[2024 TA Final Report]

A Restaurant Recommendation System Using TA

– A Restaurant Recommendation System Utilizing Naver Review Data –

TEAM 7

담당교수	Bernardo Nugroho Yahya
제출일	2024년 12월 19일
학번&이름	202000318 Kwon SoonJae 202001712 Seo MinSeok 202002439 Lee SangMin 202102670 Lee JungSeok

Table of Contents

Chapter 1: Project Summary

Section 1: Project Changes

1-1. Change in Project Topic

The original project topic was **"Chatbot-Based Restaurant Recommendation System Using Review Data"**, with the objective of designing a system where users could receive restaurant recommendations through conversations with a chatbot. However, after consulting with the professor, we realized that our current skill set was insufficient to implement a chatbot effectively.

In particular, maintaining a natural conversational flow while accurately addressing the diverse requirements of users posed significant challenges.

As a result, the integration of a chatbot-based system was deemed overly ambitious, and the topic was refined to **"Restaurant Recommendation System Using TA."** This revised focus excludes the conversational system and concentrates on implementing text analysis and recommendation algorithms using review data.

1-2. Revision of Review Dataset Summary

During discussions with the professor, issues with the original review dataset were identified. The original approach involved calculating the similarity between the user input and over 200 individual reviews per restaurant. However, this approach risked generating recommendation results that were overly biased toward specific reviews.

For instance, if a single review matched the user input very closely, the system might recommend that restaurant regardless of the content of other reviews. In other words, it failed to take all reviews into consideration adequately. This raised concerns about the reliability and accuracy of the recommendation system.

To address this issue, the dataset was restructured to be more comprehensive and balanced. The revised method involved the following steps:

- Grouping all reviews for each restaurant into a single text.
- Splitting the grouped text into smaller segments based on a token limit (1024 tokens).
- Summarizing these segments repeatedly until only one summarized review remained per restaurant.

This process resulted in a single summarized review for each restaurant, which was then used for analysis. By adopting this approach, the recommendation system could generate results that were less biased and more balanced across the dataset.

This dataset summarization and restructuring contributed to enhancing the system's comprehensiveness and reliability, enabling it to provide more balanced evaluations and deliver highly trustworthy recommendation results.

Section 2: Project Summary

Our project aims to implement a system that recommends restaurants based on user-input conditions using review data. To achieve this, we utilized the KoBART model to summarize review data and designed two recommendation models.

The first step involves summarizing review data. Reviews for each restaurant are grouped and summarized using the KoBART model. The reviews are segmented into tokens (512–1024 tokens per segment) and summarized, resulting in comprehensive and balanced data. This summarized data is used as training data for the recommendation algorithm.

The recommendation system consists of two models. The first model focuses on the similarity between user input and review data to recommend the most suitable restaurants. This model features a simple structure, operates quickly, and effectively reflects user needs. The second model analyzes detailed information from user input, such as wait time, restaurant category, companion information, and location. It assigns weights to each element to generate refined recommendation results.

This system is designed to combine text summarization with precise data analysis, addressing diverse user needs and providing personalized recommendation results.

.

Chapter 2: Project Outline

Section 1: Project Topic

In modern society, users often struggle to find restaurants that match their preferences among the overwhelming amount of restaurant information and reviews available. Instead of relying solely on ratings or scores, this project aims to develop a more sophisticated and personalized restaurant recommendation system that reflects users' specific needs.

When searching for a restaurant suitable for occasions like family gatherings or special celebrations, people typically rely on limited information they already know or randomly search the internet. However, this approach can be time-consuming and may lead to unsatisfactory recommendations.

The restaurant recommendation system we aim to develop uses review data to analyze users' specific requirements and recommend the most suitable restaurants, thereby enhancing user satisfaction. This system extracts keywords that match users' needs from review data and provides refined recommendations based on those keywords. As a result, users will be able to easily find restaurants that align with their desired atmosphere or specific criteria.

Section 2: Project Objectives

Based on restaurant review data and NLP technologies, our goal is to develop a restaurant recommendation system that analyzes user requirements and recommends the optimal restaurant. By analyzing review data, the system aims to accurately understand user needs and enhance the precision and convenience of restaurant recommendations.

Overall, the purpose of this restaurant recommendation system is to comprehend users' specific requirements and provide personalized recommendations to improve user satisfaction. Additionally, it seeks to contribute to saving time and reducing stress during the restaurant selection process by ensuring an efficient recommendation process.

Section 3: Similar Systems

- **Yelp Recommendation System:** Various recommendation systems have been developed using the Yelp dataset to provide the most suitable restaurants based on user preferences and reviews. Most of these systems leverage NLP technologies, such as sentiment analysis and keyword extraction, to enhance accuracy.

- **Conversational Agents for Recommendations:** Chatbots like Replika and Google's Dialogflow have been used to create similar conversational agents that help users find relevant services based on their preferences.

- **Restaurant Review Summarizers:** Some projects focus on developing systems that summarize restaurant reviews to assist users in making better decisions.

Chapter 3: Project Progress

Section 1: Project Definition

1-1. Functional Definitions

Function	Description
Review Data Preprocessing	Removes special characters, duplicates, repetitive patterns, and performs quality improvement tasks on review data.
Review Data Summarization	Processes user input in real-time conversations and analyzes requirements.
Recommendation Model 1: Similarity-Based Recommendation	Develops and trains a model to recommend suitable restaurants to users based on the similarity between review data and user requirements.
Recommendation Model 2: Detail-Based Recommendation	Develops and trains a model to recommend suitable restaurants to users based on detailed information, including various parameters.
Result Output	Outputs a list of recommended restaurants, including similarity scores, visit rates, wait times, and other detailed information.

1-2. Technologies and Methods Required for Project Execution

KoBART

- KoBART is a Korean-language-specific Transformer-based BART model developed by Kakao Brain, optimized for Korean natural language generation and understanding tasks. BART is based on the autoencoder approach, a common language model training method, and leverages text reconstruction ability to perform well in various natural language processing tasks such as sentence generation, summarization, translation, and question answering. KoBART, built on this BART architecture, is trained on Korean data to better understand and utilize Korean grammar and meaning. This design is particularly strong in handling the complex grammatical structure and lexical characteristics of the Korean language and is widely used in various Korean NLP applications.

Sentence-BERT

- Sentence-BERT (SBERT) is a natural language processing model that effectively generates sentence embeddings, performing well in restaurant recommendation systems by comparing user input with review data. SBERT accurately reflects the context and meaning of text, using cosine similarity to provide refined recommendations tailored to user preferences. The model's exceptional ability to understand context makes it ideally suited for personalized restaurant recommendation systems.

Cosine Similarity

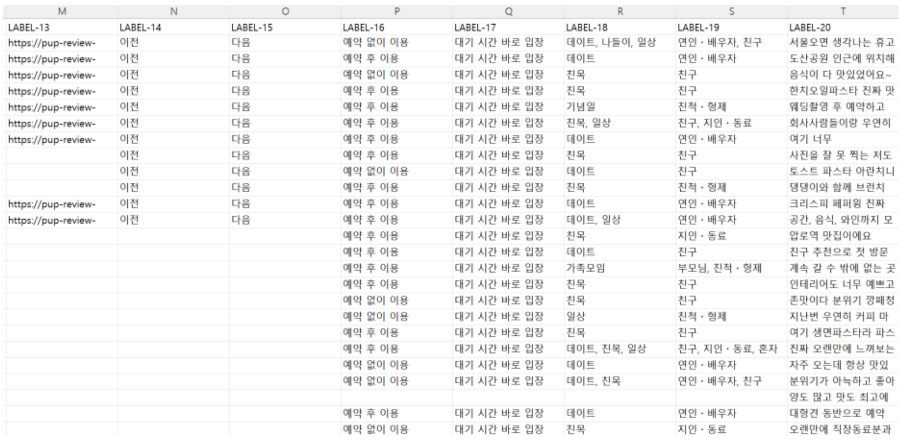
- Cosine Similarity is a mathematical method used to measure the similarity between two vectors by calculating their similarity based on the direction of the vectors. This method helps to precisely compare the similarity between two texts after converting the text data into numerical values. Particularly, cosine similarity is not affected by the length of the text, making it suitable for comparing the meaning between reviews and user input, and providing the most appropriate recommendations.

