

텍스트데이터분석

AI빅데이터융합경영학과

20212570 이수인

DATA

- 01 수집
- 02 전처리
- 03 분석
- 04 결론

#미국 #여행 #네이버 #네이버쇼핑
#네이버지도 #네이버블로그 #네이버클라우드

#전처리 함수 #postaging 함수

#wordcloud #LDA

#시각화 #결과 해석

01 텍스트 데이터 수집: 키워드



네이버

01 텍스트 데이터 수집: 방법

<데이터 수집 방법>

1. 네이버 검색 API 활용

-> 키워드에 대한 검색 결과 생성

2. BeautifulSoup 활용하여 task 수행

-> 각 검색 결과에 해당하는 네이버 뉴스 링크 접속

-> BeautifulSoup 활용하여 제목, 링크, 본문 추출

<SOURCE>

(1) 네이버 블로그

(2) 네이버 뉴스

01 텍스트 데이터 수집: 키워드 선정

#미국 - 살아보고 싶은 국가

#여행 - 인생의 값진 경험

#네이버 - 가장 관심있는 기업

#네이버 블로그

#네이버 쇼핑

#네이버 지도

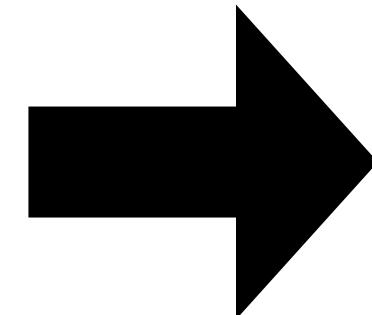
#네이버 클라우드

삶에 가장 밀접한
네이버 SERVICE

01 텍스트 데이터 수집: 파일 구성 1

<스크래핑>

뉴스와 블로그로 분류하여
스크래핑 진행



- 텍스트 데이터 수집 (뉴스).ipynb
- 텍스트 데이터 수집 (블로그).ipynb

01 텍스트 데이터 수집: 파일 구성 2

#미국 - 1042개(뉴스), 1094개(블로그)

#여행 - 991개(뉴스) , 1098개(블로그)

#네이버 - 815개(뉴스), 1944개(블로그)

#네이버 블로그 - 453개(뉴스), 975개(블로그)

#네이버 쇼핑 - 300개(뉴스), 1000개(블로그)

#네이버 지도 - 508개(뉴스), 924개(블로그)

#네이버 클라우드 - 376개(뉴스), 927개(블로그)

Total
12,447개

01 텍스트 데이터 수집: API 요청

주제: 미국 - 네이버블로그

```
In [5]: query = '미국'
display = 100
start = 1000
sort = 'sim'

result_USA = []
for i in range(0,10):
    start = 1 + 100*i # (1)
    result_usa = getresult(client_id, client_secret, query, display, start, sort) # (2)
    result_USA.append(result_usa)

result_USA
```

```
Out[5]: [[{'title': '<b>미국</b> 여행 준비를 LA 공항 입국심사 질문 서류 백신',
'link': 'https://blog.naver.com/pinkyluv83/223115767295',
'description': '2월 <b>미국</b> 서부 여행을 다녀 왔는데, 그 당시만 해도 코로나 백신 2차 접종이 의무였는데 이제는 백신 && 코로나 관련 서류 아무것도 필요 없이 <b>미국</b> 입국이 가능해졌어요! 라스베가스 문수쇼 그럼 먼저 <b>미국</b> 여행... ',
'bloggername': 'Kimi의 여행일기',
'bloggerlink': 'blog.naver.com/pinkyluv83',
'postdate': '20230530'},
{'title': '괌여행 오버체류 이스타 거절 후 <b>미국</b>관광비자 합격사례',
'link': 'https://blog.naver.com/jitaesera/223116527761',
'description': '그래서 오늘은 괌에서 오버체류 하게 된 후 <b>미국</b>관광비자 신청해서 합격 된 사례를 안내해 드리려고... 그런데 최근에 없다라고 체크를 해도 승인 거절이 나오는 경우가 있었고 만약 그럴 경우 <b>미국</b>관광비자... ',
'bloggername': '문홍구름♥',
'bloggerlink': 'blog.naver.com/jitaesera',
'postdate': '20230531'},
{'title': '<b>미국</b>달러 SOFR 초단기채권 ETF (Kodex, Tiger, Ace, KBStar)',
'link': 'https://blog.naver.com/1_stepmore/223115123055',
'description': '있는 <b>미국</b>달러SOFR(초단기채권) ETF 4종을 비교해보자 1. SOFR 이란? 2. 국내 ETF 4종 비교 3. 현명한 초단기채권 투자에 한걸음더 1. SOFR란? - 1일 초단기상품 SOFR(Secured Overnight Financing Rate)란 <b>미국</b> 국채를... ',
'bloggername': '매일매일 한걸음더',
```

query: 검색어를 '미국'으로 지정

display: 한 번에 100개의 검색 결과 개수

start: 검색 결과의 시작 위치

sort='sim': 유사도에 따라 정렬

(1) 검색 시작 위치 계산

(2) 검색 결과를 가져오는 함수 호출

01 텍스트 데이터 수집: Chat GPT 활용 1

수인

네이버 API로 데이터를 1,000개 이상 불러오고 싶어. 근데 100개 이상만 불러와도 예러가 떠. 어떻게 해결할까?



네이버 API로 데이터를 1,000개 이상 불러올 때 발생하는 문제는 API의 한도나 제한으로 인해 발생할 수 있습니다. 이러한 경우에는 몇 가지 해결 방법이 있습니다:

1. 페이징 기능 활용: 네이버 API에서는 페이징 기능을 제공하는 경우가 많습니다. 데이터를 여러 번에 나누어 요청하고, 각 요청에서 페이지 번호를 변경하여 데이터를 받아올 수 있습니다. 예를 들어, 한 번에 100개씩 데이터를 요청하고 페이지 번호를 조정하여 10번의 요청을 통해 1,000개의 데이터를 받을 수 있습니다.
2. 다른 필터링 옵션 사용: API 요청 시 다른 필터링 옵션을 사용하여 데이터 양을 줄일 수도 있습니다. 예를 들어, 특정 기간 내의 데이터만 요청하거나, 특정 카테고리 또는 키워드와 관련된 데이터만 요청하는 등의 방법을 사용하여 데이터 양을 제한할 수 있습니다.
3. API 제한 해제 요청: 만약 API 요청 제한이 너무 엄격하다고 생각되는 경우, 네이버에 API 제한 해제를 요청할 수 있습니다. 이는 네이버 개발자 센터나 관련 문서를 참고하여 진행할 수 있습니다. 제한 해제 요청이 승인되면 더 많은 데이터를 요청할 수 있을 수 있습니다.

위의 방법 중 하나를 선택하여 문제를 해결해보시기 바랍니다. 각 API에 따라 제공되는 기능과 제한은 다를 수 있으므로 해당 API의 문서를 참고하여 정확한 방법을 확인하시는 것이 좋습니다.

한번에 데이터 500개 이상을 불러오고
싶어서 챗지피티에게 물어봄

-> 페이징 기능 활용에 대해 자세히 알아
보았고 아래의 유용한 사이트를 발견

<https://yenpa.tistory.com/3>

API

[네이버 개발자센터 API] 검색 블로그 데이터 가져오기(파이썬)

01 텍스트 데이터 수집: getresult 함수

```
def getresult(client_id,client_secret,query,display=100,start=1,sort='sim'):
    encText = urllib.parse.quote(query)
    url = "https://openapi.naver.com/v1/search/news?query=" + encText + "&
    "&display=" + str(display) + "&start=" + str(start) + "&sort=" + sort

    request = urllib.request.Request(url)
    request.add_header("X-Naver-Client-Id",client_id)
    request.add_header("X-Naver-Client-Secret",client_secret)
    response = urllib.request.urlopen(request)
    rescode = response.getcode()

    if(rescode==200):
        response_body = response.read()
        response_str = response_body.decode('utf-8')
        response_json = json.loads(response_str)
    else:
        print("Error Code:" + rescode)

    return response_json['items']
```

한번의 코드로 500개 이상을 반환하도록 구현
(하지만 이 코드도 한번에 1000개 까지만 가능)

뉴스 수집

[https://openapi.naver.com/v1/search/ne
ws?query=](https://openapi.naver.com/v1/search/news?query=)

참고: <https://yenpa.tistory.com/3>

01 텍스트 데이터 수집: slack 질의

이수인 오후 9:41

수인 과제 1 텍스트 데이터 수집 과정에서 네이버 API를 이용하여 데이터를 수집하고 있습니다. 하나의 주제로 2,000개 이상의 데이터를 가져오고 싶은데, 네이버 API는 한 번 호출에 최대 100개까지만 가능합니다. 다음 코드는 반복문과 getresult라는 함수를 사용하여 100개 이상의 데이터를 한번에 호출할 수 있도록 만든 코드입니다.

```
def getresult(client_id,client_secret,query,display=10,start=1,sort='sim'):
    encText = urllib.parse.quote(query)
    url = "https://openapi.naver.com/v1/search/blog?query=" + encText + \
    "&display=" + str(display) + "&start=" + str(start) + "&sort=" + sort

    request = urllib.request.Request(url)
    request.add_header("X-Naver-Client-Id",client_id)
    request.add_header("X-Naver-Client-Secret",client_secret)
    response = urllib.request.urlopen(request)
    rescode = response.getcode()

    if(rescode==200):
        response_body = response.read()
        response_json = json.loads(response_body)
    else:
        print("Error Code:" + rescode)

    return pd.DataFrame(response_json['items'])
```

```
query = '주제'
display=100
start=1
sort='sim'

result_all=pd.DataFrame()
for i in range(0,2):
    start= 1 + 100*i
    result= getresult(client_id,client_secret,query,display,start,sort)

    result_all=pd.concat([result_all,result])
```

display 100개를 두 번 가져와야 200개이고 start를 다른 포지션(첫번째는 start=1, 두번째는 start=101)으로 가져왔습니다. result_all에 각 for 문의 결과 result를 합쳤습니다.

