여행 블로그 데이터 분석

TextAnalytics Assignment

AI빅데이터융합경영학과 20212568 이서연

CONTENT

01

데이터 수집

02

데이터 전처리

- Cleansing
- Tokenization
- Stopword Removal

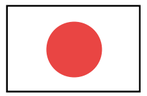
03

모델링

- Word Cloud
- LDA

주제

Japan



USA

Thailand

Italy

France

5개국 여행 관련 네이버 블로그 데이터 분석

01 데이터 수집

Crawling

네이버 API는 하나의 주제에 대해 최대 1000개까지만 수집이 가능하기 때문에 BeautifulSoup, Selenium만 활용하여 데이터 수집

크롤링을 위한 2개의 함수 생성

- parse_blog_post
: 주어진 url에 있는 포스트의 제목 / 본문 내용을 크롤링하여 반환하는 함수

- get_result
: 페이지에 접속하여 각 포스트의 url을 저장한 후 parse_blog_post 함수에 전달하여 데이터를 추출하는 함수

01 데이터 수집

Crawling

- parse_blog_post()

```
def parse_blog_post(driver, url):

    driver.get(url) # 해당 url에 접속
    time.sleep(3)
    soup = BeautifulSoup(driver.page_source, 'lxml') # 웹 페이지의 소스코드를 BeatifulSoup를 이용해 파싱

    # ifrmae의 src URL을 추출
    iframe_tag = soup.find('iframe')
    if iframe_tag is not None:
        iframe_src = iframe_tag['src']
    else:
        return None # 만약 iframe 태그가 없다면 None을 반환

    # iframe_src가 상대 URL일 경우 'https://blog.naver.com'을 추가
    if not iframe_src.startswith('http'):
        iframe_src = 'https://blog.naver.com' + iframe_src
```

'iframe': HTML 문서 내에 다른 HTML 문서를 삽입할 수 있는 태그
네이버 블로그의 경우 'iframe' 내부에 본문이 위치하는 경우가 많기 때문에
iframe의 src URL을 추출

01 데이터 수집

Crawling

- parse_blog_post()

```
# 본문 URL로 다시 요청
res = requests.get(iframe_src)
res.raise_for_status() # 요청이 실패하면 예외 발생시킴
soup = BeautifulSoup(res.text, 'lxml') # 요청한 결과를 다시 BeatifulSoup로 파싱
```

```
# 블로그 제목 추출
title_tag = soup.find('meta', property='og:title')
if title_tag is not None:
    title = title_tag['content']
else:
    return None
```

'meta' 태그 중 'property' 속성의 값이 'og:title'인 요소를 선택하여
블로그 제목 추출

```
# 블로그 본문 추출
content_tag = soup.find('div', {'class': ['se-main-container', 'se_component_wrap sect_dsc __se_component_area']})
if content_tag is not None:
    content = content_tag.get_text()
else:
    return None
```

'div' 태그를 이용하여
블로그 본문 내용 추출

```
return {
    'title': title,
    'content': content
}
```

제목과 본문 내용을
딕셔너리 형태로 반환

01 데이터 수집

Crawling

- get_result()

```
def get_result(query, page_count=100):
    blog_posts = [] # 데이터를 저장할 리스트

    driver = webdriver.Chrome('C:/Users/tjdus/Downloads/chromedriver_win32/chromedriver') # 웹 드라이버 초기화

    for page in range(1, page_count + 1): # 지정한 페이지 수만큼 반복

        url = f'https://search.naver.com/search.naver?where=post&sm=tab_jum&query={query}&start={30 * (page-1) + 1}'
        driver.get(url)
        time.sleep(3)

        soup = BeautifulSoup(driver.page_source, 'html.parser') # 페이지 HTML 파싱
        links = soup.select('a.api_txt_lines.total_tit') # 링크 저장
```

- 네이버 블로그는 하나의 페이지 당 30개의 포스트 존재
- 100개의 페이지를 반복하며 하나의 검색어 당 총 3000개의 포스트 url 추출 시도

01 데이터 수집

Crawling

- get_result()

```
        for link in links:
            post_link = link['href']
            try: # 크롤링 도중 문제가 발생할 경우를 대비해 예외 처리
                post = parse_blog_post(driver, post_link)

                if post is not None: # 추출 결과가 None이 아닐  때만
                    post['link'] = post_link # 링크도 추가
                    blog_posts.append(post) # 추출 결과를 리스트에 추가
            except:
                continue


    driver.quit()  # 웹 드라이버 종료

    return blog_posts # 결과 리스트 반환
```

저장한 url을 parse_blog_post 함수에 전달하여 데이터 추출

01 데이터 수집

Crawling

- '일본 여행' 검색어에 대한 데이터 수집 → 2397개

```
query = '일본 여행'
japan_result = get_result(query)
Japan = pd.DataFrame(japan_result)
Japan.insert(0, 'country', '일본')
```

- '미국 여행' 검색어에 대한 데이터 수집 → 1952개

```
query = '미국 여행'
USA_result = get_result(query)
USA = pd.DataFrame(USA_result)
USA.insert(0, 'country', '미국')
```

- '태국 여행' 검색어에 대한 데이터 수집 → 2168개

```
query = '태국 여행'
Thailand_result = get_result(query)
Thailand = pd.DataFrame(Thailand_result)
Thailand.insert(0, 'country', '태국')
```

- '이탈리아 여행' 검색어에 대한 데이터 수집 → 2410개

```
query = '이탈리아 여행'
Italy_result = get_result(query)
Italy = pd.DataFrame(Italy_result)
Italy.insert(0, 'country', '이탈리아')
```

- '프랑스 여행' 검색어에 대한 데이터 수집 → 1921개

```
query = '프랑스 여행'
France_result = get_result(query)
France = pd.DataFrame(France_result)
France.insert(0, 'country', '프랑스')
```

01 데이터 수집

데이터 병합

나라별로 수집한 데이터를 병합하여 하나의 데이터프레임 형태로 만듦

```
total = pd.concat([JAPAN, USA, THAILAND, ITALY, FRANCE])
total = total.drop_duplicates() # 중복값 제거
total = total.reset_index(drop=True)
```

01 데이터 수집

수집 결과

	country	title	content	link
0	일본	일본여행추천 도쿄맛집 긴자 미도리스시	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n일본여행추천 도쿄맛집 긴...	https://blog.naver.com/hacelluvia/223108143016
1	일본	일본 여행 후쿠오카 모츠나베 맛집 모츠나베 타슈 다이묘점	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n \n\n...	https://blog.naver.com/esilvia/223114234954
2	일본	일본여행준비물 교통패스 오사카 주유패스 예약 수령 안내	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n오사카여행을 앞두고 있는...	https://blog.naver.com/lsh5755/223115815690
3	일본	일본여행 홋카이도 렌트카 여행 계획짜는 중	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n안녕하세요. 여행 인플루언...	https://blog.naver.com/jamjma98/223085377457
4	일본	후쿠오카 유후인 여행 코스 버스투어 패키지 일본여행 추천	\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n \n\n\...	https://blog.naver.com/luvxox/223103066729
...	...	...	...	...
10772	프랑스	프랑스 여행 준비할 것	\n\n\n\n\n\n프랑스 여행을 준비하면서 이래저래 손품 발품 팔았던 경험을 토대로...	https://blog.naver.com/cheesekim1208/222993087365
10773	프랑스	[프랑스 여행#4] 몽마르트 언덕에서 한끼, Deli's Cafe	\n\n\n\n\n\n\n\n/ 프랑스 여행#4 /Vacation to the Fran...	https://blog.naver.com/gpfla67/221223950594
10774	프랑스	[프랑스여행] 환상의 섬 코르시카 1편	\n\n\n\n\n\n나의 사전에 불가능은 없다.\n나폴레옹\n\n\n\n\n\n\n\...	https://blog.naver.com/vocae/221816426000
10775	프랑스	프랑스 여행 추천, 숨겨진 프랑스 소도시 5곳	\n\n\n\n\n\n안녕하세요~프랑스 여행 하면 흔히 파리 에펠탑을 떠올릴 텐데요,프...	https://blog.naver.com/mastersalt/221805889050
10776	프랑스	프랑스 여행 8일째	\n\n\n\n\n\n8일차 2022-09-21(Wednesday)아침 일찍 어제 돌...	https://blog.naver.com/mhkimwine/222881329232

10777 rows × 4 columns

'여행 국가', '블로그 제목', '블로그 본문 내용', '블로그 링크' 4개의 Column을 가진 총 10777개의 데이터 추출

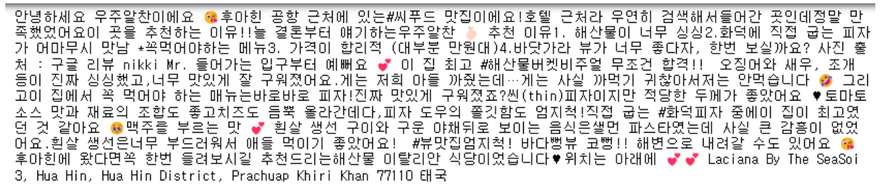
02 데이터 전처리

Cleaning

- 개행 문자 및 제로 너비 공백 제거

```
data['content'] = data['content'].apply(lambda x: x.replace('\n', ' '))
data['content'] = data['content'].apply(lambda x: x.replace('\u200b', ' '))
```