Section 2: Data Collection

In this project, we will explain the process of collecting and processing data to design a restaurant recommendation system based on review data.

1. Data Collection

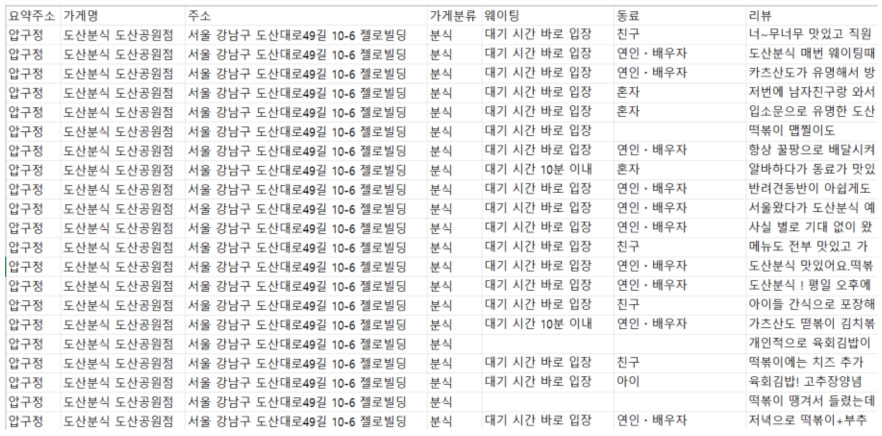
The review data required for this project was collected from Naver Maps using the Chrome extension Listly. Listly is a tool that allows users to efficiently extract specified information from web pages.



M	N	O	P	Q	R	S	T
LABEL-13	LABEL-14	LABEL-15	LABEL-16	LABEL-17	LABEL-18	LABEL-19	LABEL-20
https://pup-review-	이전	다음	예약 없이 이용	대기 시간 바로 입장	데이트, 나들이, 일상	연인・배우자, 친구	서울오면 생각나는 휴고
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	데이트	연인・배우자	도산공원 인근에 위치해
https://pup-review-	이전	다음	예약 없이 이용	대기 시간 바로 입장	친목	친구	음식이 다 맛있었어요~
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	친목	친구	한치오일파스타 진짜 맛
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	기념일	친척・형제	웨딩촬영 후 예약하고
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	친목, 일상	친구, 지인・동료	회사사람들이랑 우연히
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	데이트	연인・배우자	여기 너무
	이전	다음	예약 후 이용	대기 시간 바로 입장	친목	친구	사진을 잘 못 찍는 저도
	이전	다음	예약 없이 이용	대기 시간 바로 입장	데이트	친구	토스트 파스타 아란치니
	이전	다음	예약 후 이용	대기 시간 바로 입장	친목	친척・형제	댕댕이와 함께 브런치
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	데이트	연인・배우자	크리스피 페퍼윙 진짜
https://pup-review-	이전	다음	예약 후 이용	대기 시간 바로 입장	데이트, 일상	연인・배우자	공간, 음식, 와인까지 모
			예약 후 이용	대기 시간 바로 입장	친목	지인・동료	압로역 맛집이에요
			예약 후 이용	대기 시간 바로 입장	데이트	친구	친구 추천으로 첫 방문
			예약 후 이용	대기 시간 바로 입장	가족모임	부모님, 친척・형제	계속 갈 수 밖에 없는 곳
			예약 후 이용	대기 시간 바로 입장	친목	친구	인테리어도 너무 예쁘고
			예약 없이 이용	대기 시간 바로 입장	친목	친구	존맛이다 분위기 깡패청
			예약 없이 이용	대기 시간 바로 입장	일상	친척・형제	지난번 우연히 커피 마
			예약 후 이용	대기 시간 바로 입장	친목	친구	여기 생면파스타라 파스
			예약 후 이용	대기 시간 바로 입장	데이트, 친목, 일상	친구, 지인・동료, 혼자	진짜 오랜만에 느껴보는
			예약 없이 이용	대기 시간 바로 입장	데이트	연인・배우자	자주 오는데 항상 맛있
			예약 없이 이용	대기 시간 바로 입장	데이트, 친목	연인・배우자, 친구	분위기가 아늑하고 좋아
							양도 많고 맛도 최고에
			예약 후 이용	대기 시간 바로 입장	데이트	연인・배우자	대형견 동반으로 예약
			예약 없이 이용	대기 시간 바로 입장	친목	지인・동료	오랜만에 직장동료분과

Collected Data

The data collected using Listly was not organized in the format shown above. We have cleaned and organized the data by retaining only the relevant columns for use, such as wait time, companion type, review content, restaurant name, address, and category.



요약주소	가게명	주소	가게분류	웨이팅	동료	리뷰
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	친구	너~무너무 맛있고 직원
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	도산분식 매번 웨이팅때
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	카츠산도가 유명해서 방
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	혼자	저번에 남자친구랑 와서
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	혼자	입소문으로 유명한 도산
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장		떡볶이 맵찔이도
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	항상 꿀팡으로 배달시켜
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 10분 이내	혼자	알바하다가 동료가 맛있
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	반려견동반이 아쉽게도
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	서울왔다가 도산분식 예
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	사실 별로 기대 없이 왔
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	친구	메뉴도 전부 맛있고 가
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	도산분식 맛있어요.떡볶
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	도산분식 ! 평일 오후에
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	친구	아이들 간식으로 포장해
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 10분 이내	연인・배우자	가츠산도 떡볶이 김치볶
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식			개인적으로 육회김밥이
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	친구	떡볶이에는 치즈 추가
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	아이	육회김밥! 고추장양념
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식			떡볶이 땡겨서 들렀는데
압구정	도산분식 도산공원점	서울 강남구 도산대로49길 10-6 젤로빌딩	분식	대기 시간 바로 입장	연인・배우자	저녁으로 떡볶이+부추

Organized Data

This approach allowed us to quickly and systematically gather large-scale data.

2. Data Preprocessing

```
def preprocess_reviews(reviews):

    processed_reviews = []
    for review in reviews:
        review = re.sub(r'http[s]?://\S+', '', str(review))   # https 또는 http 링크 제거
        review = re.sub(r'[^\w\s ㄱ-ㅎ ㅏ-ㅣ 가-힣]', '', review)   # 특수문자 제거
        review = re.sub(r'\s+', ' ', review.strip())   # 공백 정리
        processed_reviews.append(review)
    return list(set(processed_reviews))
```

We performed preprocessing on the review data by removing special characters, cleaning up spaces, and removing https links, and then returned the processed results.

Section 3: Data Summary

We perform the task of summarizing review data for each restaurant using the KoBART model. This process includes data preprocessing, summary generation, and saving the results to an Excel file. Below, we will explain the code step by step.

- Load KoBART Model and Tokenizer

```
from transformers import BartForConditionalGeneration, PreTrainedTokenizerFast

# KoBART 모델 및 토크나이저 로드
model = BartForConditionalGeneration.from_pretrained('gogamza/kobart-base-v2')
tokenizer = PreTrainedTokenizerFast.from_pretrained('gogamza/kobart-base-v2',
                                                    bos_token='</s>', eos_token='</s>',
                                                    pad_token='<pad>')
```

We initialize the natural language processing model using the KoBART model and PreTrainedTokenizerFast. **gogamza/kobart-base-v2** is a BART-based model optimized for Korean data.

