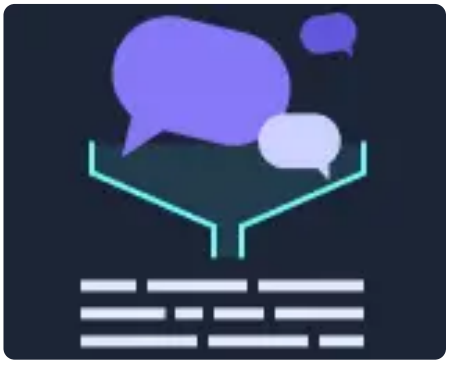
Dialogue Summarization I 일상 대화 요약



학교 생활, 직장, 치료, 쇼핑, 여가, 여행 등 광범위한 일상 생활 중 하는 대화들에 대해 요약합니다.

#비공개대회 #AI부트캠프14기 #NLPAdvanced

30 | 2025.09.26 10:00 ~ 2025.10.15 19:00 • 진행중

✓ 대회 참여중

[공유] 텍스트 데이터에서 인사이트를 얻기

크리스(운영진) · 2025.06.18 14:34 · 조회수 16 2 0

1. 텍스트 데이터

텍스트 데이터는 기존의 숫자로 구성된 데이터와 달리, 인사이트를 얻기가 상대적으로 어려울 수 있습니다. 하지만, 가설을 만들거나 조건을 주어 텍스트들의 패턴을 파악해볼 수 있답니다. 이번 대회의 데이터는 여러 사람들과 나눈 데이터를 바탕으로 요약문을 만드는 데이터이므로, 사람별로 발화가 구분되어 있을거고, 나눈 대화의 주제나 길이 등도 분석해볼 수 있겠죠?

2. 텍스트 데이터 살펴보기

그럼 훈련 데이터를 먼저 확인해보도록 하겠습니다.

	fname	dialogue	summary
0	train_0	#Person1#: 안녕하세요, 스미스씨. 저는 호킨스 의사입니다. 오늘 왜 오셨나...	스미스씨가 건강검진을 받고 있고, 호킨스 의사는 매년 건강검진을 받는 것을 권장합니...
1	train_1	#Person1#: 안녕하세요, 파커 부인, 어떻게 지내셨나요?\n#Person2#...	파커 부인이 리키를 데리고 백신 접종을 하러 갔다. 피터스 박사는 기록을 확인한 후...
2	train_2	#Person1#: 실례합니다, 열쇠 한 묶음 보셨나요?\n#Person2#: 어떤...	#Person1#은 열쇠 한 묶음을 찾고 있고, 그것을 찾기 위해 #Person2#...
3	train_3	#Person1#: 왜 너는 여자친구가 있다는 걸 말해주지 않았어?\n#Person...	#Person1#은 #Person2#가 여자친구가 있고 그녀와 결혼할 것이라는 사실...
4	train_4	#Person1#: 안녕, 숙녀분들! 오늘 밤 당신들은 정말 멋져 보여. 이 춤을 ...	말릭이 니키에게 춤을 요청한다. 말릭이 발을 밟는 것을 신경 쓰지 않는다면 니키는 ...
...	...	...	...
12452	train_12455	#Person1#: 실례합니다. 맨체스터 출신의 그린 씨이신가요?\n#Person2...	탄 링은 흰머리와 수염으로 쉽게 인식되는 그린 씨를 만나 호텔로 데려갈 예정입니다....
12453	train_12456	#Person1#: 이윙 씨가 우리가 컨퍼런스 센터에 오후 4시에 도착해야 한다고 ...	#Person1#과 #Person2#는 이윙 씨가 늦지 않도록 요청했기 때문에 컨퍼...
12454	train_12457	#Person1#: 오늘 어떻게 도와드릴까요?\n#Person2#: 차를 빌리고 싶...	#Person2#는 #Person1#의 도움으로 5일 동안 소형 차를 빌립니다.
12455	train_12458	#Person1#: 오늘 좀 행복해 보이지 않아. 무슨 일 있어?\n#Person2...	#Person2#의 엄마가 일자리를 잃었다. #Person2#는 엄마가 우울해하지 ...
12456	train_12459	#Person1#: 엄마, 다음 토요일에 이 삼촌네 가족을 방문하기 위해 비행기를 ...	#Person1#은 다음 토요일에 이 삼촌네를 방문할 때 가방을 어떻게 싸야 할지 ...