- 이모지 및 이모티콘 제거


안녕하세요 우주알찬이에요 😘후아힌 공항 근처에 있는#씨푸드 맛집이에요!호텔 근처라 우연히 검색해서들어간 곳인데정말 만족했었어요이 곳을 추천하는 이유!!늘 결론부터 얘기하는우주알찬 👌🏻 추천 이유1. 해산물이 너무 싱싱2.화덕에 직접 굽는 피자가 어마무시 맛남 +쏙먹어야하는 메뉴3. 가격이 합리적 (대부분 만원대)4.바닷가라 뷰가 너무 좋다자, 한번 보실까요? 사진 출처 : 구글 리뷰 nikki Mr. 들어가는 입구부터 예뻐요 💕 이 집 최고 #해산물버켓비주얼 무조건 합격!! 오징어와 새우, 조개 등이 진짜 싱싱했고,너무 맛있게 잘 구워졌어요.게는 저희 아들 까줬는데…게는 사실 까먹기 귀찮아서저는 안먹습니다 🤭 그리고이 집에서 꼭 먹어야 하는 매뉴는바로바로 피자!진짜 맛있게 구워졌죠?씬(thin)피자이지만 적당한 두께가 좋았어요 ♥토마토 소스 맛과 재료의 조합도 좋고치즈도 듬뿍 올라간데다,피자 도우의 쫄깃함도 엄지척!직접 굽는 #화덕피자 꿀에이 집이 최고였던 것 같아요 😡맥주를 부르는 맛 💕 흰살 생선 구이와 구운 야채뒤로 보이는 음식은샐먼 파스타였는데 사실 큰 감흥이 없었어요.흰살 생선은너무 부드러워서 애들 먹이기 좋았어요! #뷰맛집엄지척! 바다뷰뷰 코랭!! 해변으로 내려갈 수도 있어요 😘 후아힌에 왔다면꼭 한번 들려보시길 추천드리는해산물 이탈리안 식당이었습니다♥위치는 아래에 💕💕 Laciana By The SeaSoi 3, Hua Hin, Hua Hin District, Prachuap Khiri Khan 77110 태국

블로그 특성 상 본문 내용에 각종 이모지 존재

```
# ASCII 코드 및 한글 유니코드 범위를 제외한 모든 문자 제거

def remove_non_ascii_except_korean(text):
    return re.sub(r'[^\x00-\x7F\uAC00-\uD7A3]+','', text)
```

```
data['content'] = data['content'].apply(lambda x: remove_non_ascii_except_korean(x))
```

02 데이터 전처리

Cleaning

- 띄어쓰기 교정

pycospacing 패키지 활용

```
spacer = pykospacing.Spacing()
```

```
data['content'] = data['content'].progress_apply(lambda x: spacer(x))
```

- 특수문자 제거

구두점을 제외한 특수문자는 공백으로 치환

```
def replace_non_alphanumeric3(text):

    special_characters_pattern = re.compile(r'(?![#w#s])[^.,!?]')
    text = special_characters_pattern.sub(' ', text)

    return text
```

```
data['content'] = data['content'].apply(replace_non_alphanumeric3)
```

02 데이터 전처리

Cleaning

- 맞춤법 검사

py-hanspell 라이브러리 활용

```
def check_spelling(data):

    sentences = data.split('.')  # 문장을 마침표(.)를 기준으로 분리
    checked_sentences = []
    for sentence in sentences:
        result = spell_checker.check(sentence.strip())  # 문장 앞뒤에 공백을 제거
        checked_sentences.append(result.checked)

    return '. '.join(checked_sentences)  # 수정된 문장을 다시 합쳐서 반환
```

```
data['content'] = data['content'].progress_apply(check_spelling)
```

py-hanspell은 한번에 검사할 수 있는 문장의 길이가 최대 500자이기 때문에 문장 분리 후 검사를 진행한 뒤 다시 합치는 방식으로 수행

02 데이터 전처리

Tokenization

- Mecab 패키지를 활용하여 Tokenization을 수행하는 함수 생성

```
def tokenization_korean(text):
    sents = kss.split_sentences(text) # sentence tokenization

    p = re.compile('[^ㄱ-ㅎㅏ-ㅣ가-힣 ]')
    results = []

    for sent in sents:
        result = []
        tokens = mecab.morphs(sent) # word tokenization

        for token in tokens:
            token = p.sub('', token) # 특수문자 제거
            result.append(token)
        results.extend(result)

    result = ' '.join(results)

    return result
```

sentence tokenization 진행 후
sentence 안에서 word tokenization 진행

```
data.insert(3, 'token', data['content'].progress_apply(tokenization_korean))
```

Tokenization 수행 후 새로운 Column에 추가

02 데이터 전처리

Stopword Removal

- Stopword 파일 불러오기

```
with open('Korean_Stopwords.txt', 'r', encoding='utf-8') as f:
    stopwords = f.readlines()
stopwords = [x.replace('₩n','') for x in stopwords]
```

- Stopword 제거 함수 생성

```
def rm_stopword(text):
    result = []

    if isinstance(text, str):
        tokens = text.split() # 띄어쓰기 기준으로 단어 분리

        for t in tokens:
            t = str(t)
            if t not in stopwords: # stopword 리스트에 없는 단어만 저장
                result.append(t)
        result = ' '.join(result) # 단어들을 다시 합침

    return result
```

```
data['token'] = data['token'].apply(rm_stopword)
data = data[data['token'].apply(lambda x: len(x) != 0)]
data = data.rename(columns = {'token':'preprocessed_content'})
```

Stopword 제거 후 전처리가 완료됨을 의미하는
'preprocessed_content'로 Column명 변경

03 MODELING

WordCloud

- WordCloud 생성 함수

```
def wordcloud(query):
    country = data[data['country'] == query]
    total_text = ''

    for _, row in country.iterrows():
        total_text += row['preprocessed_content'] # 각 데이터의 본문 내용을 하나로 합침

    cloud = WordCloud(font_path = font_path,
                      background_color = 'white',
                      width=800, height=800)

    # 단어의 빈도를 계산하여 이를 기반으로 wordcloud 생성
    country_cloud = Japan_cloud = cloud.generate_from_text(total_text)
    country_arr = country_cloud.to_array()

    return country_arr
```

함수의 인자로 여행 국가를 받고,
해당 나라의 WordCloud 생성

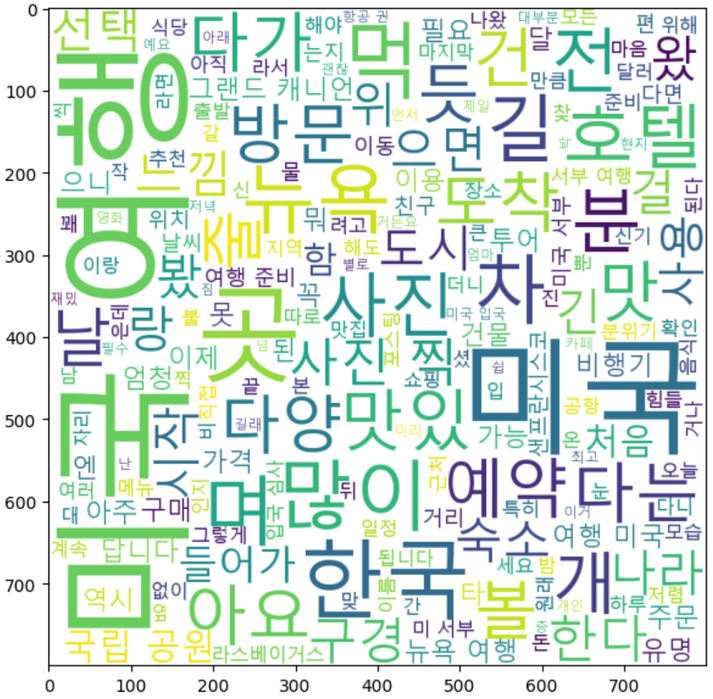
03 MODELING - WordCloud

일본

- '일본 여행'과 '일본'이라는 단어가 가장 많이 나타남
- '오사카', '교토', '후쿠오카', '도쿄' 등 일본 도시명이 나타남
- '차', '라면' 등 일본 특산품과 관련된 단어들이 나타남

03 MODELING - WordCloud

미국



- '미국 여행'과 '미국'이라는 단어가 가장 많이 나타남
- '뉴욕', '라스베이거스', '샌프란시스코' 등 미국 도시명이 나타남
- '예약', '사진', '구경' 등의 여행 관련 용어들이 나타남

03 MODELING - WordCloud

태국

- '태국 여행'과 '태국'이라는 단어가 가장 많이 나타남
- '치앙마이', '방콕', '태국 도시명이 나타남
- 파타야, 야시장 등 태국 특색을 나타내는 단어들이 나타남

03 MODELING - WordCloud

이탈리아

- '이탈리아 여행'과 '이탈리아'라는 단어가 가장 많이 나타남
- '로마', '베네치아', '피렌체' 등 이탈리아 도시명이 나타남
- '트레비 분수', '우피치 미술관' 등 이탈리아 유명소들이 나타남

03 MODELING - WordCloud

프랑스

- '프랑스 여행'과 '프랑스'라는 단어가 가장 많이 나타남
- '파리', '스트라스부르' 등 프랑스 도시명이 나타남
- '에펠탑', '루브르 박물관', '몽마르트르 언덕' 등 프랑스 유명소들이 나타남

03 MODELING - WordCloud

Conclusion

- 모든 나라에서 '나라 이름'과 '나라 이름 + 여행' 단어가 가장 크게 나타남

- 각 나라별로 도시와 유명소가 뚜렷하게 나타남

- 유명소 이외에도, 나라의 특색을 알 수 있는 단어들이 나타남

- 장소만큼 음식이 큰 비중을 차지하는 것으로 보아, 음식 역시 여행에서 중요한 요소임을 알 수 있음

03 MODELING - Topic Modeling (LDA)

Topic Modeling (LDA)