- Function to Remove Repeated Word

```python
def clean_text(text):
    sentences = re.split(r'[.!?]', text)
    sentences = [s.strip() for s in sentences if s.strip()]
    sentence_counts = Counter(sentences)
    unique_sentences = [sentence for sentence in sentences if sentence_counts[sentence] == 1]
    text = ". ".join(unique_sentences) + "."

    words = text.split()
    filtered_words = []
    last_word = None
    for word in words:
        if word != last_word:
            filtered_words.append(word)
            last_word = word
    text = " ".join(filtered_words)

    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

This function removes duplicated sentences and consecutively repeated words from the given text and cleans up spaces to return a neat version of the text. The main goal of the text cleaning process is to generate concise and non-redundant data.

- Review Grouping

```python
def group_reviews_by_tokens(reviews, tokenizer, max_tokens=512):
    grouped_reviews = []
    current_group = []
    current_token_count = 0

    for review in reviews:
        token_count = len(tokenizer.encode(review, truncation=False))

        if current_token_count + token_count > max_tokens:
            if current_group:
                grouped_reviews.append(" ".join(current_group))
            current_group = [review]
            current_token_count = token_count
        else:
            current_group.append(review)
            current_token_count += token_count

    if current_group:
        grouped_reviews.append(" ".join(current_group))

    return grouped_reviews
```

The review data was grouped to avoid exceeding the maximum token length of the KoBART model. Although the maximum token length for KoBART is 1024, the grouping was limited to 512 tokens to prevent the focus of the summary from becoming blurred and to improve performance. To preserve the context of the reviews, each review was ensured not to be split across different groups.

- Summary Generation

```
def summarize_groups(groups, model, tokenizer, max_input_length=512, max_output_length=200):
    summaries = []
    for group in groups:
        if not group.strip():
            continue

        try:
            group = preprocess_text(group)
            input_ids = tokenizer.encode(group, return_tensors="pt", max_length=max_input_length, truncation=True)
            summary_ids = model.generate(input_ids, max_length=max_output_length, num_beams=6, early_stopping=True)
            summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
            summary = clean_text(summary)
            summaries.append(summary)
        except Exception as e:
            print(f"Error summarizing group: {e}")

    return summaries
```

Each review group is summarized using the KoBART model, and beam search (num_beams=6) is employed to select the most suitable summary. Beam search is an algorithm that considers multiple candidates simultaneously during text generation to create the optimal sentence. Additionally, the maximum token length for the summary sentence is limited to 200 to produce a more concise and effective summary.

Since unexpected, repeated sentences or unnecessary redundancy can occur during the summarization process, the clean_text function is used after the summary is generated to further refine the results by removing duplicated sentences or unnecessary spaces, ultimately providing a concise summary.

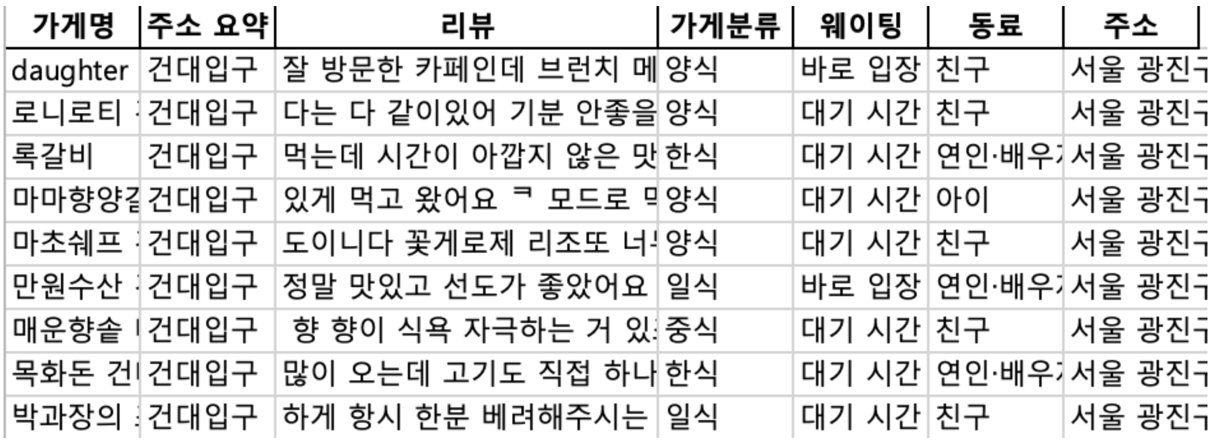
- Recursive Summarization

```
def recursive_group_and_summarize(groups, model, tokenizer, max_tokens=512, max_input_length=512,
max_output_length=200):
    while len(groups) > 1:
        combined_text = " ".join(groups)
        combined_text = clean_text(combined_text)
        new_groups = group_reviews_by_tokens(combined_text.split(". "), tokenizer, max_tokens=max_tokens)
        groups = summarize_groups(new_groups, model, tokenizer, max_input_length, max_output_length)

    final_summary = groups[0] if groups else ""
    if len(tokenizer.encode(final_summary, truncation=False)) > max_output_length:
        final_summary = summarize_groups([final_summary], model, tokenizer, max_input_length, max_output_length)[0]

    return final_summary
```

This code recursively summarizes the grouped reviews to ultimately generate a single, complete summary.

After reading the review data for each restaurant from the data file, the above functions are applied step by step to create the final summary Excel file. Below, Figure 1 shows an example of this process.

This method was applied to four different regions, and all the summarized data was compiled into a single Excel file.



가게명	주소 요약	리뷰	가게분류	웨이팅	동료	주소
daughter	건대입구	잘 방문한 카페인데 브런치 메	양식	바로 입장	친구	서울 광진구
로니로티	건대입구	다는 다 같이있어 기분 안좋을	양식	대기 시간	친구	서울 광진구
록갈비	건대입구	먹는데 시간이 아깝지 않은 맛	한식	대기 시간	연인·배우	서울 광진구
마마향양	건대입구	있게 먹고 왔어요 ㅋ 모드로	양식	대기 시간	아이	서울 광진구
마초쉐프	건대입구	도이니다 꽃게로제 리조또 너	양식	대기 시간	친구	서울 광진구
만원수산	건대입구	정말 맛있고 선도가 좋았어요	일식	바로 입장	연인·배우	서울 광진구
매운향솥	건대입구	향 향이 식욕 자극하는 거 있	중식	대기 시간	친구	서울 광진구
목화돈 건	건대입구	많이 오는데 고기도 직접 하나	한식	대기 시간	연인·배우	서울 광진구
박과장의	건대입구	하게 항시 한분 베려해주시는	일식	대기 시간	친구	서울 광진구

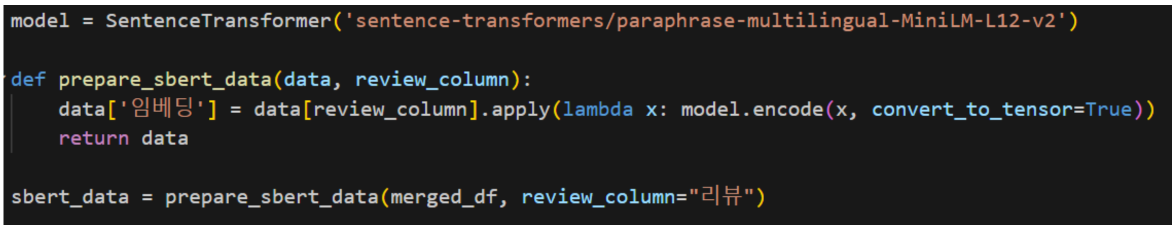
(Figure 1 – Review Summary of Konkuk University)

Section 4: Model Fitting

4-1. Recommendation through Keyword Filtering Based on SBERT and Cosine Similarity (Focusing on Review Similarity)

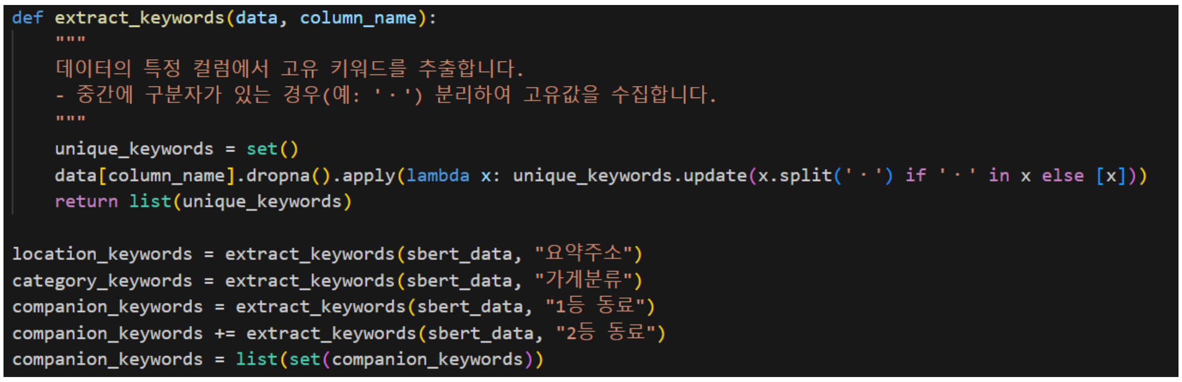
The first recommendation model calculates the similarity between review data and user input using SBERT and cosine similarity and provides personalized recommendations by adding keyword filtering.

- Embedding review data into vectors using the SBERT model



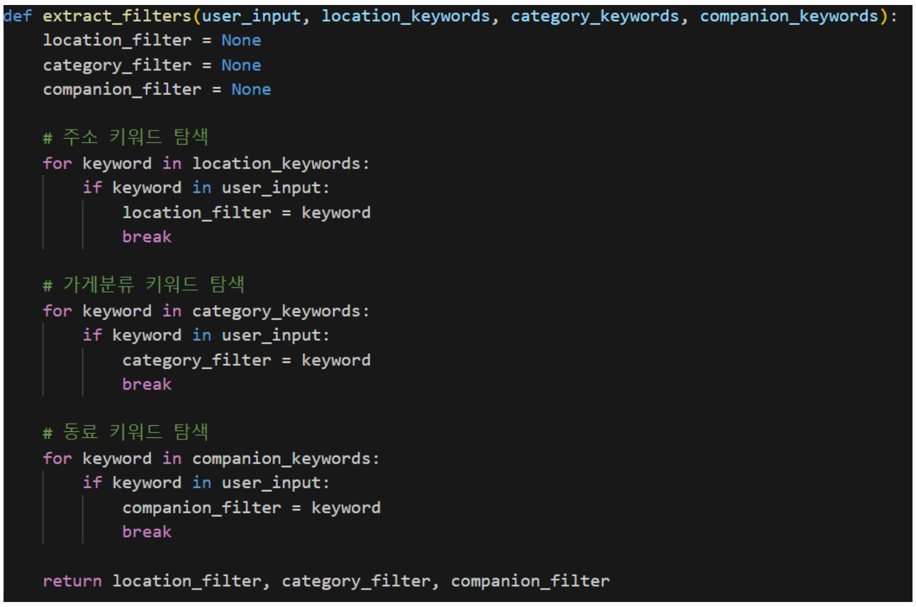
```
model = SentenceTransformer('sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2')