12457 rows × 4 columns

훈련 데이터를 먼저 살펴보니, dialogue 와 summary 로 데이터가 구성되어있네요! 각각의 대화에 대한 고유 index 가 fname 에 저장되어 있습니다. dialogue 가 대화에 참여한 발화자들이 나눈 전체 대화이고, 각각의 대화는 \n 으로 구분되어 있네요! 실제 하나의 대화를 출력해보면 아래와 같습니다.

```
전체 대화 내용
#Person1#: 안녕하세요, 스미스씨. 저는 호킨스 의사입니다. 오늘 왜 오셨나요?
#Person2#: 건강검진을 받는 것이 좋을 것 같아서요.
#Person1#: 그렇군요, 당신은 5년 동안 건강검진을 받지 않았습니다. 매년 받아야 합니다.
#Person2#: 알고 있습니다. 하지만 아무 문제가 없다면 왜 의사를 만나러 가야 하나요?
#Person1#: 심각한 질병을 피하는 가장 좋은 방법은 이를 조기에 발견하는 것입니다. 그러니 당신의 건강을 위해 최소한 매년 한 번은 오세요.
#Person2#: 알겠습니다.
#Person1#: 여기 보세요. 당신의 눈과 귀는 괜찮아 보입니다. 깊게 숨을 들이쉬세요. 스미스씨, 담배 피우시나요?
#Person2#: 네.
#Person1#: 당신도 알다시피, 담배는 폐암과 심장병의 주요 원인입니다. 정말로 끊으셔야 합니다.
#Person2#: 수백 번 시도했지만, 습관을 버리는 것이 어렵습니다.
#Person1#: 우리는 도움이 될 수 있는 수업과 약물들을 제공하고 있습니다. 나가기 전에 더 많은 정보를 드리겠습니다.
#Person2#: 알겠습니다, 감사합니다, 의사선생님.
==========
대화 요약문
스미스씨가 건강검진을 받고 있고, 호킨스 의사는 매년 건강검진을 받는 것을 권장합니다. 호킨스 의사는 스미스씨가 담배를 끊는 데 도움이 될 수 있는 수업과 약물에 대한 정보를 제공할 것입니다.
```

실제로 대화문들은 구어체로 구성되어 있는데, 요약문은 문어체로 구성되어 있는 걸 볼 수 있습니다!

이 대화를 각 turn 별로 구분해서 보면 좀 더 분석하기가 쉽겠네요? 대화를 시작하는 첫 turn 에는 안부를 묻거나 본인이 누구인지, 주요 대화의 주제가 등장하는 대화가 이뤄지고 있습니다.

그렇다면 평가 데이터는 어떻게 구성되어 있을까요?