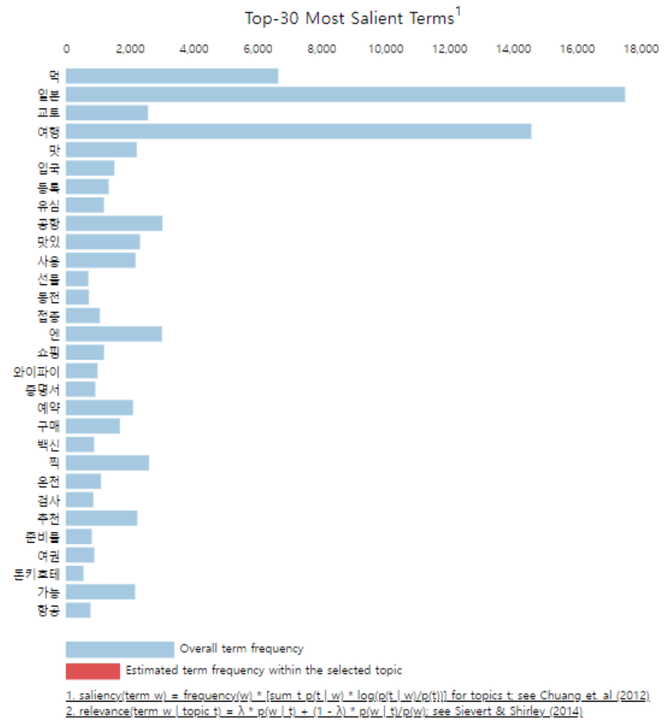
- 각 나라별 4개의 Topic에 대한 LDA model 생성 함수

```
def LDA_modeling(country):

    # 문서에서 단어의 frequency 계산
    content_word = country['preprocessed_content'].apply(lambda x: x.split())
    word_dict = corpora.Dictionary(content_word)
    corpus = [word_dict.doc2bow(text) for text in content_word]

    # LDA 모델 훈련 (Topic 4개로 설정)
    n_topics = 4
    LDA_model = gensim.models.ldamodel.LdaModel(corpus, num_topics = n_topics, id2word=word_dict, passes = 100)

    # LDA 시각화를 위한 객체
    vis = pyLDAvis.gensim_models.prepare(LDA_model, corpus, word_dict)

    # LDA 모델 및 시각화 객체 반환
    return LDA_model, vis
```

03 MODELING - Topic Modeling (LDA)

일본

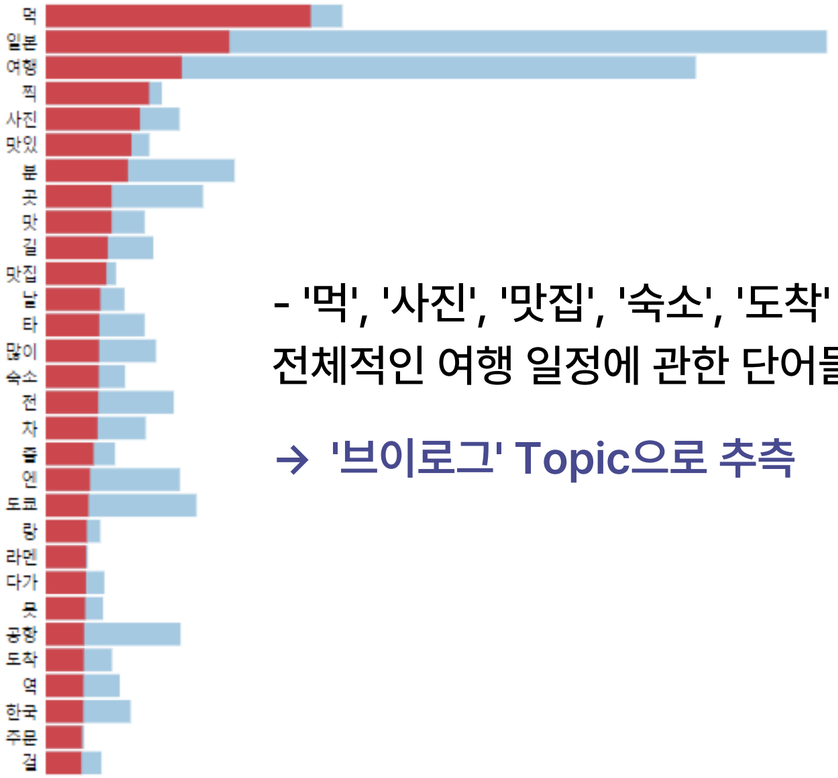
- Topic과 전체 단어 분포



03 MODELING - Topic Modeling (LDA)

일본

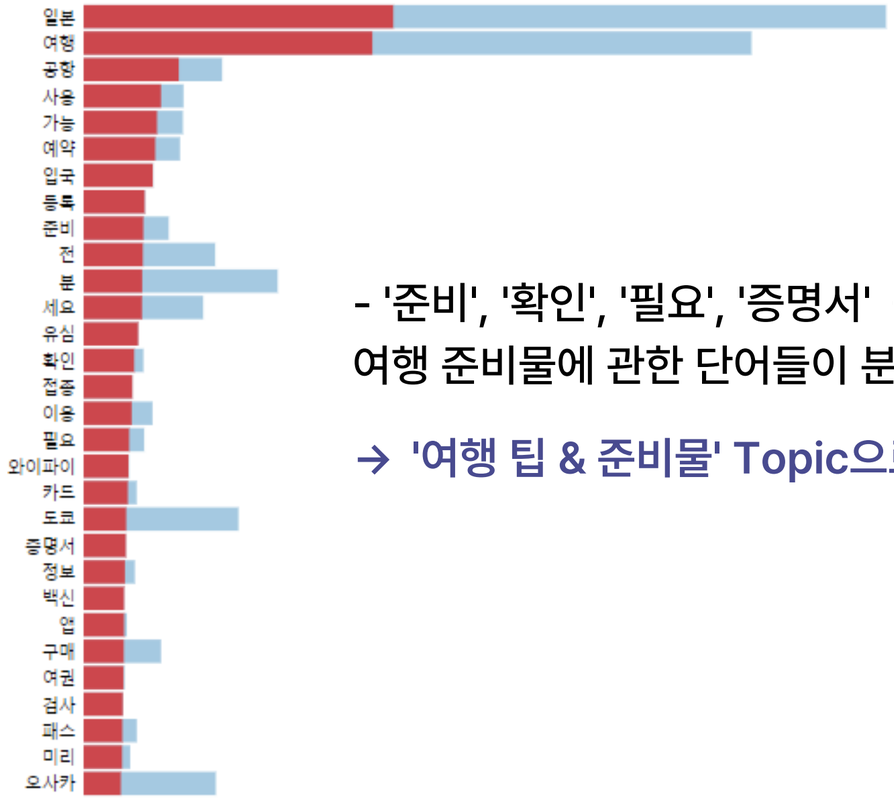
- Topic1



- '먹', '사진', '맛집', '숙소', '도착' 등
전체적인 여행 일정에 관한 단어들이 분포

→ '브이로그' Topic으로 추측

- Topic2



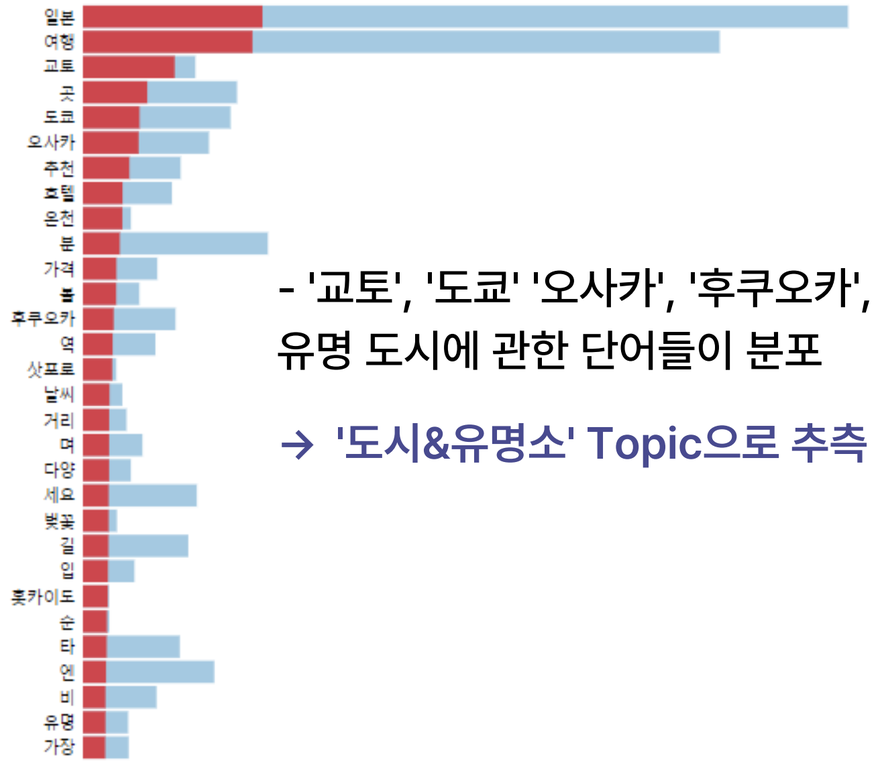
- '준비', '확인', '필요', '증명서' 등
여행 준비물에 관한 단어들이 분포