하지만 for문을 (0, 10)으로 하여 최대 1,000개까지의 데이터를 불러올 수 있지만, 그 이상은 HTTP Error 400: Bad Request 에러가 나옵니다. 네이버 API에서 1,000개 이상의 데이터를 한번에 불러올 수 있는 방법을 아시는 분 댓글 부탁드립니다 :)

참고: <https://yenpa.tistory.com/3> (편집됨)

한번에 1000개만까지만 불러오는게 아쉬웠음

-> Slack에 질의

하지만 어렵기도 하고 1000개로도

목표했던 양의 데이터를 모을 수 있을 것으로

예상하여 그대로 진행

01 텍스트 데이터 수집: HTML 태그 제거 및 변환 함수

```
In [44]: def remove_tag(my_str):
    ## 태그를 지우는 함수
    p = re.compile('<(([^>]+)>)')
    return p.sub('', my_str)

def sub_html_special_char(my_str):
    ## 특수문자를 나타내는 &apos;, &quot;를 실제 특수문자로 변환
    p1 = re.compile('&lt;')
    p2 = re.compile('&gt;')
    p3 = re.compile('&amp;')
    p4 = re.compile('&apos;')
    p5 = re.compile('&quot;')

    result = p1.sub('<', my_str)
    result = p2.sub('>', result)
    result = p3.sub('&', result)
    result = p4.sub('\'', result)
    result = p5.sub('"', result)
    return result
```

데이터 수집에 필요한 과정을 함수로 지정

08. WebScraping(5)_student 실습 코드 참고

01 텍스트 데이터 수집: List Comprehension

```
In [6]: # 리스트 컴프리헨션을 사용하여 이중 리스트 풀기  
result_USA = [item for sublist in result_USA for item in sublist]  
  
print(result_USA)
```

```
[{'title': '"하락세 끝났나" <b>미국</b> 집값 2개월 연속 반등', 'originalLink': 'http://www.newsis.com/view/?id=NISX20230531_0002321715&cID=10101&pID=10100', 'link': 'https://n.news.naver.com/mnews/article/003/0011887679?sid=104', 'description': '<b>미국</b>의 집값이 두 달 연속 반등한 것으로 나타났다. 30일(현지시간) 글로벌 시장지수 제공업체인 S&P 다우존슨 인덱스는 지난 3월 S&P 코어로직 케이스-실러 주택가격지수가 전월대비 0.4% 상승했다고 발표했다. 7개월... ', 'pubDate': 'Wed, 31 May 2023 05:16:00 +0900'}, {'title': '<b>미국</b> 3월 집값, 두 달 연속 반등…S&P &quot;하락세 끝났을지도&quot;', 'originalLink': 'https://www.yna.co.kr/view/AKR20230531002400072?input=1195m', 'link': 'https://n.news.naver.com/mnews/article/001/0013972737?sid=104', 'description': '매물 공급 부족 탓…연준 통화 긴축 지속시 매수 다시 꺾일 수도 강건택 특파원 = <b>미국</b>의 집값이 두 달 연속... 직전까지 <b>미국</b>의 집값은 7개월 연속 하락 중이었다. 3월 10개 주요 도시 주택가격지수와 20개 주요 도시...', 'pubDate': 'Wed, 31 May 2023 03:09:00 +0900'}, {'title': '미래에셋, <b>미국</b> 채권 투자 TIGER ETF 2종 신규 상장', 'originalLink': 'http://news.mt.co.kr/mtview.php?no=2023053109580491727', 'link': 'https://n.news.naver.com/mnews/article/008/0004893715?sid=101', 'description': '미래에셋자산운용은 31일 한국거래소에 &apos;TIGER <b>미국</b>채30년스트립액티브(합성H)&apos;, &apos;TIGER <b>미국</b>투자등급회사채액티브(H)&apos; ETF(상장지수펀드) 2종을 신규 상장한다고 밝혔다. TIGER <b>미국</b>채30년스트립액티브(합성H)는...', 'pubDate': 'Wed, 31 May 2023 10:00:00 +0900'}]
```

```
In [7]: len(result_USA)
```

```
Out[7]: 1000
```

수집한 데이터의 형태가
이중 리스트로 되어 있어서
푸는 작업 진행

-> 이후, 데이터의 길이는 1000개

01 텍스트 데이터 수집: 에러

```
:titles = []
links = []
descriptions = []

p = re.compile('https://blog.naver.com/.+')
for item in result_NAVER_1:
    if p.match(item['link']): ## <link>태그의 문자열이 blog.naver.com으로 시작하는 결과만 추출
        title = sub_html_special_char(remove_tag(item['title']))
        link = item['link']
        description = item['description']
        links.append(link)

        html = requests.get(link)
        bs_obj01 = BeautifulSoup(html.text, 'html.parser')
        src_url = "https://blog.naver.com/" + bs_obj01iframe["src"]

        html = requests.get(src_url)
        bs_obj02 = BeautifulSoup(html.text, 'html.parser')
        title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
        title = title[0].text.replace('\n', '')
        text = bs_obj02.find_all('div', {'class': 'se-main-container'})
        text = text[0].text.replace('\n', '')

        titles.append(title)
        descriptions.append(text)

        time.sleep(1)
```

```
IndexError                                                 Traceback (most recent call last)
Cell In[19], line 20
 18 bs_obj02 = BeautifulSoup(html.text, 'html.parser')
 19 title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
--> 20 title = title[0].text.replace('\n', '')
 21 text = bs_obj02.find_all('div', {'class': 'se-main-container'})
 22 text = text[0].text.replace('\n', '')

IndexError: list index out of range
```

‘네이버’ 키워드로

블로그 데이터 수집 중

다음과 같은 오류 발생

10. WebScraping_Practice_answer_배지환님 코드 참고

01 텍스트 데이터 수집: Chat GPT 활용 2

수인

텍스트 데이터 분석 시 발생한 오류야.
'네이버'라는 주제로 네이버 API에서 데이터를 수집하는 과정에서 인덱스 오류가 발생했어.
다음은 에러 메시지야.

IndexError

Cell In[19], line 20

```
18 bs_obj02 = BeautifulSoup(html.text, 'html.parser')
19 title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
---> 20 title = title[0].text.replace('\n','')
21 text = bs_obj02.find_all('div', {'class':'se-main-container'})
22 text = text[0].text.replace('\n','')
```

Traceback (most recent call last)

IndexError: list index out of range

이 오류를 해결하는 방법을 제시해줘.

01 텍스트 데이터 수집: Chat GPT 활용 2



인덱스 오류가 발생한 이유는 데이터 수집 과정에서 원하는 요소를 찾지 못하여 리스트가 비어 있는 상태에서 인덱스에 접근하려고 하기 때문입니다. 이를 해결하기 위해 다음과 같은 접근 방법을 고려할 수 있습니다:

- 데이터 유효성 확인: 수집한 데이터에 원하는 요소가 실제로 존재하는지 확인해야 합니다. 데이터가 제대로 수집되었는지 API 응답을 확인하고, HTML 구조를 검토하여 해당 요소가 존재하는지 확인해보세요.
- 요소의 존재 여부 확인: `find_all()` 메서드로 요소를 찾을 때, 반환되는 리스트의 길이를 먼저 확인하고, 요소가 존재하는지 확인해야 합니다. 이를 위해 `if` 문을 사용하여 리스트의 길이를 검사할 수 있습니다. 예를 들어, 다음과 같이 수정해 볼 수 있습니다:

```
python
title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title'})
if len(title) > 0:
    title = title[0].text.replace('\n', '')
else:
    title = "No title found" # 원하는 요소가 없을 경우 대체값을 설정합니다.
```

- 예외 처리: 원하는 요소가 없을 때 발생하는 예외를 처리할 수도 있습니다. `try-except` 문을 사용하여 `IndexError` 예외를 처리하고, 예외가 발생할 경우에 대한 대체 동작을 설정 할 수 있습니다. 예를 들어:

```
python
try:
    title = title[0].text.replace('\n', '')
except IndexError:
    title = "No title found" # 원하는 요소가 없을 경우 대체값을 설정합니다.
```

이러한 접근 방법을 사용하여 원하는 요소를 찾지 못했을 때 발생하는 인덱스 오류를 처리할 수 있습니다. 하지만 데이터의 구조와 API 응답의 내용을 정확히 파악해야 하므로, 해당 부분에 대한 상세한 정보를 제공해주시면 더 구체적인 도움을 드릴 수 있습니다.

01 텍스트 데이터 수집: Chat GPT 활용 2

```
In [20]: titles = []
links = []
descriptions = []

p = re.compile('https://blog.naver.com/.+')
for item in result_NAVER:
    if p.match(item['link']): ## <link>태그의 문자열이 blog.naver.com/으로 시작하는 결과만 추출
        title = sub_html_special_char(remove_tag(item['title']))
        link = item['link']
        description = item['description']
        links.append(link)

        html = requests.get(link)
        bs_obj01 = BeautifulSoup(html.text, 'html.parser')
        src_url = "https://blog.naver.com/" + bs_obj01iframe["src"]

        html = requests.get(src_url)
        bs_obj02 = BeautifulSoup(html.text, 'html.parser')

        title_elements = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
        if title_elements:
            title = title_elements[0].text.replace('\n', '')
        else:
            title = "No title found"

        text_elements = bs_obj02.find_all('div', {'class': 'se-main-container'})
        if text_elements:
            text = text_elements[0].text.replace('\n', '')
        else:
            text = "No text found"

        # title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
        # title = title[0].text.replace('\n', '')
        # text = bs_obj02.find_all('div', {'class': 'se-main-container'})
        # text = text[0].text.replace('\n', '')

        titles.append(title)
        descriptions.append(text)

        time.sleep(1)
```

Chatgpt가 제시해준 방안 중
<3.예외처리>를 통해 오류 해결

01 텍스트 데이터 수집: 각 항목 처리(블로그)

```
In [20]: titles = []
links = []
descriptions = []

p = re.compile('https://blog.naver.com/.+') # (1)
for item in result_NAVER:
    if p.match(item['link']): ## <link>태그의 문자열이 blog.naver.com으로 시작하는 결과만 추출
        title = sub_html_special_char(remove_tag(item['title'])) # (2)
        link = item['link']
        description = item['description']
        links.append(link)

        html = requests.get(link)
        bs_obj01 = BeautifulSoup(html.text, 'html.parser')
        src_url = "https://blog.naver.com/" + bs_obj01iframe["src"] # (3)

        # 블로그 내용 파싱을 위해 src_url을 생성하고 요청을 보냄
        html = requests.get(src_url)
        bs_obj02 = BeautifulSoup(html.text, 'html.parser')

        # 제목 파싱
        title_elements = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
        if title_elements:
            title = title_elements[0].text.replace('\n', '')
        else:
            title = "No title found"

        # 본문 내용 파싱
        text_elements = bs_obj02.find_all('div', {'class': 'se-main-container'})
        if text_elements:
            text = text_elements[0].text.replace('\n', '')
        else:
            text = "No text found"

        # title = bs_obj02.find_all('div', {'class': 'se-module se-module-text se-title-text'})
        # title = title[0].text.replace('\n', '')
        # text = bs_obj02.find_all('div', {'class': 'se-main-container'})
        # text = text[0].text.replace('\n', '')

        titles.append(title)
        descriptions.append(text)

        time.sleep(1)
```

(1) p에 맞는 링크를 찾기 위한
정규 표현식 패턴 컴파일

(2) 'title'에서 HTML 태그와
특수 문자를 제거한 결과를 저장

(3) iframe 태그의 src URL 생성

01 텍스트 데이터 수집: 각 항목 처리(뉴스)

```
In [8]: p = re.compile('https://n.news.naver.com/.+')
links = []
for item in result_USA:
    link = item['link']
    if p.search(link):
        links.append(link)
```

```
In [9]: len(links)
```

```
Out[9]: 648
```

수집한 뉴스 데이터 중,

p에 맞는 링크를 찾기 위한 정규 표현식 패턴 컴파일

08. WebScraping(5)_student 실습 코드 참고

01 텍스트 데이터 수집: 각 항목 처리(뉴스)

```
In [10]: article_ids = ['newsct_article', 'articeBody']
title_id = 'title_area'
```

```
In [11]: titles = []
descriptions = []
```

```
# links 리스트의 각 링크를 순회하면서 웹 페이지를 가져와서 제목과 설명을 추출
for i, link in enumerate(links):
    html = urllib.request.urlopen(link) # (1)
    bs_obj = BeautifulSoup(html, 'html.parser') # (2)

    # 제목 추출
    title = bs_obj.find_all('h2', {'id':title_id}) # (3)
    if len(title) > 0:
        titles.append(title[0].text)
    else:
        titles.append('')

    # 설명 추출
    description = bs_obj.find_all('div', {'id':article_ids}) # (4)
    if len(description) > 0:
        descriptions.append(description[0].text)
    else:
        descriptions.append('')
    time.sleep(1)
```

```
In [12]: len(descriptions)
```

```
Out[12]: 648
```