def prepare_sbert_data(data, review_column):
    data['임베딩'] = data[review_column].apply(lambda x: model.encode(x, convert_to_tensor=True))
    return data

sbert_data = prepare_sbert_data(merged_df, review_column="리뷰")
```

- Creating a keyword set for companion, summarized address, and restaurant category



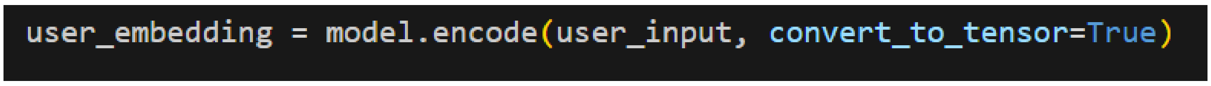
```
def extract_keywords(data, column_name):
    """
    데이터의 특정 컬럼에서 고유 키워드를 추출합니다.
    - 중간에 구분자가 있는 경우(예: '·') 분리하여 고유값을 수집합니다.
    """
    unique_keywords = set()
    data[column_name].dropna().apply(lambda x: unique_keywords.update(x.split('·') if '·' in x else [x]))
    return list(unique_keywords)

location_keywords = extract_keywords(sbert_data, "요약주소")
category_keywords = extract_keywords(sbert_data, "가게분류")
companion_keywords = extract_keywords(sbert_data, "1등 동료")
companion_keywords += extract_keywords(sbert_data, "2등 동료")
companion_keywords = list(set(companion_keywords))
```

- Check if there are keywords in the user input



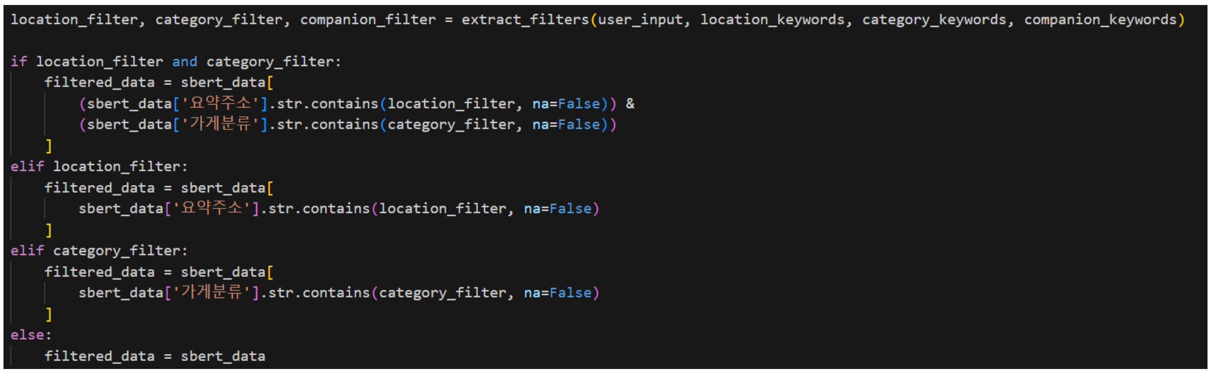
```python
def extract_filters(user_input, location_keywords, category_keywords, companion_keywords):
    location_filter = None
    category_filter = None
    companion_filter = None

    # 주소 키워드 탐색
    for keyword in location_keywords:
        if keyword in user_input:
            location_filter = keyword
            break

    # 가게분류 키워드 탐색
    for keyword in category_keywords:
        if keyword in user_input:
            category_filter = keyword
            break

    # 동료 키워드 탐색
    for keyword in companion_keywords:
        if keyword in user_input:
            companion_filter = keyword
            break

    return location_filter, category_filter, companion_filter
```

- Embed the user input into a vector using the SBERT model