→ '여행 팁 & 준비물' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

일본

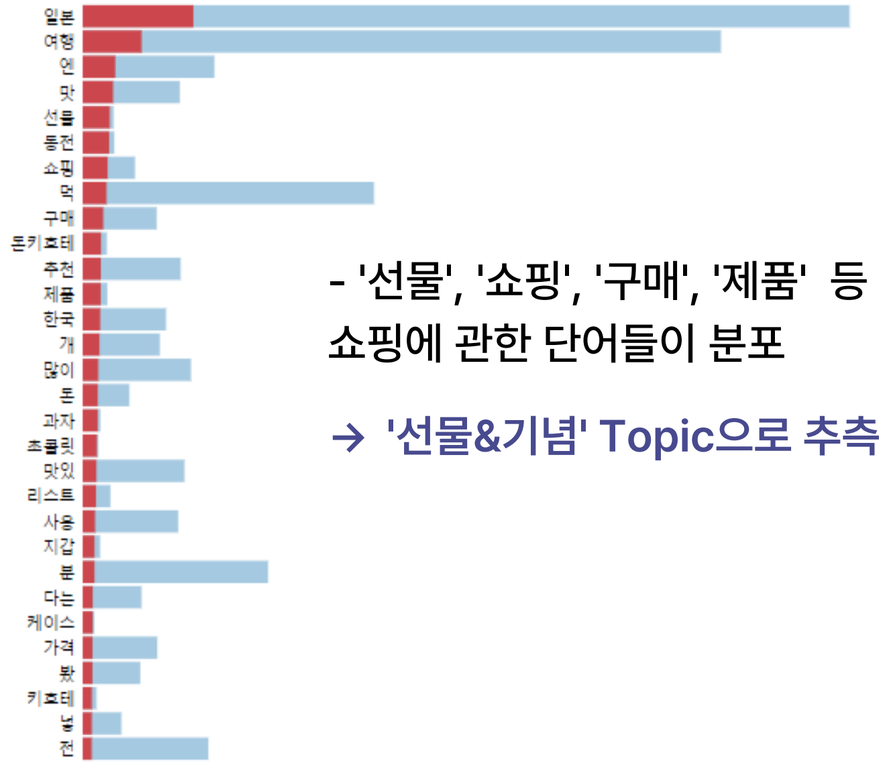
- Topic3



- '교토', '도쿄' '오사카', '후쿠오카', '유명' 등 유명 도시에 관한 단어들이 분포

→ '도시&유명소' Topic으로 추측

- Topic4



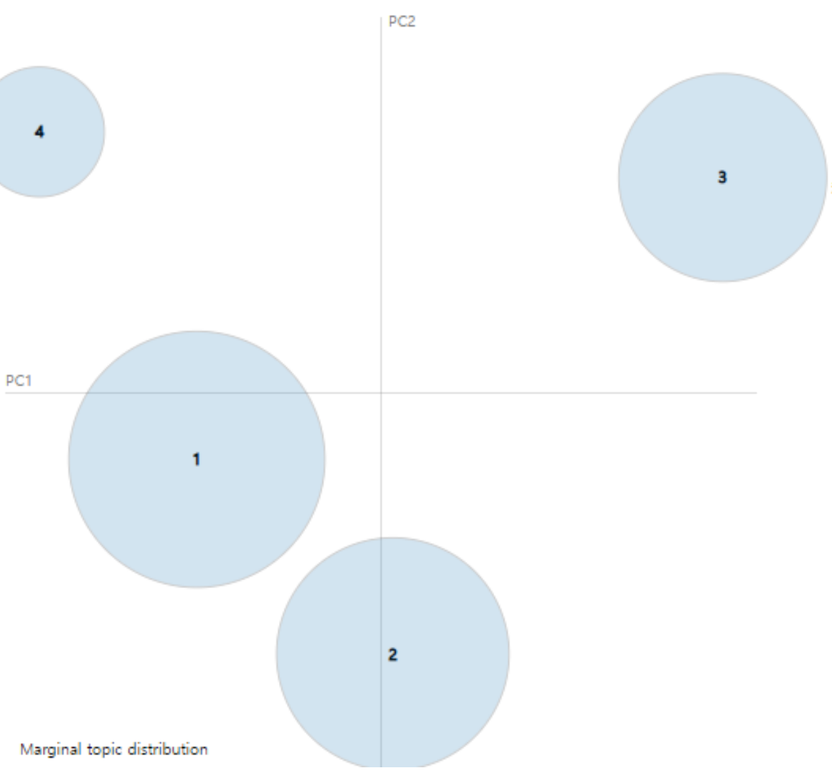
- '선물', '쇼핑', '구매', '제품' 등 쇼핑에 관한 단어들이 분포

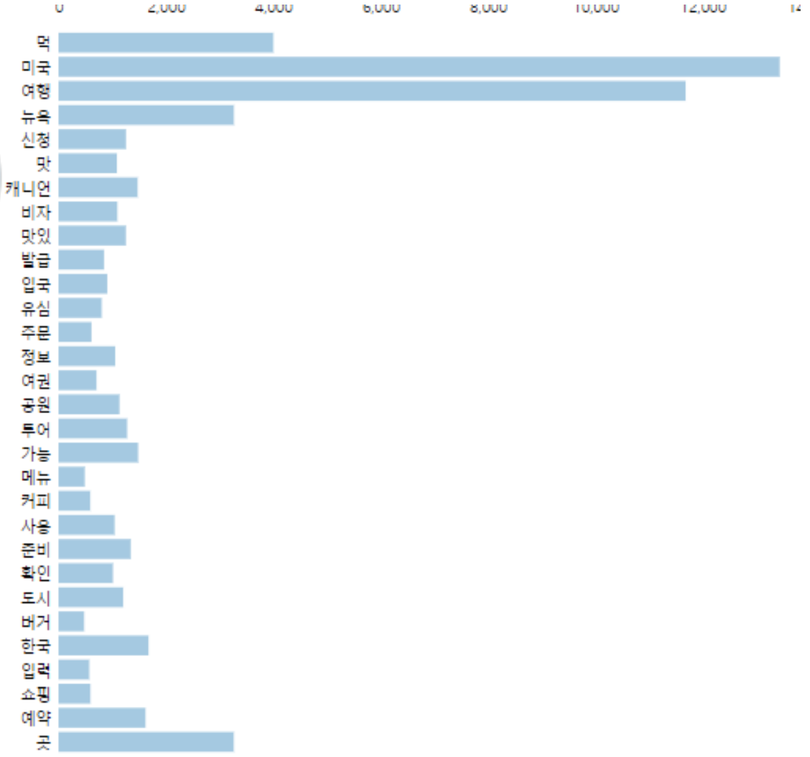
→ '선물&기념' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