(1) link에 지정된 URL로
HTTP 요청을 보내고 응답을 받아옴

(2) 받아온 HTML을
BeautifulSoup 객체로 파싱

(3) h2 태그 중에서 title_id에
해당하는 속성을 가진 요소를 찾음

(4) div 태그 중에서 article_ids에
해당하는 속성을 가진 요소를 찾음

08. WebScraping(5)_student 실습 코드 참고

01 텍스트 데이터 수집: 데이터프레임 저장

```
In [9]: result_dict_USA = {'title': titles, 'link': links, 'description': descriptions}  
df_USA = pd.DataFrame.from_dict(result_dict_USA)
```

```
In [10]: df_USA
```

```
# CSV 파일로 저장  
df_USA.to_csv('미국(블로그)1-988row.csv', index=False)
```

```
# CSV 파일로 저장  
df_USA.to_csv('미국(블로그)2-987row.csv', index=False)
```

```
# CSV 파일로 저장  
df_USA.to_csv('미국(블로그)3-989row.csv', index=False)
```

```
# CSV 파일로 저장  
df_USA.to_csv('미국(블로그)4-984row.csv', index=False)
```

title, link, description 세 가지 컬럼으로
데이터프레임 생성

데이터프레임으로 만들어 to_CSV

이런식으로 며칠 간 진행

주제가 '미국'인 블로그 데이터의 파일
총 4개 생성

나머지도 동일하게 진행

01 텍스트 데이터 수집: 최종 데이터프레임 저장

```
In [5]: df_USA_1 = pd.read_csv('미국(블로그)1-988row.csv')
df_USA_2 = pd.read_csv('미국(블로그)2-987row.csv')
df_USA_3 = pd.read_csv('미국(블로그)3-989row.csv')
df_USA_4 = pd.read_csv('미국(블로그)4-984row.csv')

In [7]: merged_df_USA = pd.merge(df_USA_1, df_USA_2, on=['title', 'link', 'description'], how='outer')
df_USA_12 = merged_df_USA.drop_duplicates()

In [8]: merged_df_USA = pd.merge(df_USA_3, df_USA_4, on=['title', 'link', 'description'], how='outer')
df_USA_34 = merged_df_USA.drop_duplicates()

In [9]: merged_df_USA = pd.merge(df_USA_12, df_USA_34, on=['title', 'link', 'description'], how='outer')
df_USA = merged_df_USA.drop_duplicates()

In [10]: df_USA

...
# CSV 파일로 저장
df_USA.to_csv('미국(블로그)-1094.csv', index=False)
```

파일 하나 당

블로그의 경우 약 900개,

뉴스의 경우 약 500개가 수집 됨

4개의 파일을 다시 불러와 중복 제거를 진행

최종적으로 남은 블로그데이터의 개수는 1,094개

나머지 주제에 대해서도 동일하게 진행

단, 서브 주제인 네이버 쇼핑, 네이버 블로그,

네이버 지도, 네이버 클라우드는

비교적 적게 수집함

02 텍스트 데이터 전처리: 방향

1. 미국, 여행, 네이버와 네이버 관련 서비스들을 분리하여 진행

2. 미국, 여행, 네이버

-> wordcloud 생성을 목적으로 전처리 진행

3. 네이버 지도, 네이버 쇼핑, 네이버 클라우드, 네이버 블로그

-> LDA 분석을 목적으로 전처리 진행

4. 먼저 wordcloud 생성을 위한 텍스트 데이터 전처리 설명

02 텍스트 데이터 전처리: 데이터 불러오기

ver. wordcloud

데이터 불러오기

```
In [2]: USA_news = pd.read_csv('미국(뉴스)-1042row.csv')  
USA_blog = pd.read_csv('미국(블로그)-1094row.csv')
```

```
In [3]: USA_news
```

```
Out[3]:
```

	title	link	description
0	'미국 주도 중국 견제' IPEF 공급망 협정 타결 https://n.news.naver.com/mnews/article/056/001...		\n\n[영어] 중국을 견제하기 위해 미국 주도로 출범한 다자 경제 협력체 IPE...
1	"디플트 없다"... 미국 부채한도 인상안 최종 타결 https://n.news.naver.com/mnews/article/469/000...		\n\n바이든·매카시 하원의장, 인상안 최종 합의31일 의회 처리, 양당 강경파 설...

본문 합치기

```
In [4]: df_news = USA_news['description'].astype(str).str.cat(sep=' ')
```

```
In [5]: df_blog = USA_blog['description'].astype(str).str.cat(sep=' ')
```

```
In [6]: df_news
```

```
Out[6]: '\n\n[앵커] 중국을 견제하기 위해 미국 주도로 출범한 다자 경제 협력체 IPEF가 출범 1년 만에 첫 공급망 협정을 타결했습니다. 중국을 배제한 아시아 태평양 지역의 경제 질서를 세우겠다는 건데, 중국은 자국의 발전을 막기 위한 행위라며 반발했습니다. 정마연 기자가 보도합니다. [리포트] 미국 디트로이트에서 열린 IPEF 회의에서 14개국 통상 장관들이 공급망 협정 타결을 선언했습니다. 공급망과 관련한 미국 주도의 첫 다자 합의입니다. 협정에 따라 한국과 미국, 일본 등 14개 참여국들은 공급망 혼란 시 대응 네트워크를 가동해 대체 공급처와 운송 경로를 찾고 신속한 통관을 진행하는 등 공조하게 됩니다. 중국이 첨단 산업 핵심 소재인 리튬과 희토류 등의 공급망을 장악하는 상황에서 자원 무기화에 대비해 공급망을 다변화한다는 취지입니다. [캐서린 타이/미국 무역대표부 대표 : "국제 경제 현실에 적응해야 하는 것은 당연한 일입니다. 그 전략으로 미국은 미래 경쟁력을 확보하는 것입니다. 그것이 우리가 미래에 대한 확신입니다."]
```

데이터의 description 열을

하나의 string으로 합쳐 진행

02 텍스트 데이터 전처리: preprocessing 함수 ver. wordcloud

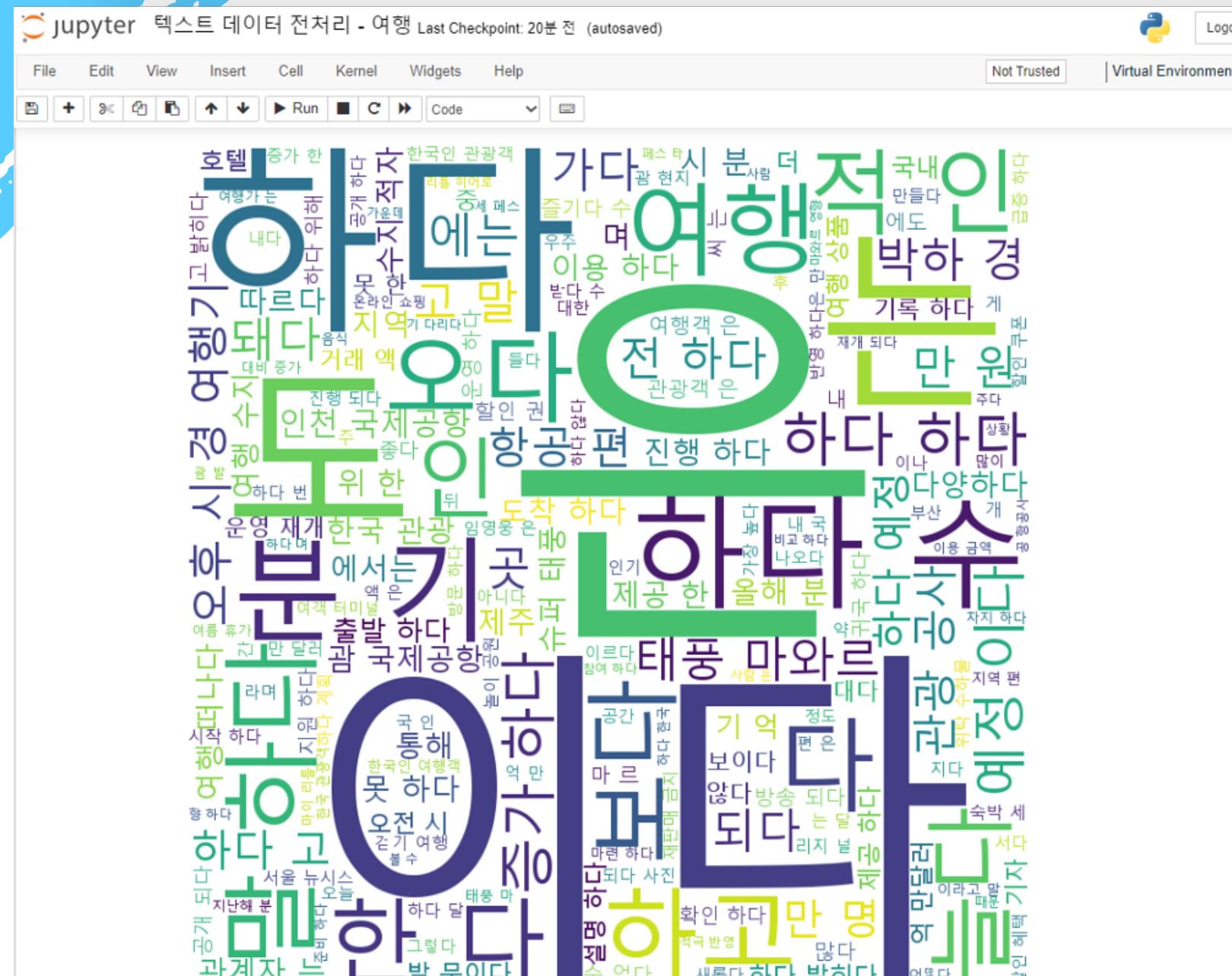
불용어 목록 생성

```
In [7]: with open('stopwords.txt', 'r', encoding='utf-8') as f:  
    stopwords = f.readlines()  
stopwords = [x.replace('\n', '') for x in stopwords] # stopword 파일의 줄바꿈 문자 제거
```

```
In [8]: def preprocessing(x):  
    p = re.compile('[ㄱ-ㅎㅏ-ㅣㅏ-ㅓ]')  
    results = []  
    result = []  
    x = x.replace('\n', ' ') # (1)  
    df_news = spacer(x) # (3)  
  
    tokens = okt.morphs(x, stem=True) # (8)  
    for token in tokens:  
        token = p.sub('', token) # (5)  
        if token not in stopwords:  
            result.append(token) # (7)  
    results.extend(result)  
    results = ' '.join(results)  
    return results
```

- 전처리 함수
 - 6주차 01.WordCloud 실습코드 참고
- # (1) 줄바꿈 문자 제거
- # (2) 오탈자 교정 (생략)
- # (3) 띄어쓰기 교정
- # (4) sentence tokenization (너무 오래걸려서 생략)
- # (5) 특수문자 문자 제거
- # (6) word tokenization (형태소 분석 및 stemming)
- # (7) stopword removal