```python
user_embedding = model.encode(user_input, convert_to_tensor=True)
```

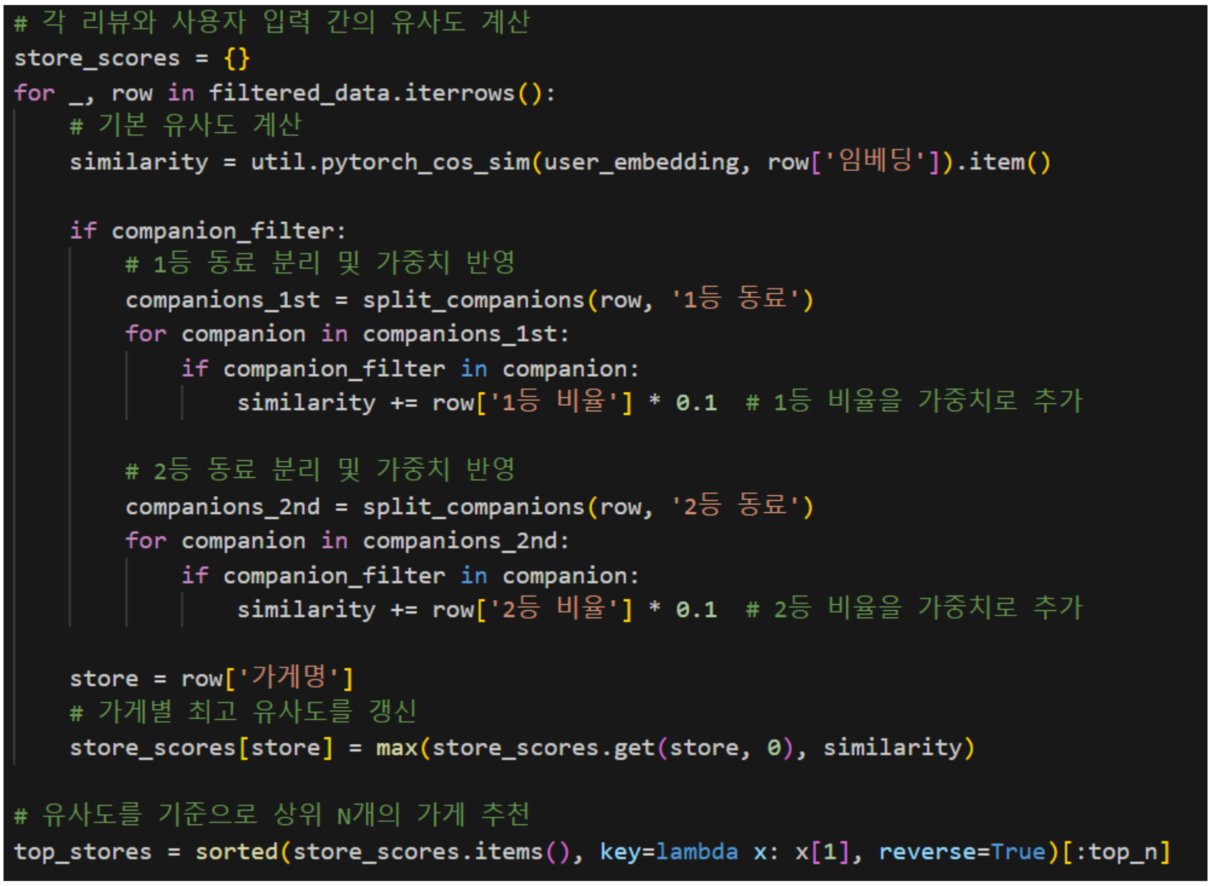
- If the user input contains keywords related to the summarized address or restaurant category, filter the restaurants based on those keywords (if no relevant keywords are found, proceed without filtering and consider all data)



```python
location_filter, category_filter, companion_filter = extract_filters(user_input, location_keywords, category_keywords, companion_keywords)

if location_filter and category_filter:
    filtered_data = sbert_data[
        (sbert_data['요약주소'].str.contains(location_filter, na=False)) &
        (sbert_data['가게분류'].str.contains(category_filter, na=False))
    ]
elif location_filter:
    filtered_data = sbert_data[
        sbert_data['요약주소'].str.contains(location_filter, na=False)
    ]
elif category_filter:
    filtered_data = sbert_data[
        sbert_data['가게분류'].str.contains(category_filter, na=False)
    ]
else:
    filtered_data = sbert_data
```

- The companion keyword is not considered an essential factor when selecting a restaurant, so it is used to calculate similarity

- Recommend restaurants based on the similarity between the review data and user input, along with the companion weight.

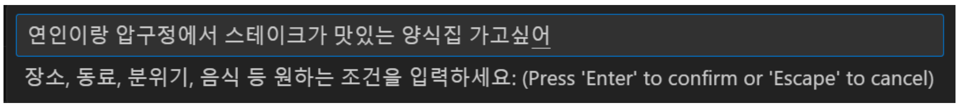


```python
# 각 리뷰와 사용자 입력 간의 유사도 계산
store_scores = {}
for _, row in filtered_data.iterrows():
    # 기본 유사도 계산
    similarity = util.pytorch_cos_sim(user_embedding, row['임베딩']).item()

    if companion_filter:
        # 1등 동료 분리 및 가중치 반영
        companions_1st = split_companions(row, '1등 동료')
        for companion in companions_1st:
            if companion_filter in companion:
                similarity += row['1등 비율'] * 0.1  # 1등 비율을 가중치로 추가

        # 2등 동료 분리 및 가중치 반영
        companions_2nd = split_companions(row, '2등 동료')
        for companion in companions_2nd:
            if companion_filter in companion:
                similarity += row['2등 비율'] * 0.1  # 2등 비율을 가중치로 추가

    store = row['가게명']
    # 가게별 최고 유사도를 갱신
    store_scores[store] = max(store_scores.get(store, 0), similarity)

# 유사도를 기준으로 상위 N개의 가게 추천
top_stores = sorted(store_scores.items(), key=lambda x: x[1], reverse=True)[:top_n]
```

The weight is set as the value obtained by multiplying the companion ratio by 0.1. This is because we want to prioritize the **text similarity between the review and the user input** as the main criterion, while using **companion similarity** as a secondary factor. The weight for the companion similarity is set to 0.1 because reducing it further would make its influence too minimal.

Additionally, the reason for multiplying both the first and second companion ratios by the same weight (0.1) is that the **first companion's ratio** is larger than the **second companion's ratio**, so multiplying by the same value allows us to create a difference in weight between the two companions.

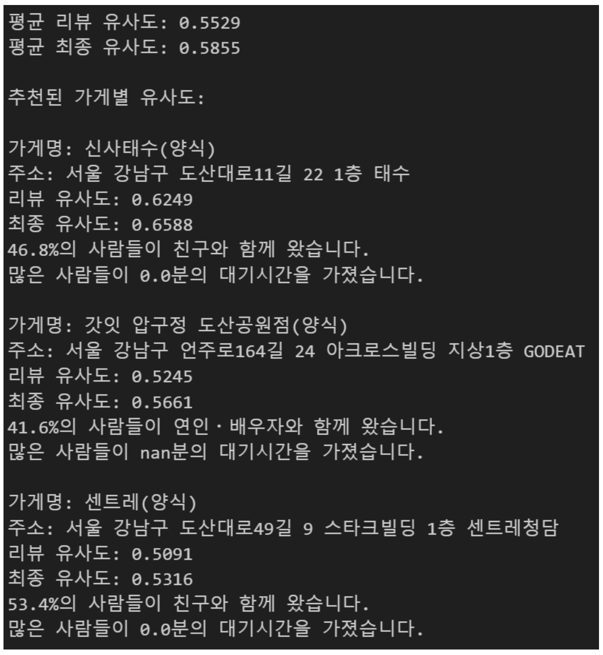
By combining the cosine similarity and companion similarity calculated in this way, we ultimately recommend the restaurant with the highest similarity to the user.



연인이랑 압구정에서 스테이크가 맛있는 양식집 가고싶어

장소, 동료, 분위기, 음식 등 원하는 조건을 입력하세요: (Press 'Enter' to confirm or 'Escape' to cancel)

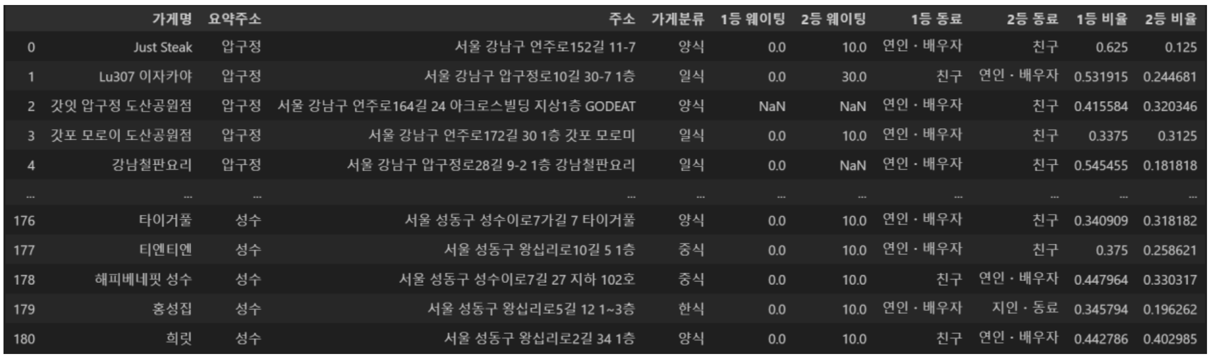
(User input)



```
평균 리뷰 유사도: 0.5529
평균 최종 유사도: 0.5855

추천된 가게별 유사도:

가게명: 신사태수(양식)
주소: 서울 강남구 도산대로11길 22 1층 태수
리뷰 유사도: 0.6249
최종 유사도: 0.6588
46.8%의 사람들이 친구와 함께 왔습니다.
많은 사람들이 0.0분의 대기시간을 가졌습니다.