미국

- Topic과 전체 단어 분포

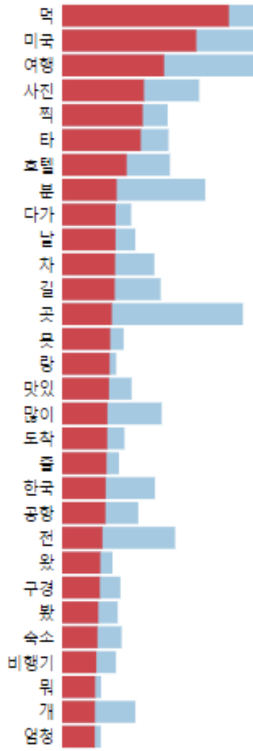




03 MODELING - Topic Modeling (LDA)

미국

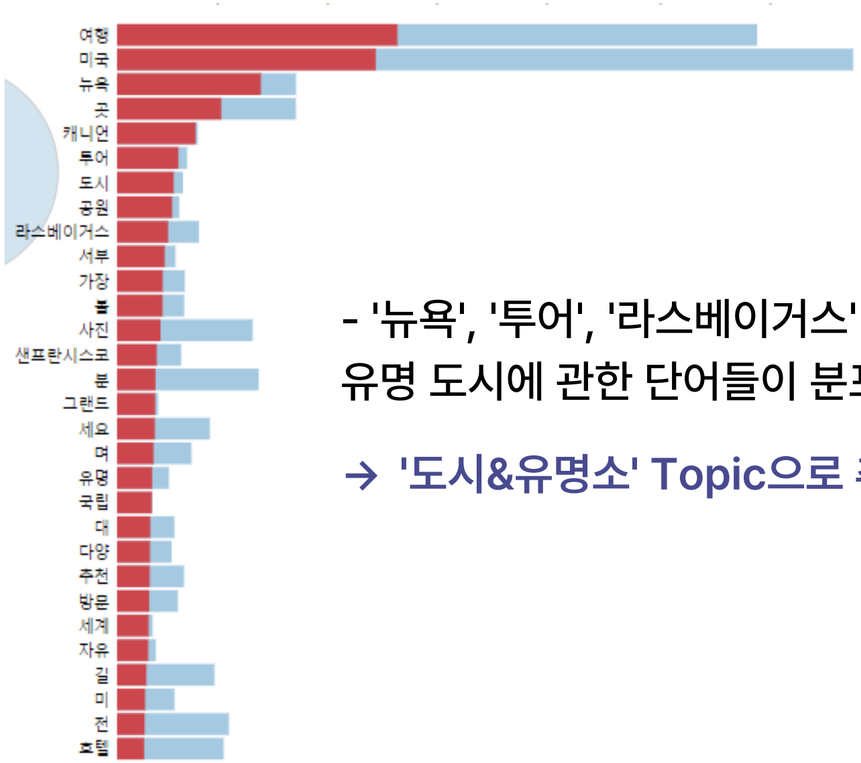
- Topic1



- '먹', '사진' '구경', '비행기' 등
전체적인 여행 일정에 관한 단어들이 분포

→ '브이로그' Topic으로 추측

- Topic2



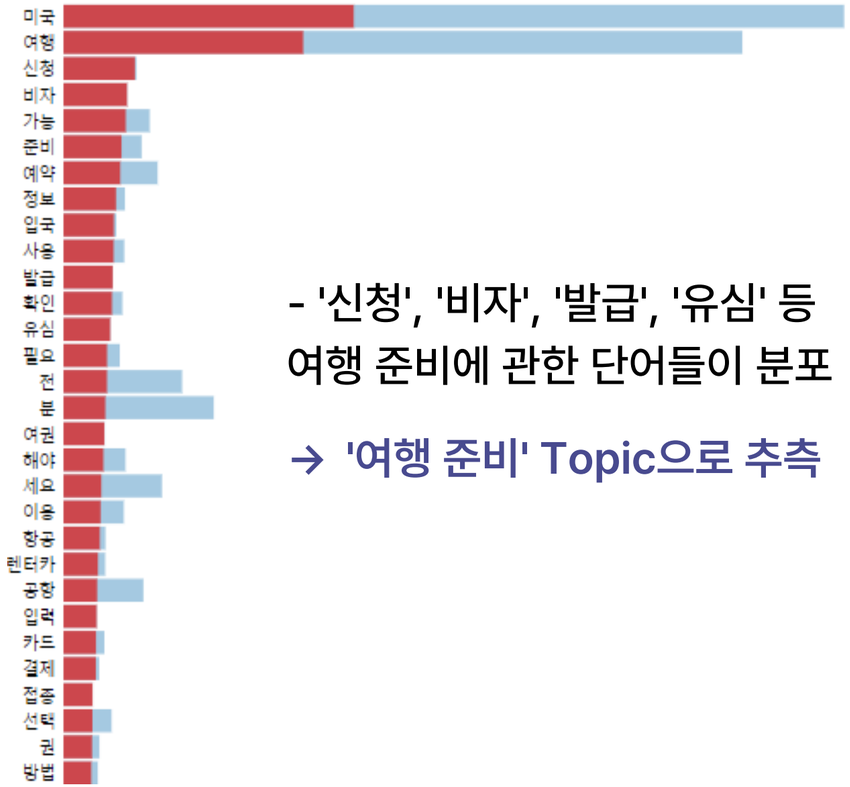
- '뉴욕', '투어', '라스베이거스' 등
유명 도시에 관한 단어들이 분포

→ '도시&유명소' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

미국

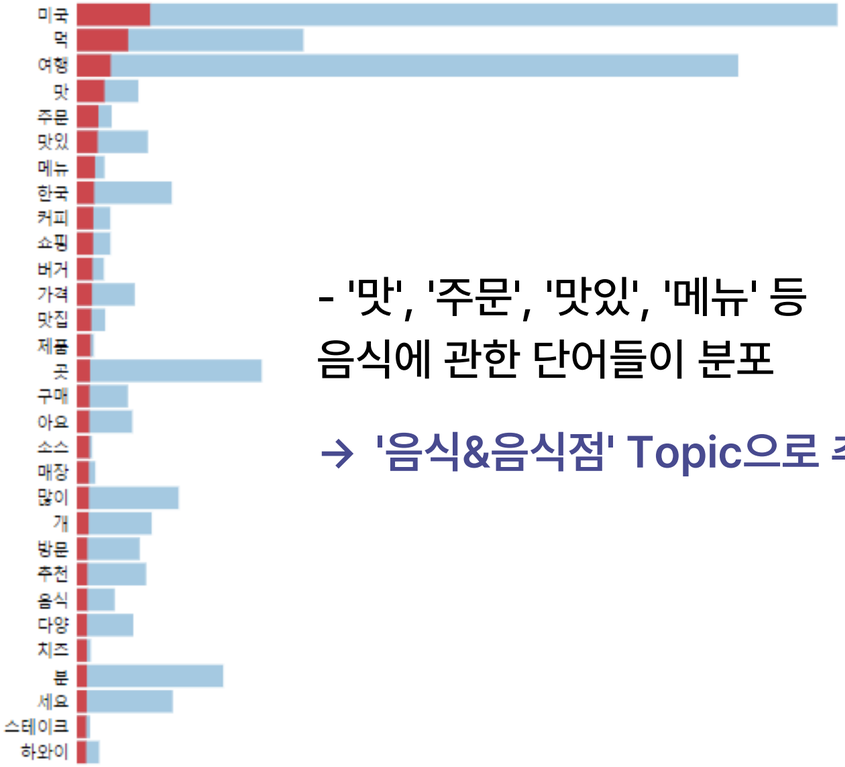
- Topic3



- '신청', '비자', '발급', '유심' 등
여행 준비에 관한 단어들이 분포

→ '여행 준비' Topic으로 추측

- Topic4



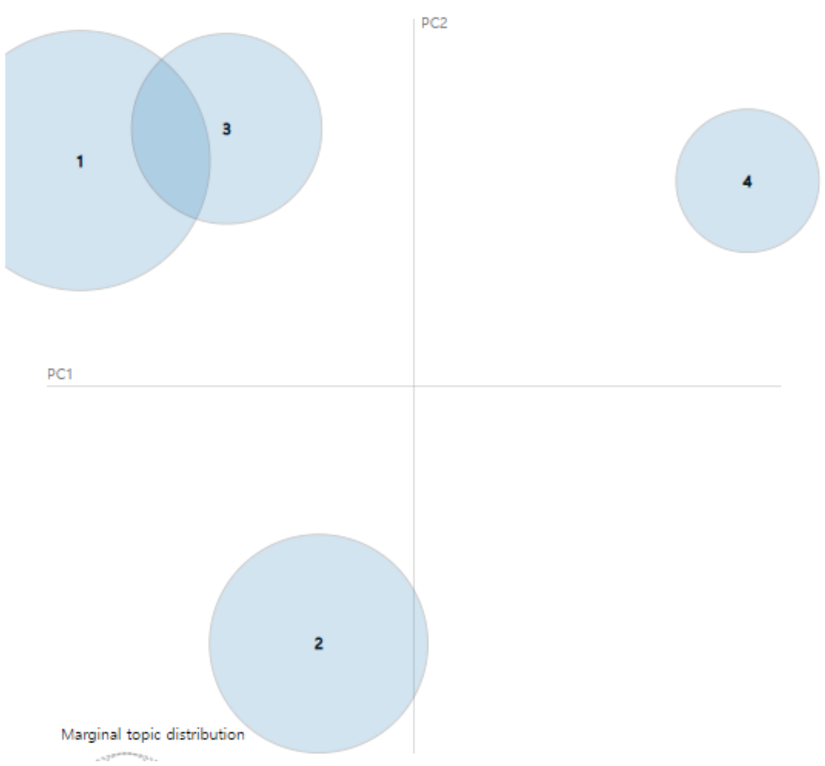
- '맛', '주문', '맛있', '메뉴' 등
음식에 관한 단어들이 분포

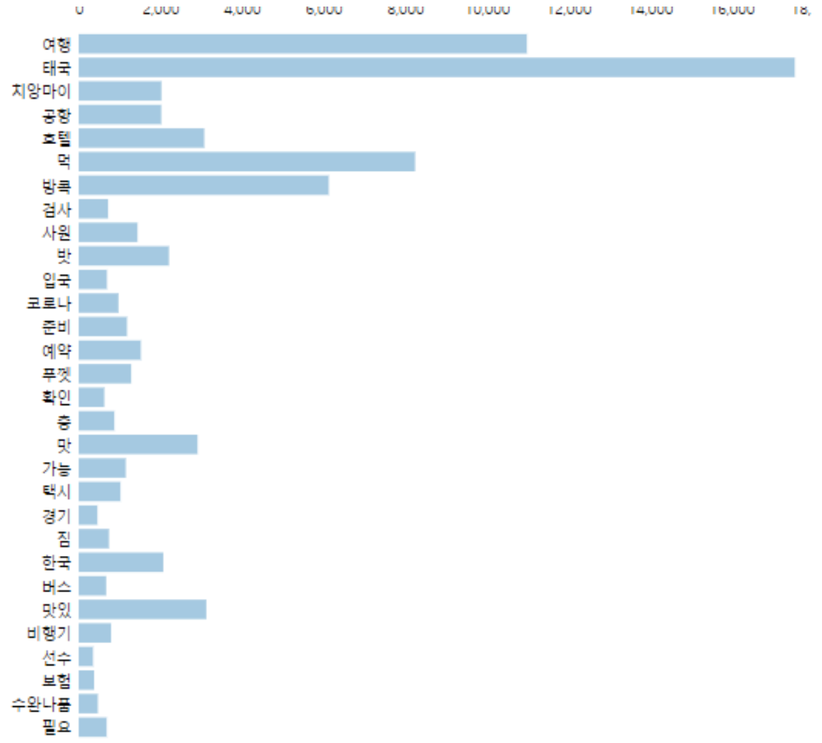
→ '음식&음식점' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