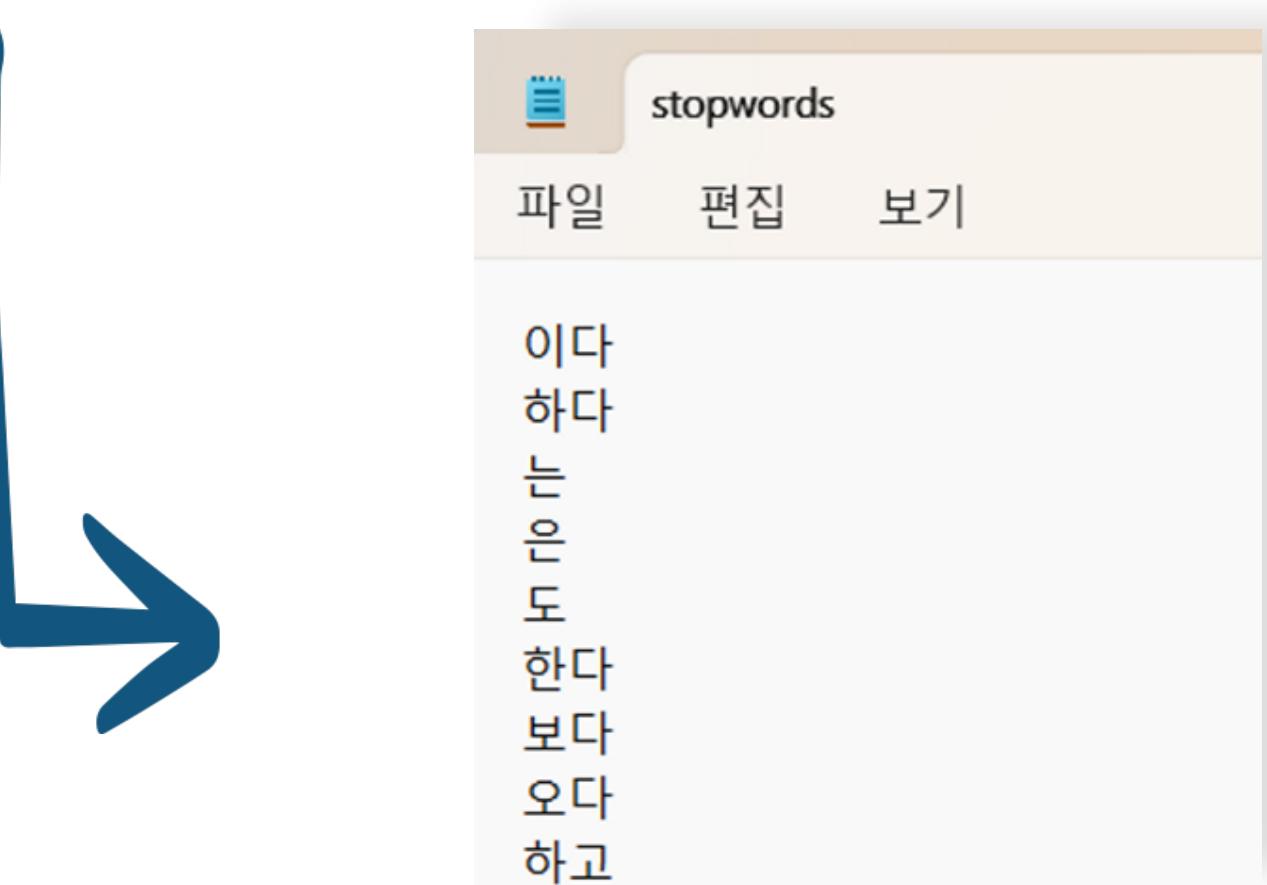
02 텍스트 데이터 전처리: pos_tagging 함수 ver. wordcloud



전처리 후 바로 wordcloud 생성했을때의 결과값

〈하다, 도, 은, 이다〉등의 의미없는 단어들만 구성됨을 볼 수 있음

-> 이를 해결하기 위해 stopwords에 항목을 추가



02 텍스트 데이터 전처리: pos_tagging 함수

ver. wordcloud



하지만 여전히 동사 및 조사로 추정되는 단어들이 존재

그래서 pos tagging을 진행

파일	편집	보기
stopwords		
이다		
하다		
는		
도		
한다		
보다		
보다		
하고		

02 텍스트 데이터 전처리: Chat GPT 활용 3

수인

okt.pos를 통해 얻은 결과값 중 ('을', 'Josa')여기에 해당하는 값을 okt.morphs여기로 통해 얻은 결과값 중에서 제거하고 싶어



Python의 KoNLPy 라이브러리에서 제공하는 Okt 클래스의 `pos` 메서드를 통해 얻은 결과에서 ('을', 'Josa')와 같이 특정 품사를 가지는 단어를 제거하려는 것이 맞나요? 만약 그렇다면, 아래의 코드를 참고하여 제거할 수 있습니다:

Okt.pos를 통해 얻은 결과 값 중,

원하는 품사를 제거하는 방법을 물어봄

python

```
from konlpy.tag import Okt

okt = Okt()

# 문장 예시
sentence = "책을 읽는 것은 지식을 쌓는 좋은 방법이다."

# okt.pos를 통해 얻은 결과
pos_result = okt.pos(sentence)

# okt.morphs를 통해 얻은 결과
morph_result = okt.morphs(sentence)

# 제거할 품사들
target_pos = ['Josa', 'Determiner', 'Punctuation']

# 특정 품사들을 가지는 단어 제거
filtered_pos_result = [(word, pos) for word, pos in pos_result if pos not in target_pos]
filtered_morph_result = [word for word in morph_result if okt.pos(word)[0][1] not in target_pos]

print(filtered_pos_result)
print(filtered_morph_result)
```

02 텍스트 데이터 전처리: pos_tagging 함수 ver. wordcloud

```
In [9]: def pos_taging(x):
    pos_news = okt.pos(x) # (1)
    filtered_result = [word for word, pos in pos_news if pos in ['Noun', 'Adjective', 'Adverb']] # (2)
    nouns_news = [n for n in filtered_result if len(n) > 1]
    count = Counter(nouns_news) # 각 단어의 출현 빈도를 계산
    tags = count.most_common(100) # 가장 빈도가 높은 단어 100개를 추출
    tags = tags[1:] # (3)
    return tags
```

(1) 품사 태깅 수행

(2) 명사, 형용사, 부사에 해당하는 단어들만 추출하여 저장

(3) 빈도가 가장 높은 단어는 주로 주제로 검색한 단어이기에

의미있는 결과를 위해 제거

* 챗지피티가 알려준 내용과

[https://financedata.github.io
/posts/python_news_cloud_
text.html](https://financedata.github.io/posts/python_news_cloud_text.html)

추가로 검색한 자료를 통해

pos_taging 함수 생성

02 텍스트 데이터 전처리: 함수 적용

ver. wordcloud

```
In [11]: blog = preprocessing(df_blog)
blog = pos_tagging(blog)
blog = [word for word in blog if not word[0].endswith('다')]
```

블로그의 경우

좋다, 없다 등이 포함된 경우가 많음

-> ~다로 끝나는 단어 제거

02 텍스트 데이터 전처리: pos_tagging 함수 ver. LDA

LDA는 본문을 하나의 string으로 합치지 않고 진행

reprocessing 함수는 동일하게 진행

```
In [6]: def pos_tagging(x):
    pos_news = okt.pos(x) # (1)
    filtered_result = [word for word, pos in pos_news if pos in 'Noun'] # (2)
    nouns_news = [n for n in filtered_result if len(n) > 1]
    return nouns_news
```

(1), # (2) 은 앞과 동일

LDA 분석을 위해, 단어 빈도수 계산은 따로 진행하지 않음

03 텍스트 데이터 분석: wordcloud

```
In [22]: blog
```

```
Out [22]: [('사진', 1571),  
           ('너무', 1512),  
           ('투어', 1391),  
           ('생각', 1353),  
           ('많이', 1347),  
           ('사람', 1160),  
           ('정도', 1092),  
           ('코스', 1070),  
           ('호텔', 1064),  
           ('숙소', 1062),  
           ('예약', 1061),  
           ('정말', 1051),  
           ('이용', 993),  
           ('일본', 963),  
           ('때문', 926),  
           ('일정', 912),  
           ('베트남', 901),  
           ('추천', 897),  
           ('제주', 883),  
           ('방문', 834),
```

```
In [21]: wc = WordCloud(font_path=font_path,
```

```
                           background_color='white',  
                           width=800, height=800)  
cloud = wc.generate_from_frequencies(dict(blog)) # (1)  
  
plt.figure(figsize=(8,8))  
plt.axis('off') # 축을 비활성화하여 표시되는 축 라벨과 눈금을 제거  
plt.imshow(cloud)
```

```
Out [21]: <matplotlib.image.AxesImage at 0x1c7cc570430>
```

다음과 같은 리스트 형태로 단어와 단어 빈도수를 만들어냄

#(1) 단어와 빈도수 정보를 사전 형태로 변환하여 WordCloud를 생성

03 텍스트 데이터 분석: LDA

LDA 진행

```
In [11]: word_dict_news = corpora.Dictionary(NAVERSHOP_news['preprocessed_description'])

In [12]: word_dict_blog = corpora.Dictionary(NAVERSHOP_blog['preprocessed_description'])

In [13]: corpus_news = [word_dict_news.doc2bow(text) for text in NAVERSHOP_news['preprocessed_description']]

In [14]: corpus_blog = [word_dict_blog.doc2bow(text) for text in NAVERSHOP_blog['preprocessed_description']]
```

(word_id, word_frequency) 형태로 변환

BOW(Bag-of-Words) 생성

02. Latent Dirichlet Allocation (LDA) 실습 코드 참고

03 텍스트 데이터 분석: LDA

```
In [47]: N_TOPICS = 3 # (1)
ldamodel_news = gensim.models.LdaModel(corpus_news, num_topics = N_TOPICS, id2word=word_dict_news, passes = 15) # (2)

topics = ldamodel_news.print_topics(num_words=4) # (3)
for topic in topics:
    print(topic)

(0, '0.030*"네이버" + 0.029*"클라우드" + 0.019*"서비스" + 0.011*"활용"')
(1, '0.042*"네이버" + 0.033*"클라우드" + 0.015*"기술" + 0.014*"서비스"')
(2, '0.042*"클라우드" + 0.025*"네이버" + 0.015*"디지털" + 0.013*"데이터"')
```

〈LDA모델 훈련〉

(1) 3개의 topic

(2) LDA 모델을 생성. ‘id2word’에는 단어 사전을 전달하고,

학습 반복 횟수는 15로 설정

(3) 생성된 모델에서 각 주제에 대한 상위 4개 단어를 출력

03 텍스트 데이터 분석: LDA

```
In [23]: import pyLDAvis.gensim_models  
  
pyLDAvis.enable_notebook() # (1)  
vis = pyLDAvis.gensim_models.prepare(ldamodel, corpus, word_dict) # (2)  
pyLDAvis.display(vis)
```

<LDA시각화>

(1) pyLDAvis 시각화 활성화

(2) 'ldamodel'은 생성된 LDA 모델을, 'corpus'는 BOW 형식의 텍스트 데이터를,

'word_dict'는 단어 사전을 전달

04 wordcloud 결과 해석: 미국



(1) 뉴스기사로부터 미국과 관련한 주요 키워드로 중국, 한국, 반도체, 정부, 기업 등이 있다.

-> 미국과 관련된 정치적인 이슈 및 기술적인 이슈를 많이 다룬다

(2) 블로그에는 미국 대학, 학생, 유학, 학생, 비자, 투자 등이 있다

-> 미국 학교 및 기업을 원하는 사람들을 위한 블로그 포스팅이 많다

04 wordcloud 결과 해석: 여행



(1) 뉴스기사로부터 여행과 관련한 주요 키워드는 관광, 여행객, 사진, 한국, 할인, 등이 있다.

-> 관광지 또는 여행객의 증감과 관련된 기사를 다룬다.

(2) 블로그에는 너무, 많이, 사진, 투어, 코스, 생각, 호텔, 숙소, 예약 등이 있다.

-> 코로나 이후, 여행을 가고 싶은 사람들의 감정이 담겼다.

그리고 숙소 예약 등의 정보를 다룬 포스팅이 많다.