가게명: 갓잇 압구정 도산공원점(양식)
주소: 서울 강남구 언주로164길 24 아크로스빌딩 지상1층 GODEAT
리뷰 유사도: 0.5245
최종 유사도: 0.5661
41.6%의 사람들이 연인·배우자와 함께 왔습니다.
많은 사람들이 nan분의 대기시간을 가졌습니다.

가게명: 센트레(양식)
주소: 서울 강남구 도산대로49길 9 스타크빌딩 1층 센트레청담
리뷰 유사도: 0.5091
최종 유사도: 0.5316
53.4%의 사람들이 친구와 함께 왔습니다.
많은 사람들이 0.0분의 대기시간을 가졌습니다.
```

(Final Recommendation Output)

In Figure 2, you can see the dataset summarizing the key information for each restaurant.

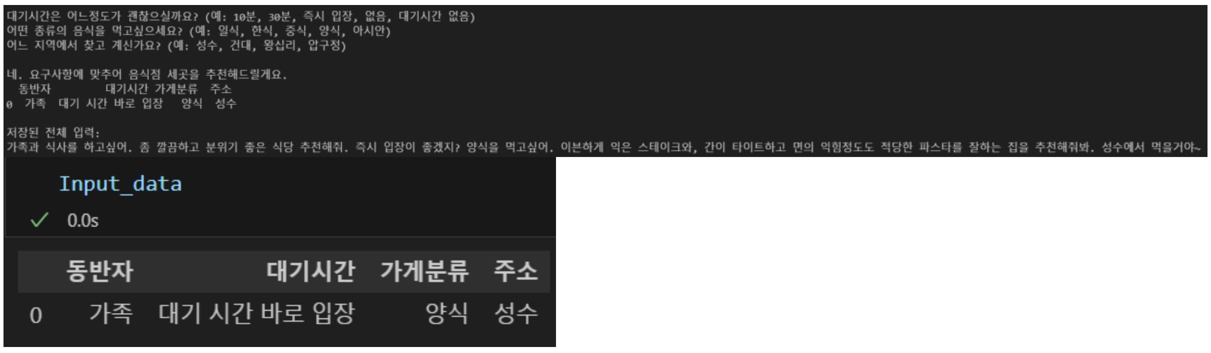


	가게명	요약주소	주소	가게분류	1등 웨이팅	2등 웨이팅	1등 동료	2등 동료	1등 비율	2등 비율
0	Just Steak	압구정	서울 강남구 언주로152길 11-7	양식	0.0	10.0	연인 · 배우자	친구	0.625	0.125
1	Lu307 이자카야	압구정	서울 강남구 압구정로10길 30-7 1층	일식	0.0	30.0	친구	연인 · 배우자	0.531915	0.244681
2	갓잇 압구정 도산공원점	압구정	서울 강남구 언주로164길 24 아크로스빌딩 지상1층 GODEAT	양식	NaN	NaN	연인 · 배우자	친구	0.415584	0.320346
3	갓포 모로이 도산공원점	압구정	서울 강남구 언주로172길 30 1층 갓포 모로미	일식	0.0	10.0	연인 · 배우자	친구	0.3375	0.3125
4	강남철판요리	압구정	서울 강남구 압구정로28길 9-2 1층 강남철판요리	일식	0.0	NaN	연인 · 배우자	친구	0.545455	0.181818
...	...	...	...	...	...	...	...	...	...	...
176	타이거풀	성수	서울 성동구 성수이로7가길 7 타이거풀	양식	0.0	10.0	연인 · 배우자	친구	0.340909	0.318182
177	티엔티엔	성수	서울 성동구 왕십리로10길 5 1층	중식	0.0	10.0	연인 · 배우자	친구	0.375	0.258621
178	해피베네핏 성수	성수	서울 성동구 성수이로7길 27 지하 102호	중식	0.0	10.0	친구	연인 · 배우자	0.447964	0.330317
179	홍성집	성수	서울 성동구 왕십리로5길 12 1~3층	한식	0.0	10.0	연인 · 배우자	지인 · 동료	0.345794	0.196262
180	희릿	성수	서울 성동구 왕십리로2길 34 1층	양식	0.0	10.0	친구	연인 · 배우자	0.442786	0.402985

(Figure 2 – Key Data for Each Restaurant)

4-2. Recommendation Model 2: Keyword Filtering Based on SBERT and Cosine Similarity (Considering Review Similarity and Various Factors)

The second recommendation model uses SBERT and cosine similarity while also balancing the incorporation of factors such as wait time, companion information, and review similarity from the user input to provide recommendations.



```
대기시간은 어느정도가 괜찮으실까요? (예: 10분, 30분, 즉시 입장, 없음, 대기시간 없음)
어떤 종류의 음식을 먹고싶으세요? (예: 일식, 한식, 중식, 양식, 아시안)
어느 지역에서 찾고 계신가요? (예: 성수, 건대, 왕십리, 압구정)

네. 요구사항에 맞추어 음식점 세곳을 추천해드릴게요.
  동반자          대기시간 가게분류  주소
0  가족  대기 시간 바로 입장   양식  성수

저장된 전체 입력:
가족과 식사를 하고싶어. 좀 깔끔하고 분위기 좋은 식당 추천해줘. 즉시 입장이 좋겠지? 양식을 먹고싶어. 이븐하게 익은 스테이크와, 간이 타이트하고 면의 익힘정도도 적당한 파스타를 잘하는 집을 추천해줘봐. 성수에서 먹을거야~
```