태국

- Topic과 전체 단어 분포





03 MODELING - Topic Modeling (LDA)

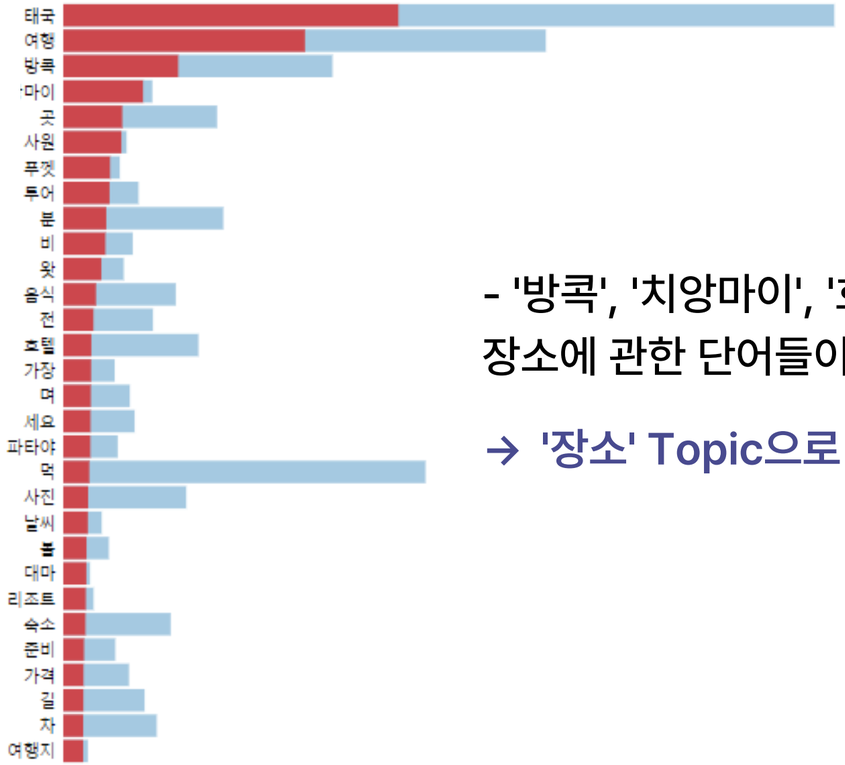
태국

- Topic1

- '맛있', '사진', '숙소', '친구', '느낌' 등
전반적인 여행 일정에 관한 단어들이 분포

→ '브이로그' Topic으로 추측

- Topic2



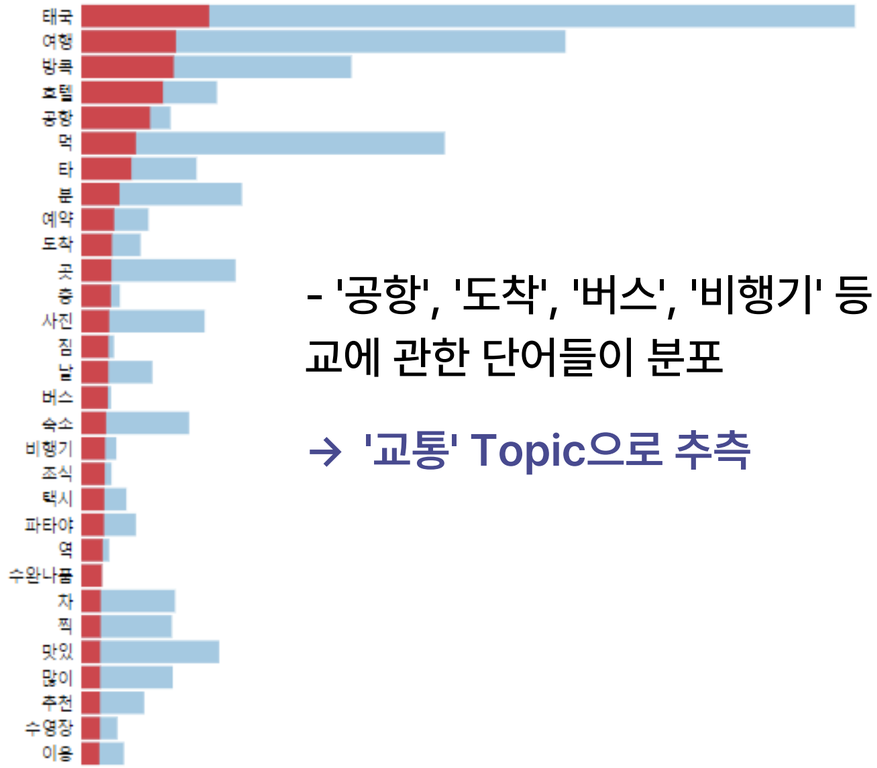
- '방콕', '치앙마이', '호텔', '리조트' 등
장소에 관한 단어들이 분포

→ '장소' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

태국

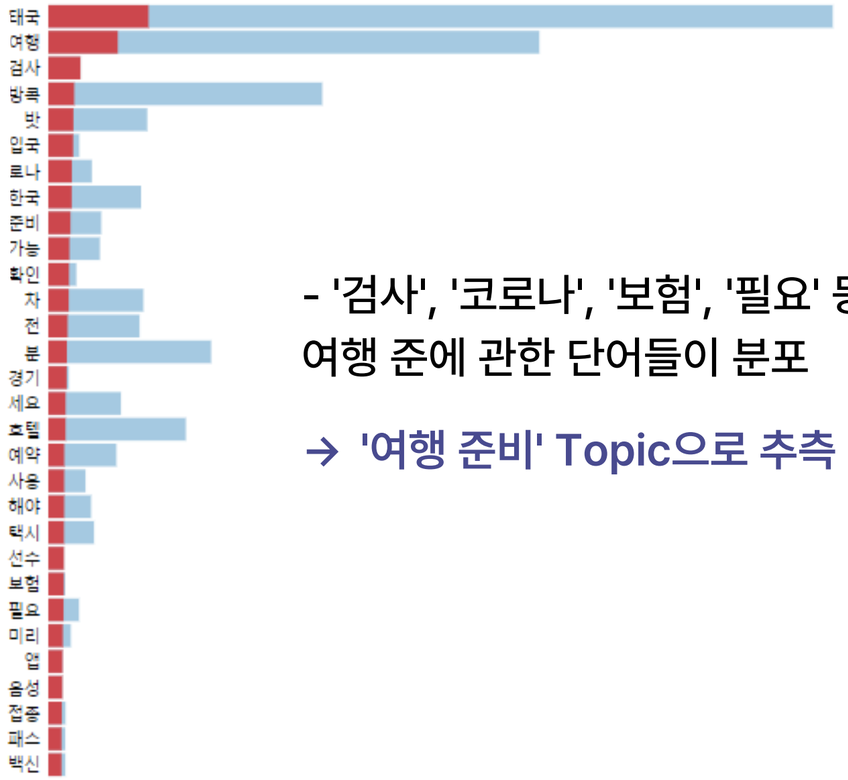
- Topic3



- '공항', '도착', '버스', '비행기' 등 교에 관한 단어들이 분포

→ '교통' Topic으로 추측

- Topic4



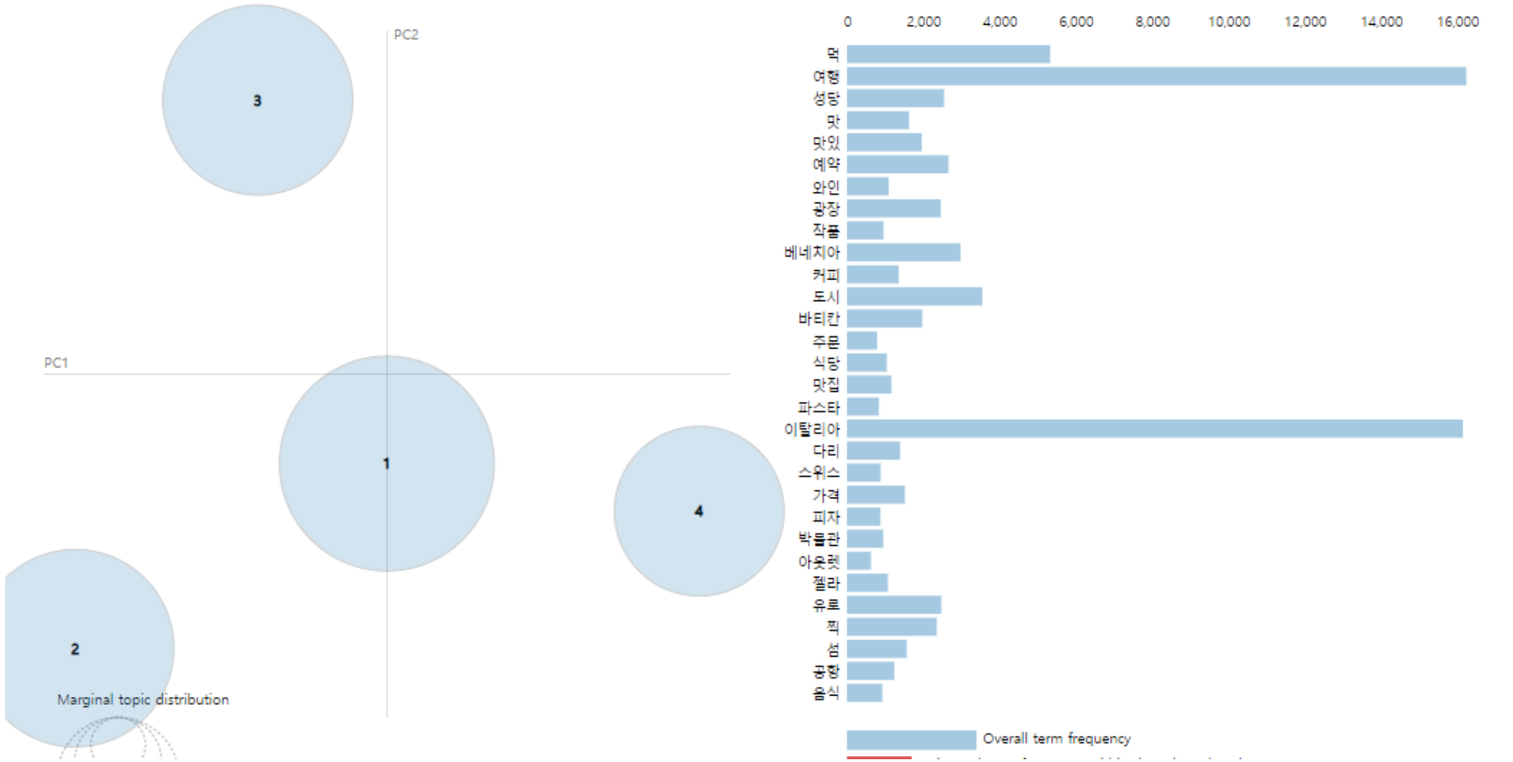
- '검사', '코로나', '보험', '필요' 등 여행 준에 관한 단어들이 분포