	fname	dialogue	summary1	summary2	summary3
0	test_0	#Person1#: 더슨 씨, 방아쓰기 좀 해주세요. \n#Person2#: 네, ...	더슨 씨는 #Person1#이 모든 직원에게 의식소통 방법을 변경하고 더 이상 즉시...	직원들이 즉시 메시지 프로그램에 시간을 낭비하는 것을 방지하기 위해, #Person...	더슨 씨는 #Person1#의 사무실에서 즉시 메시지 프로그램 사용 금지에 대한 발...
1	test_4	#Person1#: 이 롤링픽 공원이 정말 크네요!\n#Person2#: 네. 지금...	#Person1#은 롤링픽 스타디움의 크기, 수용량, 그리고 내부 설정에 놀라워한다.	#Person2#는 #Person1#에게 건설 중인 롤링픽 스타디움을 보여주고 스타...	#Person2#는 롤링픽 스타디움의 완공 시기, 수용량, 그리고 내부 설정을 #P...
2	test_6	#Person1#: 너 무슨 문제 있어? 왜 그렇게 땀이 굽고 있어?\n#Perso...	#Person2#가 가렵다고 느낀다. #Person1#은 수두라고 의심하고 #Per...	#Person1#은 #Person2#가 수두에 걸렸다고 의심하고 #Person2#로...	#Person1#은 #Person2#가 수두에 걸렸다고 생각하고 #Person2#에...
3	test_7	#Person1#: 잘 오셨습니다. 무엇을 도와드릴까요?\n#Person2#: 저는...	#Person2#가 체크아웃하고 #Person1#에게 청구서를 요청한다. #Pers...	#Person1#은 세탁 서비스에 대한 잘못된 청구서를 수정하고 #Person2#가...	#Person2#는 자신이 잘못 청구되었다는 것을 발견한다. #Person1#은 청...
4	test_8	#Person1#: 스티븐, 나 정말 당신의 도움이 필요해.\n#Person2#: ...	#Person1#은 스티븐에게 그의 아내를 설득해 이혼하지 않도록 부탁하고 있다. ...	스티븐은 #Person1#이 명백히 충실하다고 명세하면서 #Person1#의 아내를...	#Person1#은 스티븐에게 그의 아내를 설득해 이혼하지 않도록 부탁하고, 스티븐...
...	...	...	...	...	...
245	test_485	#Person1#: 안녕. 뭐하고 있어? \r\n#Person2#: 별거 없어. 너...	#Person1#와 #Person2#는 서로의 최근 상황에 대해 묻습니다.	#Person1#은 #Person2#의 활동에 대해 묻습니다.	#Person1#은 비쁘고 #Person2#는 유연합니다.
246	test_487	#Person1#: 다음 주에 할아버지 생신이에요. 깜짝 파티를 해야겠어요.\r\n...	#Person1#과 #Person2#는 할아버지의 생일을 논의합니다. 그들은 파티를...	#Person1#과 #Person2#는 할아버지의 생일을 위한 깜짝 파티를 계획하고...	다음 주에 #Person1#과 #Person2#의 할아버지 생신입니다. #Perso...
247	test_489	#Person1#: 나는 시골이 어떻게 시끄러울 수 있을지 상상도 못했어.\r\n#Person2#...	캐시는 소음에 대해 불평하고 #Person2#는 이것이 17년에 한 번씩 나타나는 ...	캐시는 이 기간에 새들이 나타나서 복제품이나 신액의 시끌이 너무 시끄럽다고 생각하...	캐시는 새들의 울음소리에 짜증이 난다. 새들은 대부분 시원할 때 나오고 주로 나무에...
248	test_492	#Person1#: 테드, 올해 휴가 어디로 가려고 해?\r\n#Person2#: ...	#Person1#와 테드는 휴가에 대해 이야기한다. 테드는 집에 머무를 예정이고 #...	테드와 #Person2#는 휴가를 보낼 장소에 대해 논의하고 있다.	테드의 아내는 몇 주 동안 부모님과 함께 있을 예정이지만, 테드는 그들과 함께 사는...
249	test_496	#Person1#: 어떻게 컨트리 음악에 관심을 가지게 되었니요?\r\n#Perso...	#Person2#는 #Person1#에게 컨트리 음악에 관심을 가지게 된 이유와 프...	#Person2#는 과거의 경력을 공유하는데, 이는 완전히 컨트리 음악과 얽혀 있습니다.	#Person2#는 #Person1#에게 다른 종류보다 컨트리 음악을 더 많이 사고...

평가 데이터는 하나의 대화에 3가지의 요약문이 부착되어 있어, 실제 평가를 할 때 모델이 다양한 응답을 반환하더라도 적절히 채점될 수 있게 되어있습니다.

우리는 모델을 학습할 것이기 때문에 max_length 를 확인하는 부분도 필요하겠죠? 물론 사용할 tokenizer 를 적용해 토큰화를 한 후에 max_length 를 찍어보는 게 좋겠지만, 일단 음절을 바탕으로 확인해보겠습니다.

```
# dialog 와 summary 각각의 모델 max_length 설정을 위한 길이 확인

train_dialog_length = train['dialogue'].apply(lambda x:len(x))
train_summary_length = train['summary'].apply(lambda x:len(x))

print("대화 길이에 대한 정보")
print(train_dialog_length.describe())
print("==================")
print("요약문 길이에 대한 정보")
print(train_summary_length.describe())
```

이를 바탕으로 대화, 요약문의 max_length를 적절히 선택하면 되겠죠?