```
Input_data
✓ 0.0s
```

	동반자	대기시간	가게분류	주소
0	가족	대기 시간 바로 입장	양식	성수

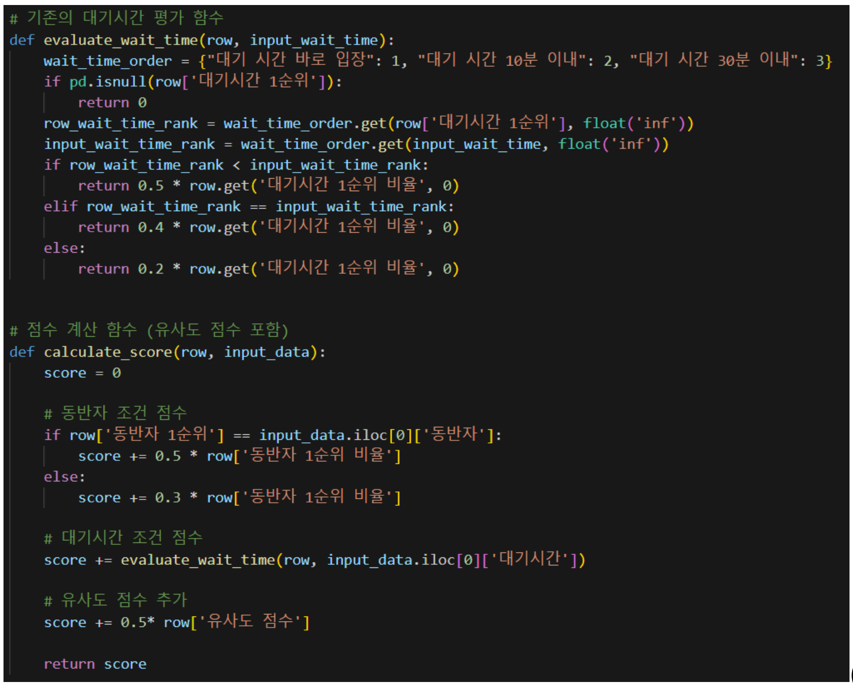
In this model, a set of keywords for companion, summarized address, wait time, and restaurant category is first created. Information such as wait time, companion details, summarized address, and restaurant category is extracted from the user input. If the user does not provide certain information, the system asks for the missing details to complete the data. (Summarized address and restaurant category are mandatory for collection, while other categories return a "nan" value if not specified). All collected inputs are then integrated into a single text, which is vectorized using SBERT along with the restaurant's summarized review. Finally, cosine similarity between the two vectors is calculated to derive the review similarity.

The wait time is compared between the user's desired wait time and the restaurant's wait time data to set a weight. If the restaurant's wait time is shorter than the user's desired wait time, a weight equal to 0.5 times the wait time ratio is applied. If the wait times are the same, a weight of 0.4 times the ratio is applied, and if the restaurant's wait time is longer, a weight of 0.2 times the ratio is applied.

For companion information, if the companion type provided by the user matches the restaurant's first-choice companion, a weight equal to 0.5 times the corresponding ratio is applied. If it matches the second-choice companion, a weight of 0.3 times the ratio is applied.

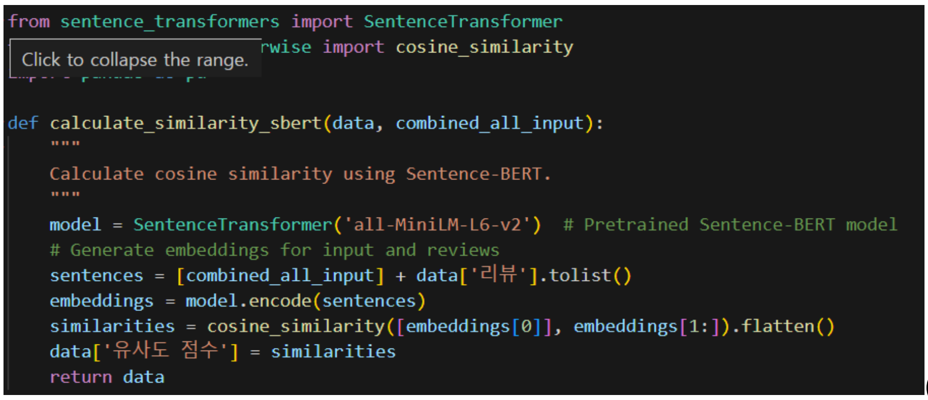
Finally, the calculated cosine similarity, wait time weight, and companion weight are all summed to determine the final restaurant score. In this process, the weights are adjusted to balance the importance of each factor.

For example, waiting time was considered an important factor when visiting a restaurant. Since the first choice wait time ratio was usually around 0.9, applying a weight of 0.5 still gave it a significant influence, so a weight of 0.5 was applied. Companion information was considered a relatively less important factor. The first-choice companion ratio was typically between 0.3 and 0.5, so even when multiplied by 0.5, it was given a lower weight compared to the wait time. For similarity, the review similarity was weighted with 0.5 to prevent it from having too much influence on the final score.

.

```python
# 기존의 대기시간 평가 함수
def evaluate_wait_time(row, input_wait_time):
    wait_time_order = {"대기 시간 바로 입장": 1, "대기 시간 10분 이내": 2, "대기 시간 30분 이내": 3}
    if pd.isnull(row['대기시간 1순위']):
        return 0
    row_wait_time_rank = wait_time_order.get(row['대기시간 1순위'], float('inf'))
    input_wait_time_rank = wait_time_order.get(input_wait_time, float('inf'))
    if row_wait_time_rank < input_wait_time_rank:
        return 0.5 * row.get('대기시간 1순위 비율', 0)
    elif row_wait_time_rank == input_wait_time_rank:
        return 0.4 * row.get('대기시간 1순위 비율', 0)
    else:
        return 0.2 * row.get('대기시간 1순위 비율', 0)


# 점수 계산 함수 (유사도 점수 포함)
def calculate_score(row, input_data):
    score = 0

    # 동반자 조건 점수
    if row['동반자 1순위'] == input_data.iloc[0]['동반자']:
        score += 0.5 * row['동반자 1순위 비율']
    else:
        score += 0.3 * row['동반자 1순위 비율']

    # 대기시간 조건 점수
    score += evaluate_wait_time(row, input_data.iloc[0]['대기시간'])

    # 유사도 점수 추가
    score += 0.5* row['유사도 점수']

    return score
```

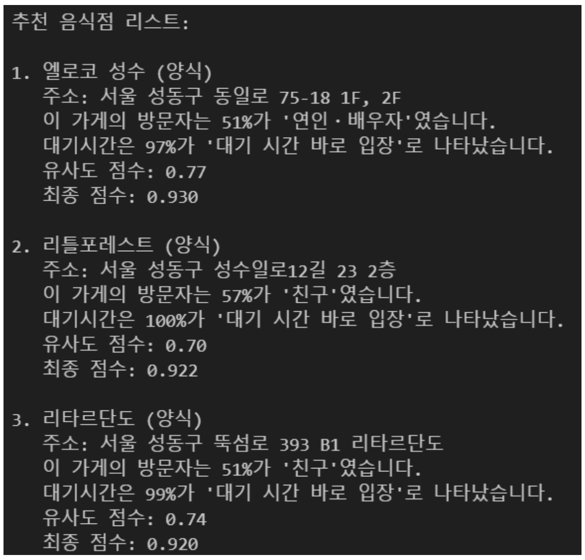
(Weight Application Code)



```
from sentence_transformers import SentenceTransformer
Click to collapse the range.rwise import cosine_similarity

def calculate_similarity_sbert(data, combined_all_input):
    """
    Calculate cosine similarity using Sentence-BERT.
    """
    model = SentenceTransformer('all-MiniLM-L6-v2')  # Pretrained Sentence-BERT model
    # Generate embeddings for input and reviews
    sentences = [combined_all_input] + data['리뷰'].tolist()
    embeddings = model.encode(sentences)
    similarities = cosine_similarity([embeddings[0]], embeddings[1:]).flatten()
    data['유사도 점수'] = similarities
    return data
```

(SBERT Code)

This model is designed to not only focus on review similarity but also to evenly incorporate user conditions such as wait time and companion information, providing more personalized recommendations.



```
추천 음식점 리스트:

1. 엘로코 성수 (양식)
   주소: 서울 성동구 동일로 75-18 1F, 2F
   이 가게의 방문자는 51%가 '연인·배우자'였습니다.
   대기시간은 97%가 '대기 시간 바로 입장'로 나타났습니다.
   유사도 점수: 0.77
   최종 점수: 0.930

2. 리틀포레스트 (양식)
   주소: 서울 성동구 성수일로12길 23 2층
   이 가게의 방문자는 57%가 '친구'였습니다.
   대기시간은 100%가 '대기 시간 바로 입장'로 나타났습니다.
   유사도 점수: 0.70
   최종 점수: 0.922