→ '여행 준비' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

이탈리아

- Topic과 전체 단어 분포



03 MODELING - Topic Modeling (LDA)

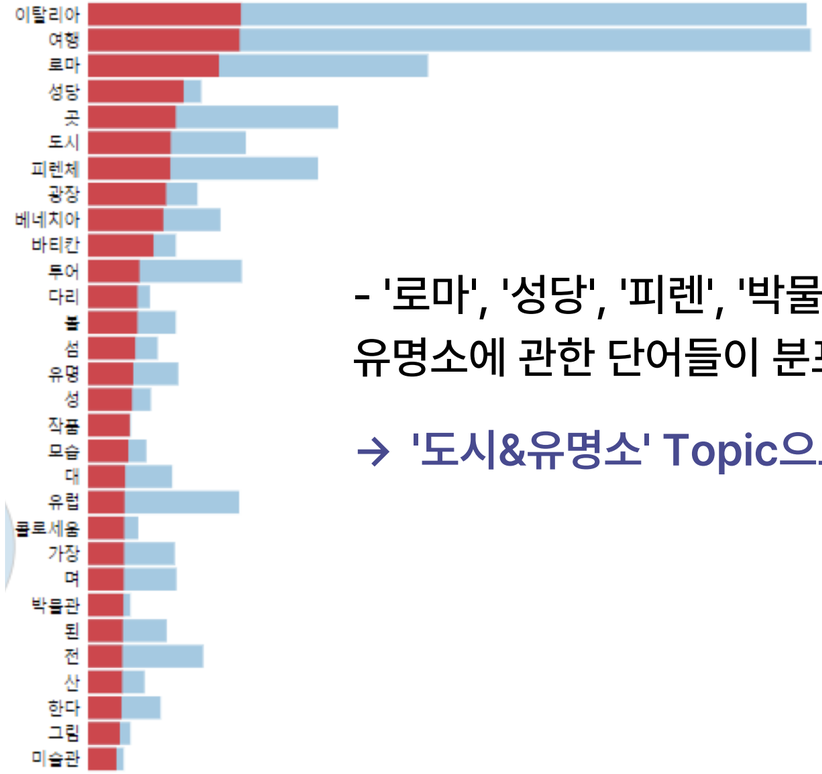
이탈리아

- Topic1

- '사진', '도착', '날씨', '구경' 등
전반적인 여행 일정에 관한 단어들이 분포

→ '브이로그' Topic으로 추측

- Topic2



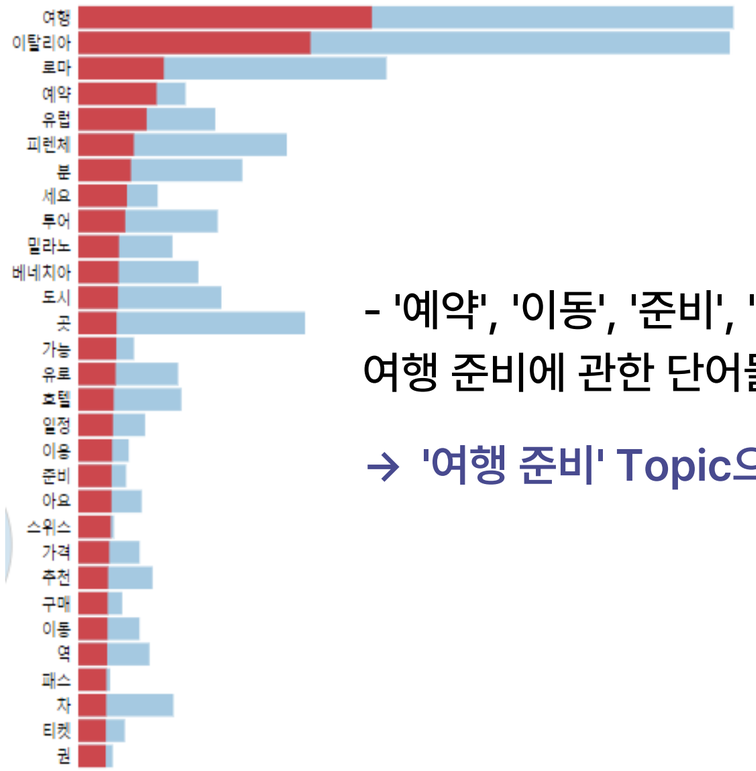
- '로마', '성당', '피렌', '박물관' 등
유명소에 관한 단어들이 분포

→ '도시&유명소' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

이탈리아

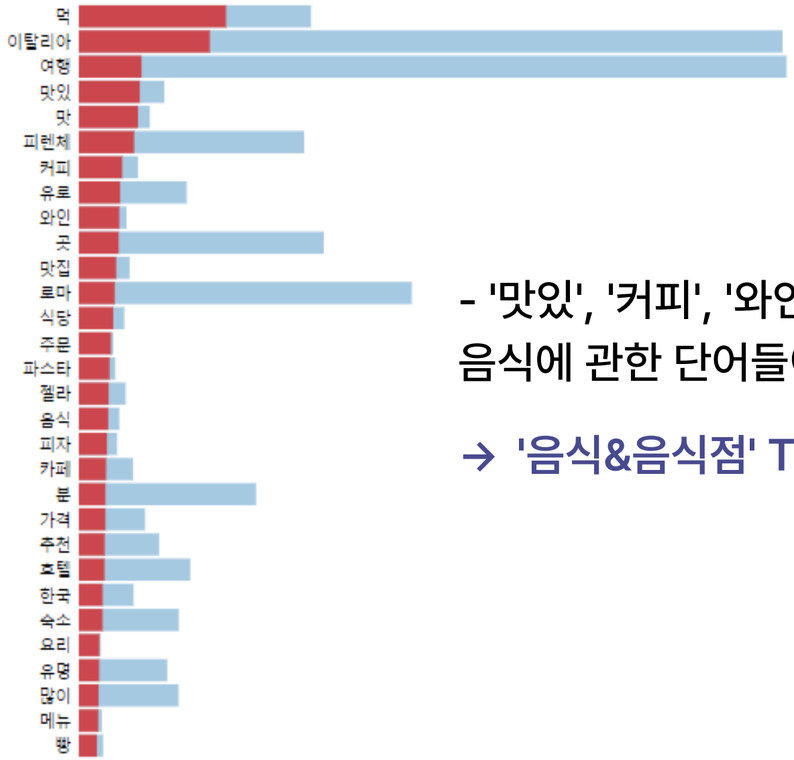
- Topic3



- '예약', '이동', '준비', '티켓' 등 여행 준비에 관한 단어들이 분포

→ '여행 준비' Topic으로 추측

- Topic4



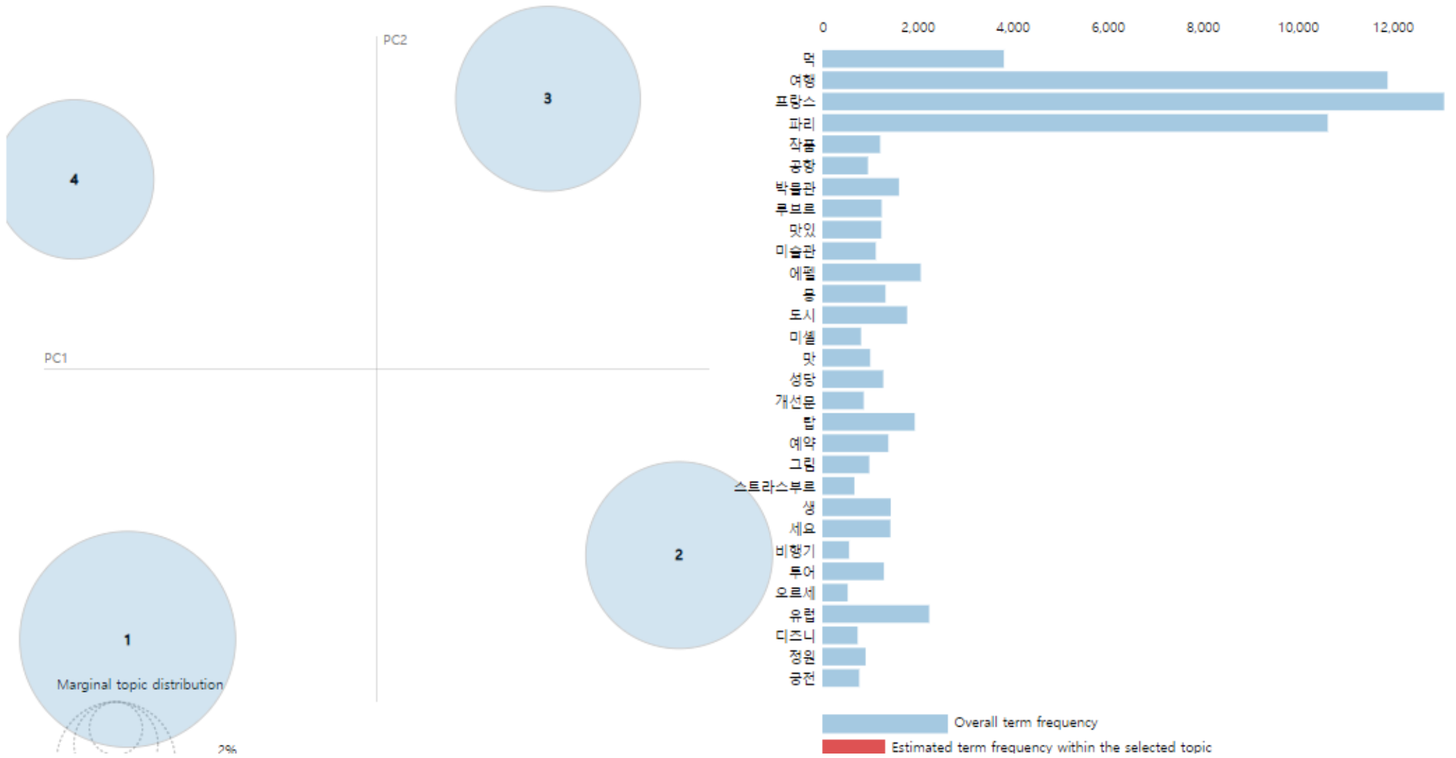
- '맛있', '커피', '와인', '식당', '음식' 등 음식에 관한 단어들이 분포