04 wordcloud 결과 해석: 네이버



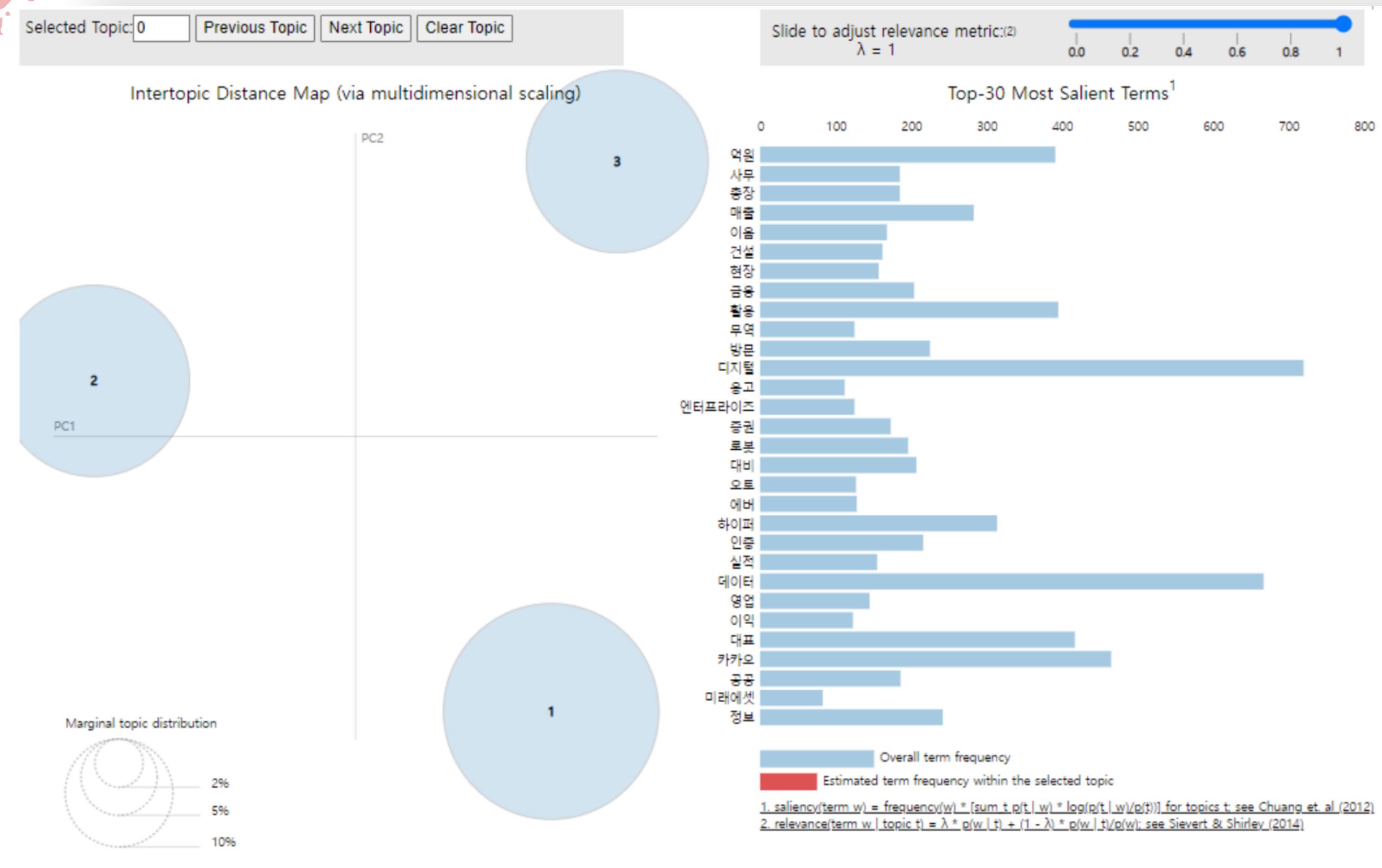
(1) 네이버와 관련하여 검색, 서비스, 중국, 접속, 등이 많이 눈에 띈다

-> 예상 외의 키워드가 떠올랐다. 최근에 네이버 중국 접속 차단 관련 논란이 영향을 준 것으로 추정한다.

(2) 블로그에는 블로그, 생각, 검색, 사용, 방법 등이 있다

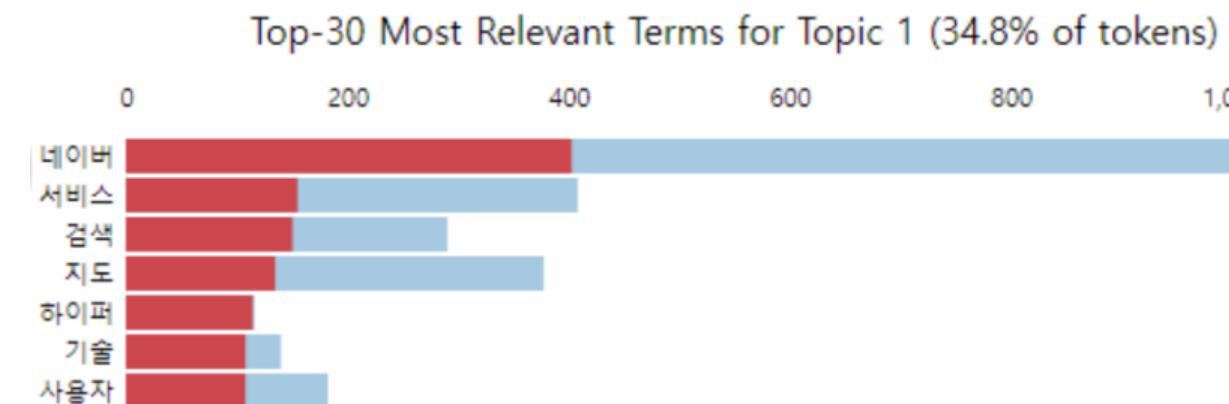
-> 예상대로 네이버에서 가장 많이 사용하는 검색 서비스와 블로그 서비스가 주를 이루었다.

04 LDA 시각화 결과

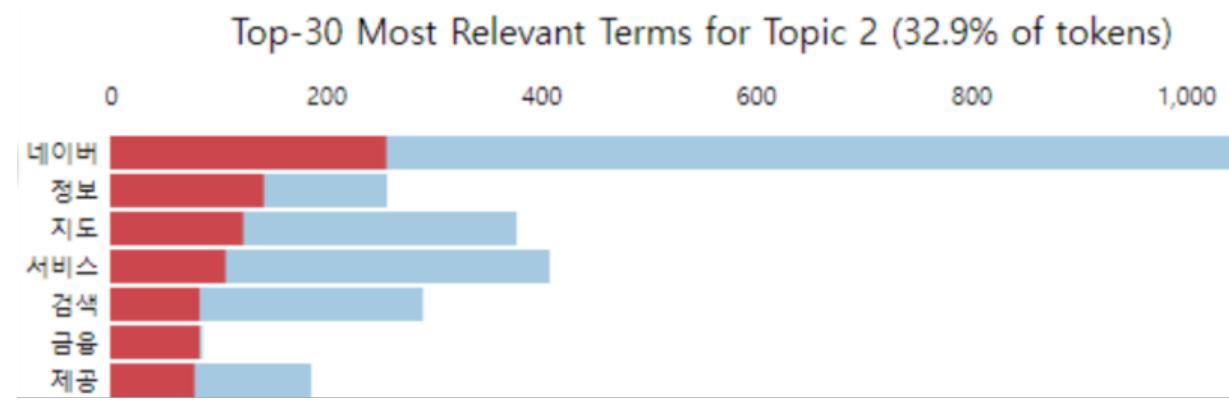


04 LDA 결과 해석: 네이버 지도(뉴스)

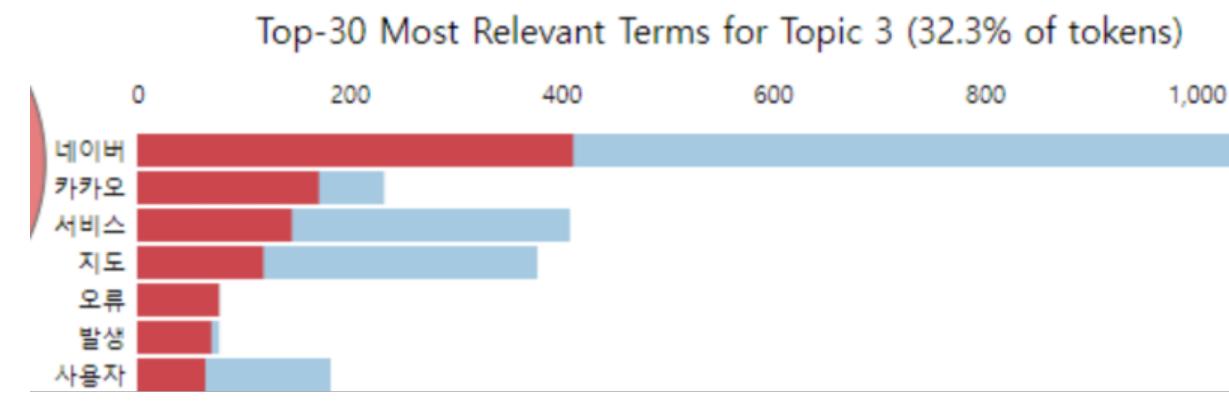
Topic 1: “네이버 지도 기술 서비스”



Topic 2: “네이버 지도 플랫폼”

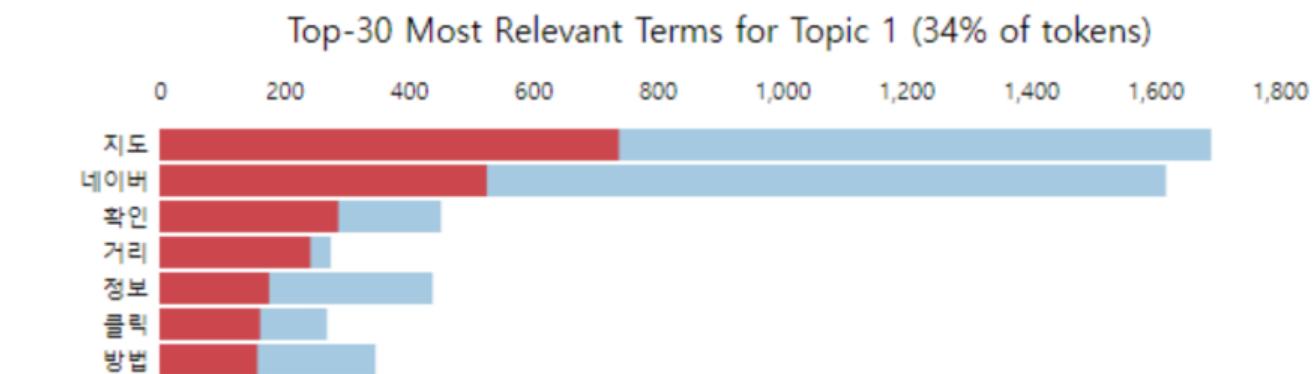


Topic 3: “네이버와 카카오 서비스 지도 오류”