3. 리타르단도 (양식)
   주소: 서울 성동구 뚝섬로 393 B1 리타르단도
   이 가게의 방문자는 51%가 '친구'였습니다.
   대기시간은 99%가 '대기 시간 바로 입장'로 나타났습니다.
   유사도 점수: 0.74
   최종 점수: 0.920
```

(Final Recommendation Output)

4-3. Differences

Category	Model 1	Model 2
Input Data Usage	Filters keywords in the user input related to the summarized address and restaurant category	Use all information from the user input, including wait time, companion info, summarized address, and restaurant category
Recommendation Criteria	Focus on text similarity between the review and user input	Focuses on review similarity combined with wait time and companion information
Weighting Method	Focuses on review similarity for weighting	Reflects all factors, including wait time, companion information, and review similarity
Recommendation Result	Personalized recommendation based solely on text similarity	Provides a more personalized recommendation by considering all the user's requirements

(Figure 3 – Model Differences)

Figure 3 shows the differences between the two models. In conclusion, Model 1 provides a simple and accurate recommendation focused on text-based review similarity, while Model 2 offers more personalized recommendation results by balancing user conditions and review similarity.

Chapter 4: Project Evaluation

Section 1: Summary Evaluation Metrics

The recommendation system we developed is ultimately based on summarized review data, so we believed that the performance of the summarization would significantly impact the recommendations. If the summaries were effective, we expected good results when using SBERT for embedding and cosine similarity for recommendations.

TF-IDF Score

TF-IDF (Term Frequency-Inverse Document Frequency) score is a statistical method used to evaluate the importance of words in text data. This score considers words that appear frequently in individual documents but rarely across the entire dataset (all documents) as more important.

TF-IDF consists of two components: the first is **Term Frequency (TF)**, which represents how often a word appears in a specific document, and the second is **Inverse Document Frequency (IDF)**, which indicates how rare the word is across all documents. By multiplying these two components, more meaningful words are given higher scores than commonly occurring words. TF-IDF is a highly useful tool for extracting important keywords from text data or calculating the relevance between user input and reviews.

BERT Score

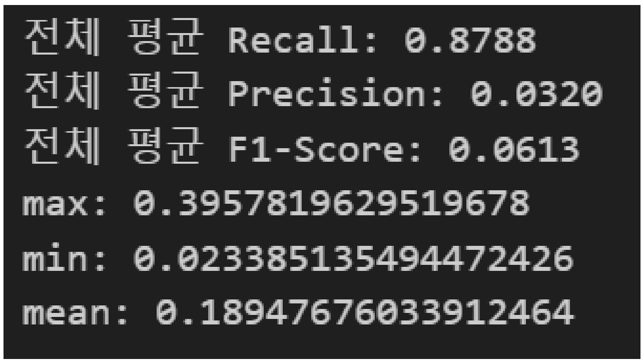
BERT Score is a score based on the BERT (Bidirectional Encoder Representations from Transformers) model, designed to evaluate similarity between texts more precisely. Unlike traditional word-level similarity calculations, BERT Score understands text contextually and measures semantic similarity. It uses text embeddings generated by the BERT model to calculate the semantic relationship between input data and comparison data, allowing for an evaluation that encompasses context and meaning, rather than just simple text similarity. This feature makes it particularly suitable for sentence-level meaning comparison or analyzing the relationship between reviews and user input.

In contrast, **BLEU** and **ROUGE** rely on n-gram matching, making them unsuitable for evaluating abstract summaries and potentially leading to distorted results when there is a significant length difference between the summary and the original text. For these reasons, BERT Score was chosen in this project to assess semantic similarity.

.

Section 2: Summary Evaluation

2-1. TF-IDF Score



```
전체 평균 Recall: 0.8788
전체 평균 Precision: 0.0320
전체 평균 F1-Score: 0.0613
max: 0.3957819629519678
min: 0.023385135494472426
mean: 0.18947676033912464
```

(Figure 4 – Summary TF-IDF Score)

TF-IDF (Term Frequency-Inverse Document Frequency) is a method for quantifying text data and measuring the importance of words, primarily by analyzing the frequency of individual words within the text. However, since the text summarization used in this project was abstract, as seen in Figure 4, the TF-IDF score tends to be relatively low. TF-IDF relies on the frequency of specific word occurrences, which does not effectively reflect semantic similarity between words and is sensitive to word rearrangements. Therefore, it had limitations in capturing the semantic similarity of the text data.

- **Recall = 0.8788**

Recall is a metric that indicates how well the summary captures the important content from the original text. A high Recall Score of 0.8788 shows that the summary includes the key terms from the original text effectively. However, Recall focuses only on "inclusion" and does not consider how "relevant" the included content is.

- **Precision = 0.0320**

Precision measures how relevant the content included in the summary is to the original text. A low Precision value indicates that the summary contains a lot of irrelevant or unnecessary information that is not related to the original text.

- **F1-Score = 0.0613**

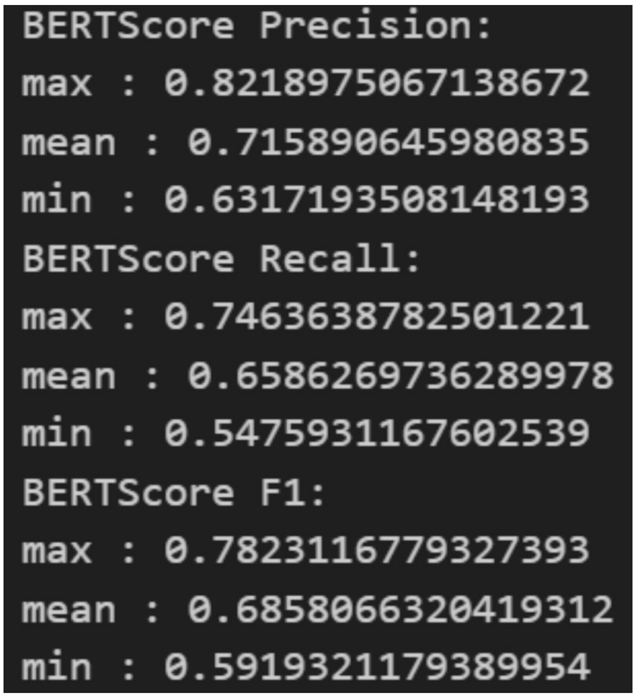
The F1-Score is the harmonic mean of Precision and Recall, providing a comprehensive evaluation of both metrics. The low Precision, which offsets the high Recall, results in a significantly lowered F1-Score. This indicates that the summary has a low relevance to the original text and contains a large amount of unnecessary information.

- **Max, Mean, Min of Similarity**

Given that both the average and maximum similarity are low, it indicates that the TF-IDF features of the document pairs do not match significantly.

Since it is an abstractive summarization, the summary does not simply copy words from the original text but instead understands the context and meaning, generating new sentences or expressions. As a result, the TF-IDF similarity, which is based on direct word-level matching, is bound to be low. I plan to re-evaluate the similarity using the BERT Score, which measures the contextual similarity between the original text and the summary through BERT embeddings.

2-2. BERT Score



```
BERTScore Precision:
max : 0.8218975067138672
mean : 0.715890645980835
min : 0.6317193508148193
BERTScore Recall:
max : 0.7463638782501221
mean : 0.6586269736289978
min : 0.5475931167602539
BERTScore F1:
max : 0.7823116779327393
mean : 0.6858066320419312
min : 0.5919321179389954
```

(Figure 5 – Summarization BERT Score)

BERT Score is used to evaluate the semantic similarity between sentences and provides three main metrics: Precision, Recall, and F1 Score. The summary score can be viewed in Figure 5.

Precision indicates how semantically similar the summary is to the original text, and the average value of 0.7159 shows that the summary aligns quite well with the original text. However, the minimum value of 0.6317 suggests that some summaries may not have fully captured the meaning of the original text.

Recall indicates how much of the original information is preserved in the summary, and the average value of 0.6586 means that most summaries maintain the original information well. However, the minimum value of 0.5476 suggests that some summaries may have omitted important information from the original text.

Finally, the F1 Score, which measures the balance between Precision and Recall, has an average value of 0.6858, showing that the summaries are semantically well-aligned with the original text. However, the minimum value of 0.5919 indicates that some summaries may not be fully semantically consistent with the original text.

In conclusion, the summarization model showed good overall performance, but it is evident that some summaries did not fully capture the meaning and information.

Section 3: Roles by Team Members

- Seo MinSeok **(PM):** Model 2(Considering various factors model), creating presentation slides (PPT), data collection, and preprocessing.

- Kwon SoonJae: Model 1(Review similarity-focused model,), report, evaluation metrics, data collection, and preprocessing.

- Lee SangMin: Review data summarization model, presentation, data collection, and preprocessing.

- Lee JungSeok: Review data summarization model, report, data collection, and preprocessing.