→ '음식&음식점' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

프랑스

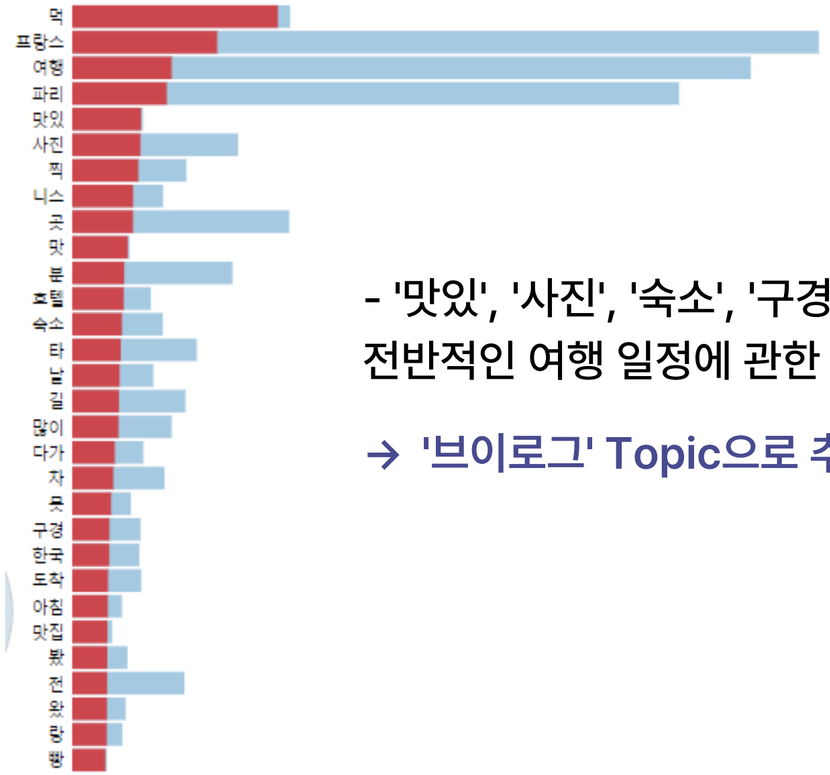
- Topic과 전체 단어 분포



03 MODELING - Topic Modeling (LDA)

프랑스

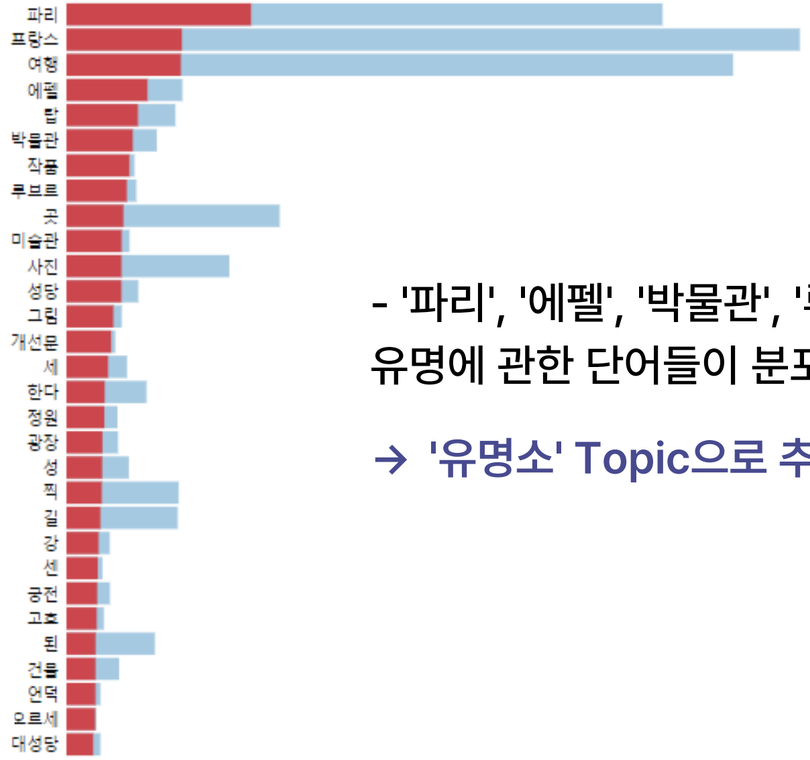
- Topic1



- '맛있', '사진', '숙소', '구경' 등
전반적인 여행 일정에 관한 단어들이 분포

→ '브이로그' Topic으로 추측

- Topic2



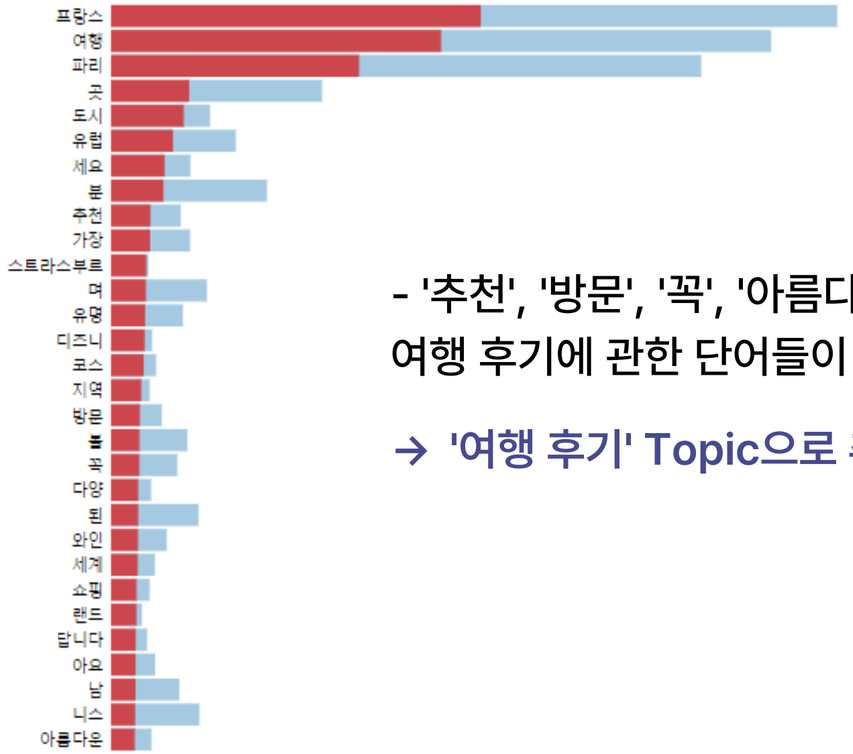
- '파리', '에펠', '박물관', '루브르' 등
유명에 관한 단어들이 분포

→ '유명소' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

프랑스

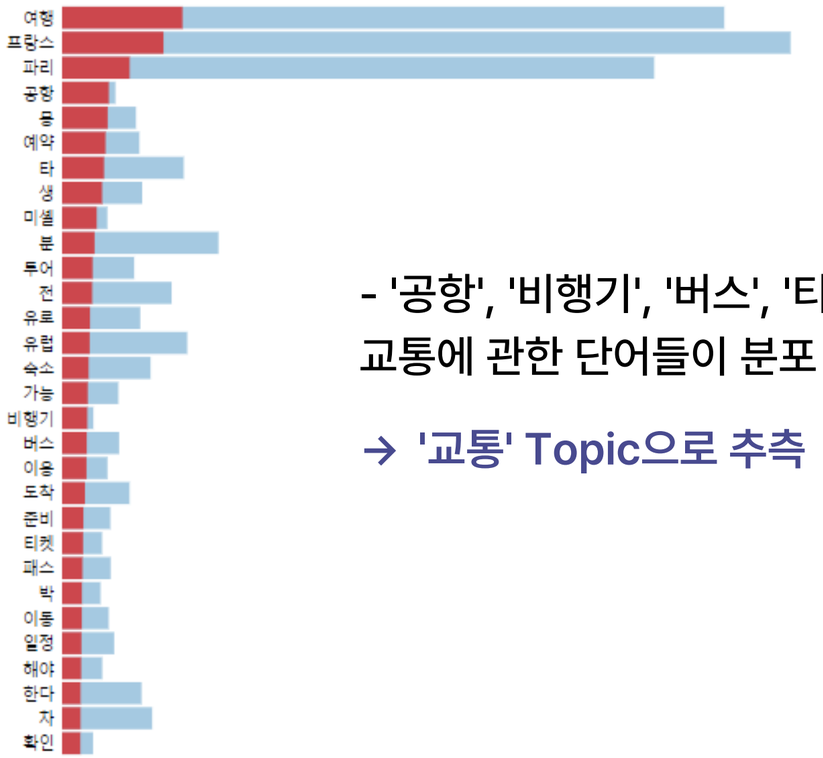
- Topic3



- '추천', '방문', '꼭', '아름다운' 등 여행 후기에 관한 단어들이 분포

→ '여행 후기' Topic으로 추측

- Topic4



- '공항', '비행기', '버스', '티켓' 등 교통에 관한 단어들이 분포

→ '교통' Topic으로 추측

03 MODELING - Topic Modeling (LDA)

Conclusion

- 여행 블로그 글은 대부분 브이로그, 유명소, 여행 준비&팁, 음식점, 교통 등의 Topic으로 나눌 수 있음

- 모든 나라에서 '여행 브이로그' Topic이 가장 큰 비중을 차지하는 것으로 나타남

- 각 나라별로 대체적으로 Topic이 유사하게 나누어지는 것으로 보아, 나라에 따라 뚜렷한 Topic이 나타나기보다는 '여행'이라는 키워드에서 Topic이 나누어짐

- 같은 Topic이어도, 각 나라별로 특징에 맞게 단어 분포가 다르게 나타남