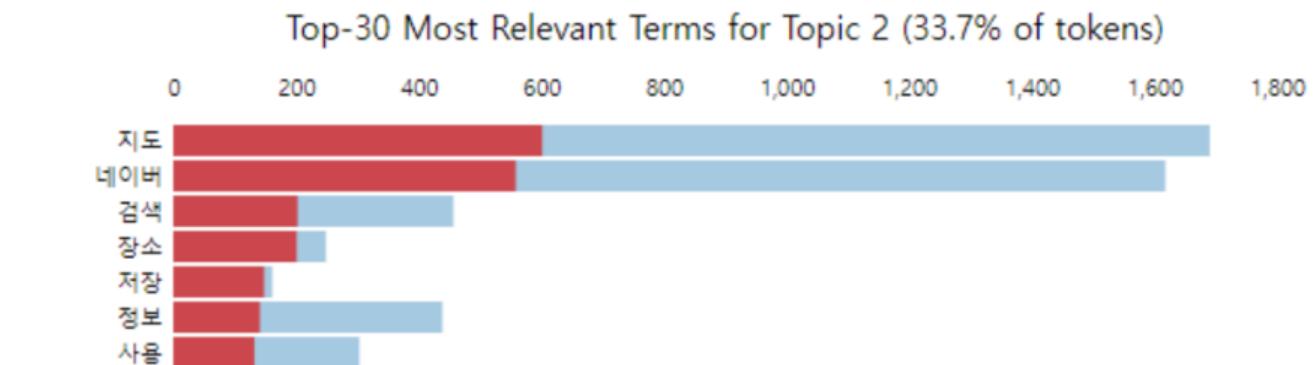


04 LDA 결과 해석: 네이버 지도(블로그)

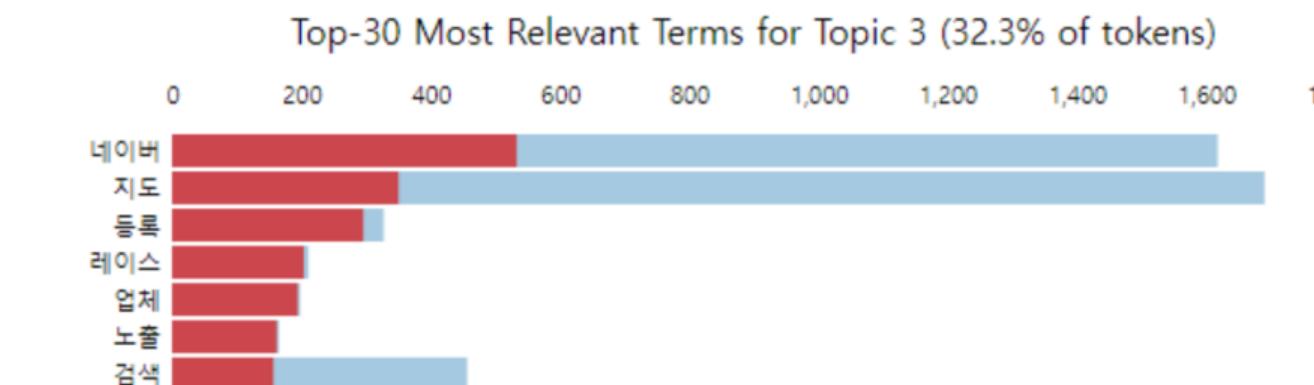
Topic 1: “네이버 지도 사용법”



Topic 2: “네이버 지도 후기”

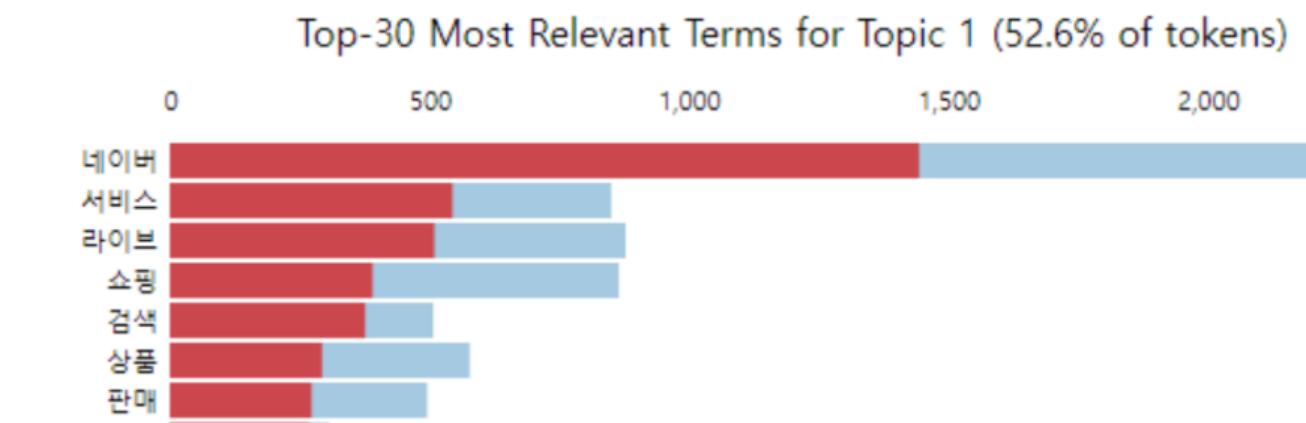


Topic 3: “네이버 지도 마케팅”

