추천

부산 호텔 5개 추천해줘



1. 파라다이스 호텔 부산 - 해운대 해수욕장과 가깝고 탁트인 바다 전망을 즐길 수 있는 호텔입니다. 2. 롯데호텔 부산 - 부산의 중심지인 서면에 위치해 있으며 편안한 객실과 다양한 시설을 갖춘 호텔입니다. 3. 그랜드 하얏트 부산 - 해운대 해변에 위치한 럭셔리 호텔로 탁트인 바다 전망과 고급스러운 서비스를 경험할 수 있습니다. 4. 신라스테이 해운대 - 해운대 해변과 가까이 위치한 모던한 디자인의 호텔로 편안한 숙박을 즐길 수 있습니다. 5. 노보텔 부산 엠베서더 도시 - 부산역과 가까이 위치한 깔끔한 디자인의 호텔로 교통편이 편리하고 편안한 휴식을 취할 수 있습니다.



Your message



TIME TABLE

구분	9월																				10월						
	12 일	13 일	14 일	15 일	16 일	17 일	18 일	19 일	20 일	21 일	22 일	23 일	24 일	25 일	26 일	27 일	28 일	29 일	30 일	1일	2일	3일	4일	5일	6일	7일	
□ 주제 선정																											
□ 주제 구체화 및 데이터셋 분석																											
□ 데이터 전처리																											
□ 초안 코드 작성																											
□ 성능 테스트 및 개선																											
□ 1차 발표																											
□ 기획 수정																											
□ 데이터 추가준비																											
□ 챗봇 구현																											
□ 중간발표																											
□ 데이터셋 변경																											
□ 메인 코드 수정																											
□ 최종 발표																											

The background features a large white circle on the left containing the text 'THANK YOU Q&A'. This circle is surrounded by several smaller circles: two solid dark blue ones and several white ones with thin grey outlines. Some of these smaller circles are partially overlapping the large circle. On the right side of the slide, there are more white circles with thin grey outlines and one solid dark blue circle.

**THANK YOU
Q&A**