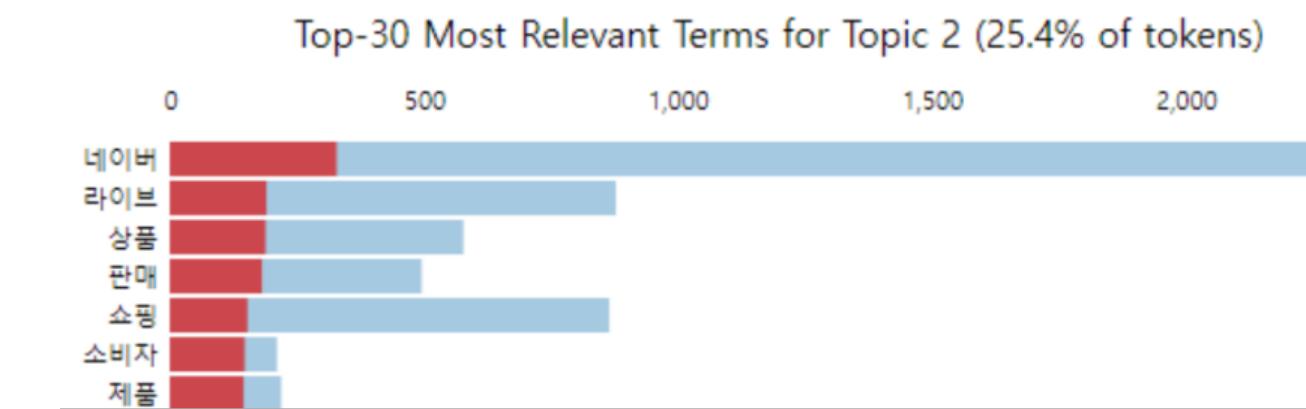


04 LDA 결과 해석: 네이버 쇼핑(뉴스)

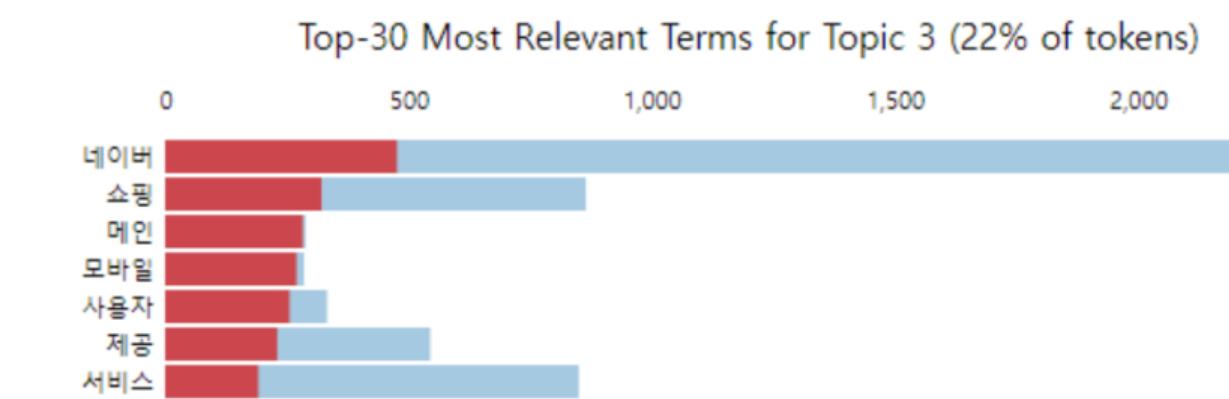
Topic 1: “네이버 쇼핑 커머스 성장“



Topic 2: “네이버 쇼핑 구매 할인 정보“

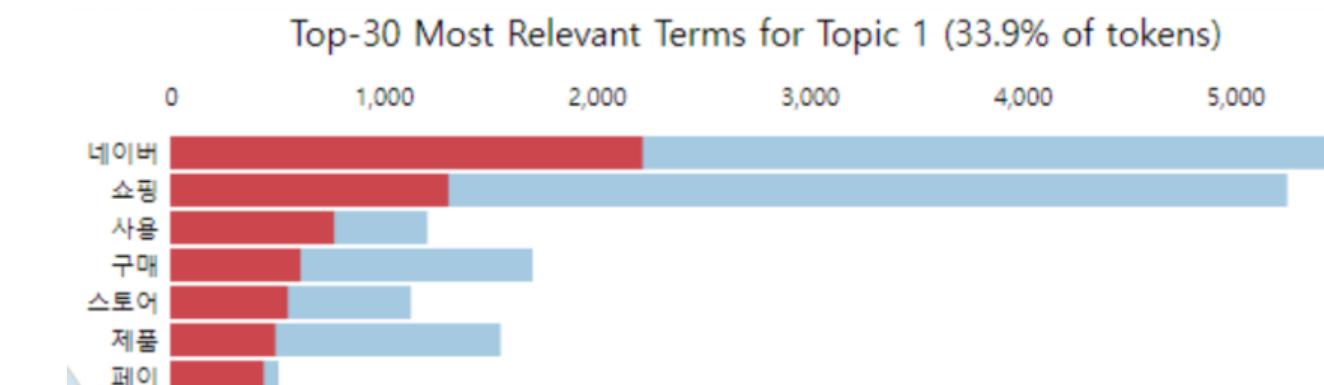


Topic 3: “네이버 쇼핑 모바일 버전“

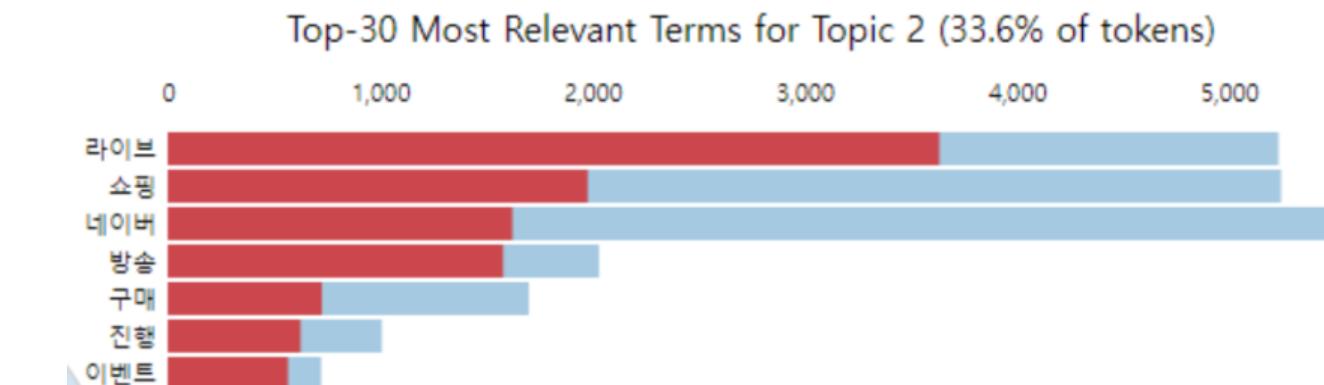


04 LDA 결과 해석: 네이버 쇼핑(블로그)

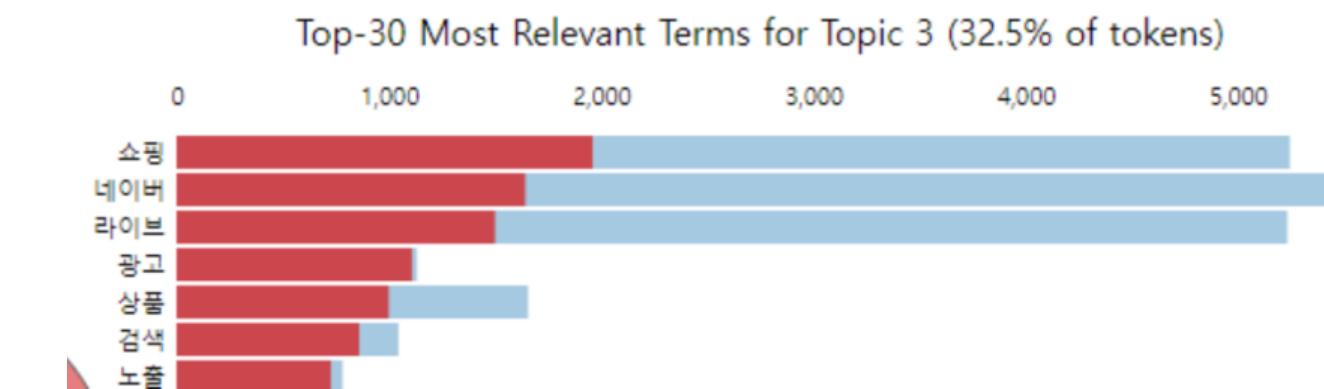
Topic 1: “네이버 페이 포인트 정보”



Topic 2: “네이버 쇼핑 이벤트 혜택”



Topic 3: “네이버 쇼핑 브랜드 마케팅”

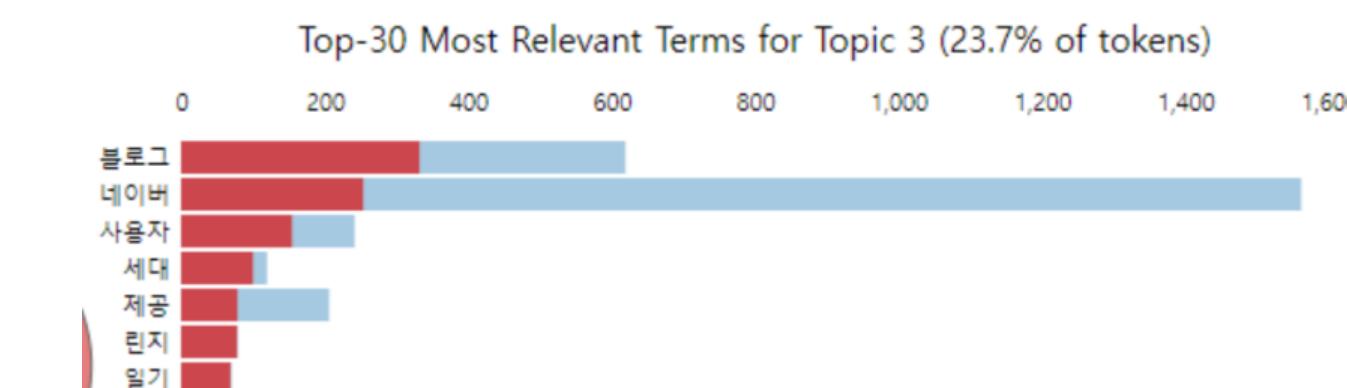
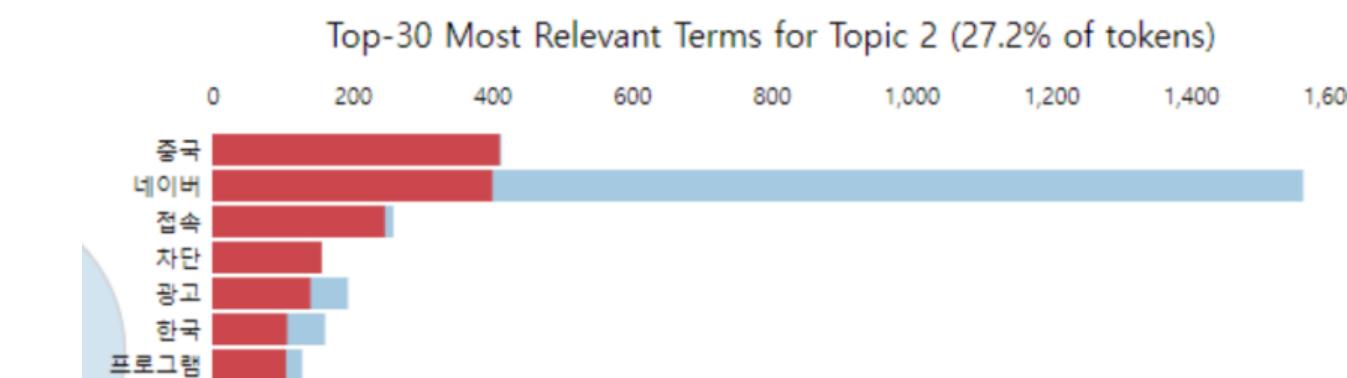
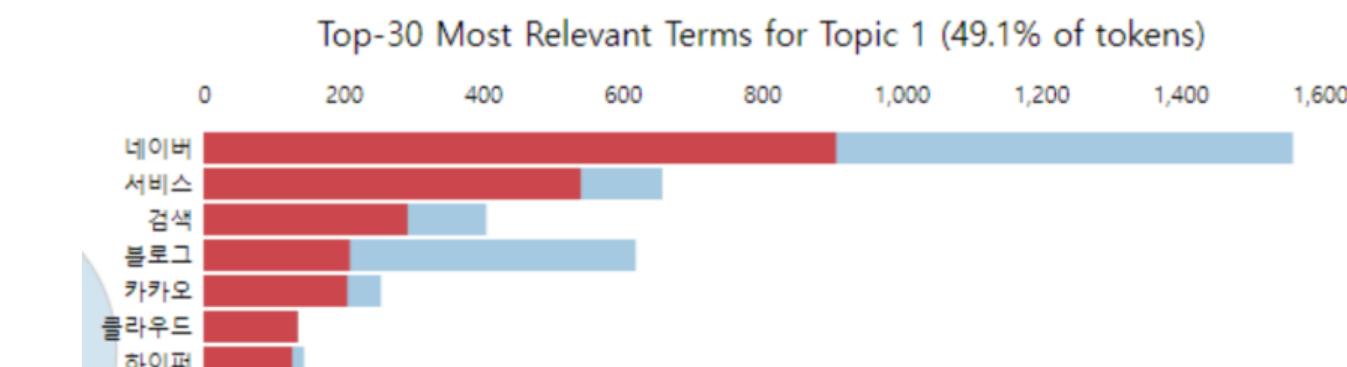


04 LDA 결과 해석: 네이버 블로그(뉴스)

Topic 1: “네이버 블로그 시장”

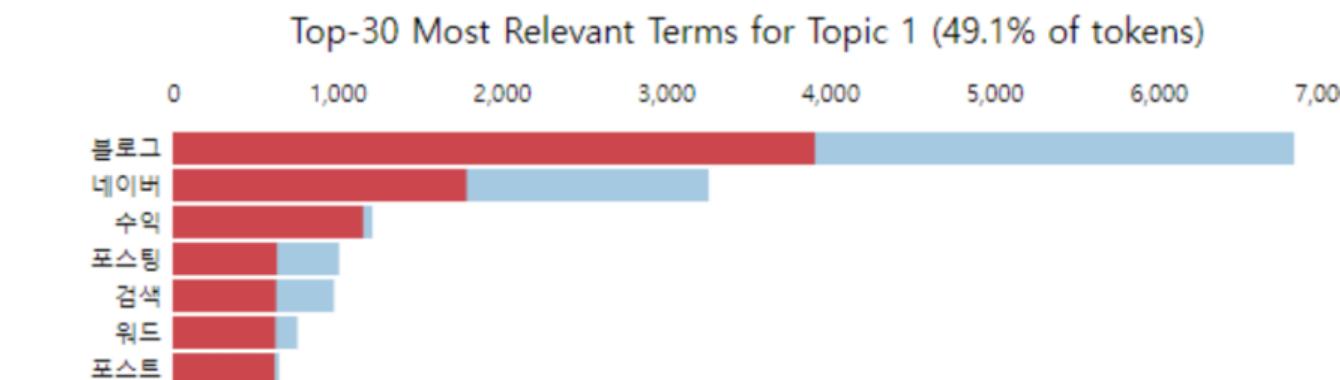
Topic 2: “네이버 블로그 중국 접속 차단”

Topic 3: “네이버 블로그의 다양한 활용”

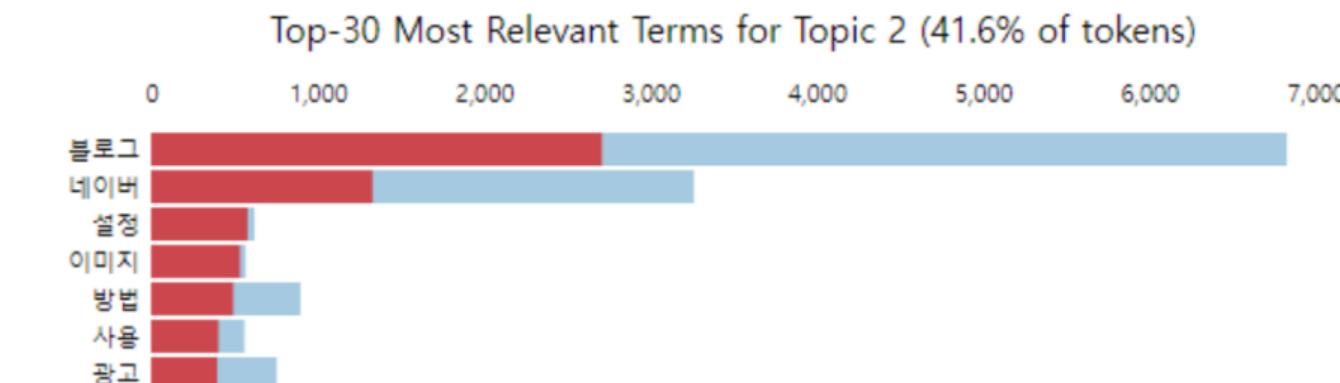


04 LDA 결과 해석: 네이버 블로그(블로그)

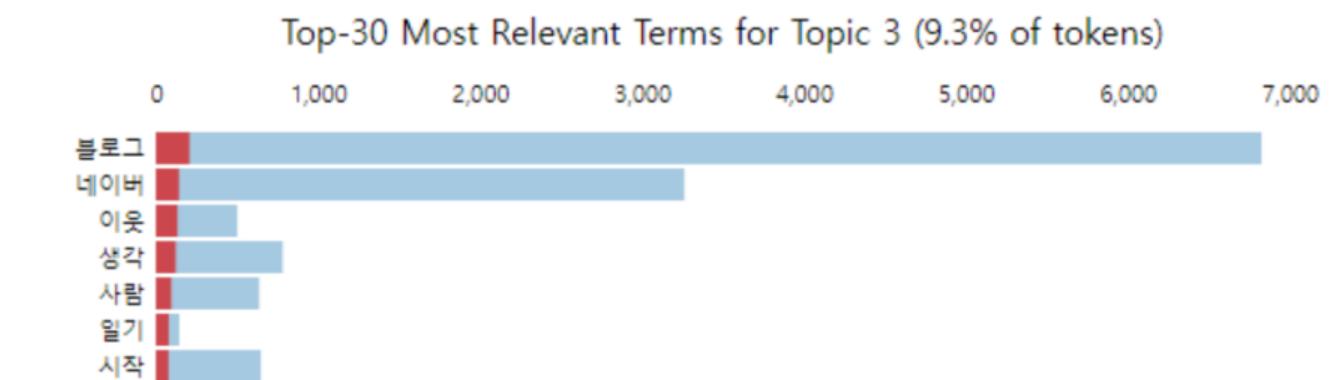
Topic 1: “네이버 블로그로 수익 창출”



Topic 2: “네이버 블로그 사용법”

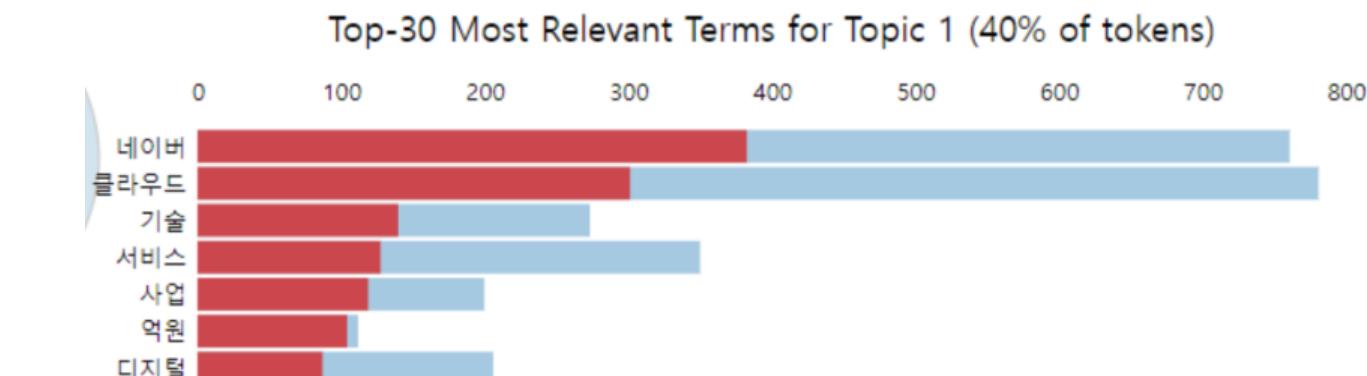


Topic 3: “네이버 블로그 글 주제”

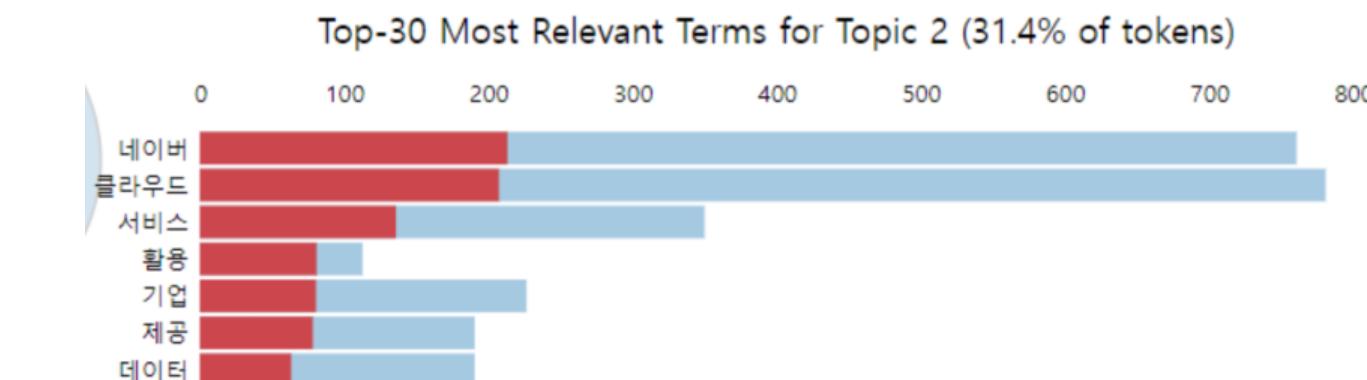


04 LDA 결과 해석: 네이버 클라우드(뉴스)

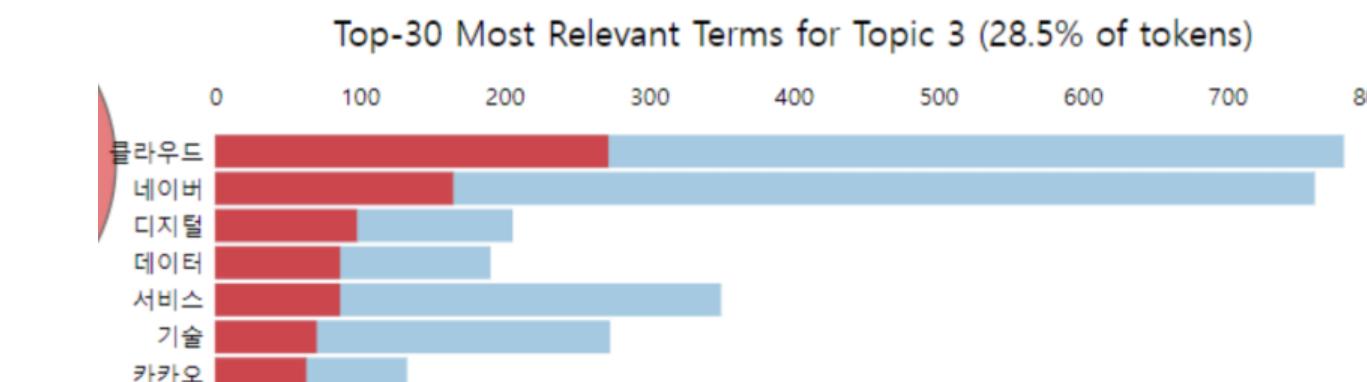
Topic 1: “네이버 클라우드 산업 시장”



Topic 2: “네이버 클라우드 서비스 플랫폼”

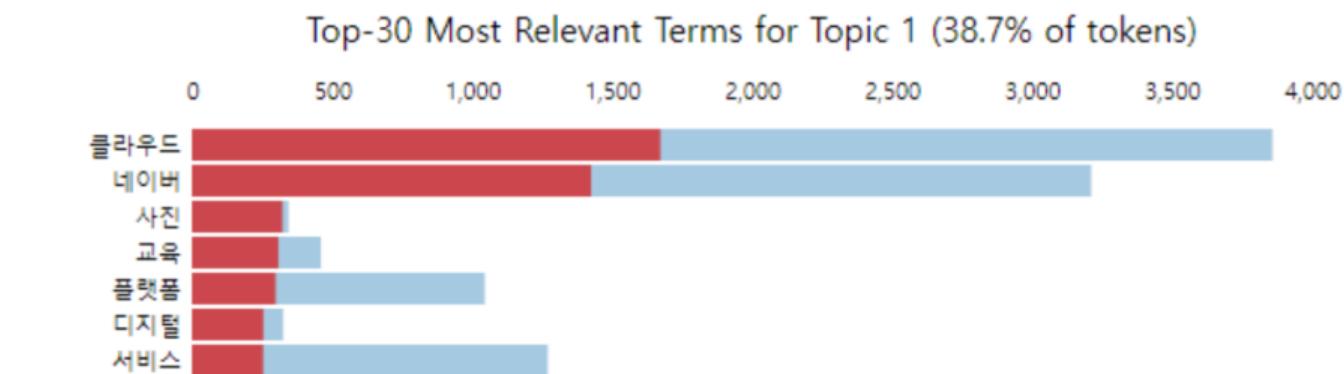


Topic 3: “네이버 클라우드 데이터 보안”

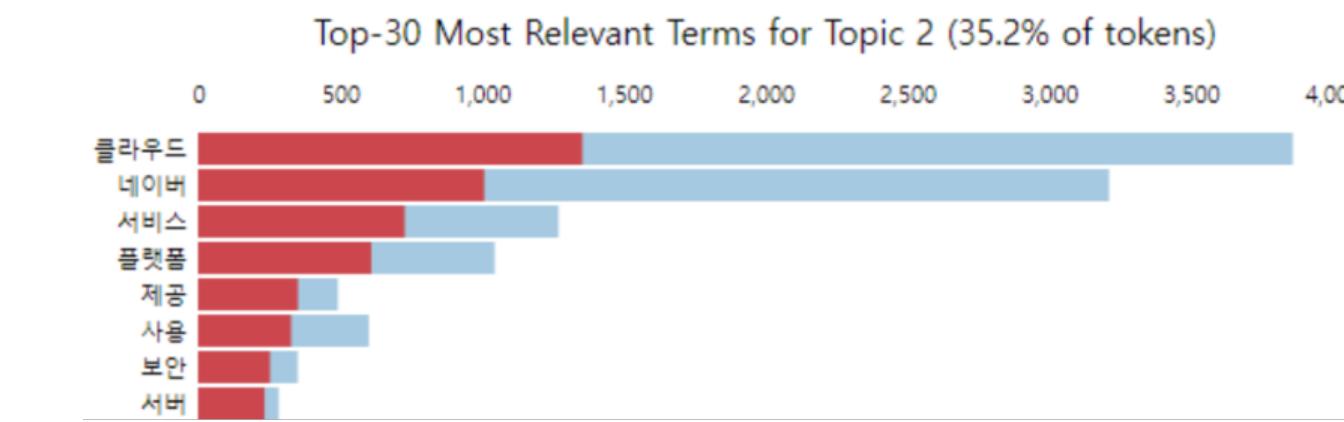


04 LDA 결과 해석: 네이버 클라우드(블로그)

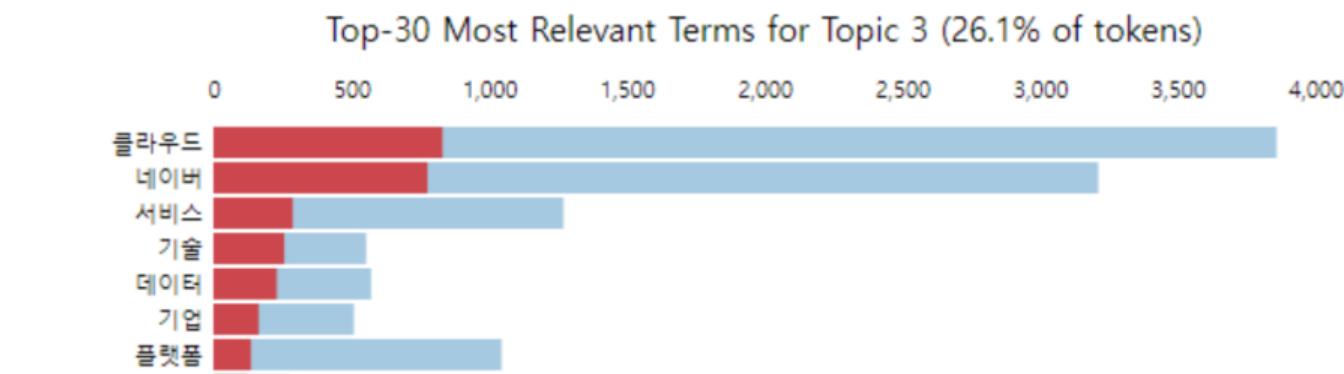
Topic 1: “네이버 클라우드 기술 교육”



Topic 2: “네이버 클라우드 고객 정보 보안”



Topic 3: “네이버 클라우드 서비스 개발”



감사합니다