이제 우리는 데이터들을 가지고 여러 방법들을 사용하여 인사이트를 얻어보겠습니다.

3. 텍스트 데이터 전처리

대회에서 주어진 데이터들은 잘 정제가 되어있지만, 문어체로 대화가 이루어지고 있기 때문에 자음이나 모음만으로 구성된 (ㅋㅋ, ㅇㅇ 등) 경우가 있는지 확인해보고, 이를 대체하는 방법까지 알아보려 합니다. 자/모음으로 구성된 경우가 아니더라도, 데이터에서 특정한 값이 포함되는지 찾거나 대체할 때 이 방법을 사용할 수 있습니다.

```
# 데이터에서 특정 텍스트로 되어있는 부분이 있는지 확인하는 방법 : find 함수 사용
train[train['dialogue'].apply(lambda x:x.find('ㅇㅇ')!= -1)]
## ㅇㅇ 이 포함되어있는 데이터 없음

# ㅋㅋ 가 포함되어 있는 데이터 확인
train[train['dialogue'].apply(lambda x:x.find('ㅋㅋ')!= -1)].values
## 확인해보니 ㅋㅋ 가 포함되어 있는 데이터 존재. 이를 웃기다 정도로 대체

# replace 를 사용하여 ㅋㅋ 를 웃기다로 대체 후 저장
train['dialogue'] = train['dialogue'].apply(lambda x:x.replace('ㅋㅋ', '웃기다'))

# 다시 확인해보니 ㅋㅋ 포함된 데이터 없어짐 확인
train[train['dialogue'].apply(lambda x:x.find('ㅋㅋ')!= -1)].values
```

4. 워드클라우드

데이터 정제를 완료했으면, 데이터에서 많이 등장하는 단어들을 확인하여 주요 대화 주제 등을 확인합니다. 물론 단어 빈도만으로도 가능하지만, 시각화를 해서 좀 더 보기 좋게 만들어보겠습니다.

모든 대화를 다 사용하면 조사가 바뀔 때 마다 각자 다르게 인식하기 때문에, 우선 대화 중 단어 토큰화, 명사 추출을 한 후 보겠습니다.

```
# 단어 토큰화, 명사 추출
from konlpy.tag import
Oktokt=Okt()

print('단어 토큰화 결과 ==>', okt.morphs(train['dialogue'].iloc[0]))
print('명사 추출 결과 ==>', okt.nouns(train['dialogue'].iloc[0]))
```

```
단어 토큰화 결과 ==> ['#Person1', '#:', '안녕하세요', ',', '스미스', '씨', ',', '저', '는', '호킨스', '의사', '입니다', '.', '오늘', '왜', '오셨나요', '?', '#n
명사 추출 결과 ==> ['스미스', '저', '호킨스', '의사', '오늘', '왜', '건강검진', '것', '당신', '동안', '건강검진', '매년', '달', '아무', '문제', '왜', '의,
```

실제로 모델을 학습할 때는 토큰화를 진행하여 사용하지만, 지금은 인사이트를 얻기 위해서 하기 때문에 명사만 추출한 결과를 사용하도록 하겠습니다. 이를 바탕으로 가장 많이 등장한 단어를 워드 클라우드로 시각화해보면 다음과 같습니다.

```
def get_noun(text):
    okt = Okt()
    noun = okt.nouns(text)
    for i,v in enumerate(noun):
        if len(v)<2:
            noun.pop(i)
    count = Counter(noun)
    noun_list = count.most_common(100)
    return noun_list

def visualize(noun_list, title):
    # 워드클라우드 이미지 생성
    wc= WordCloud(
        font_path = '/content/drive/MyDrive/NLP_Advanced/data_csv/NanumGothic.ttf',    # 한글폰트 경로
설정
        background_color='white',                          # 배경 색깔 정하기
        colormap = 'Dark2',                               # 폰트 색깔 정하기
        width = 800,
        height = 800).generate_from_frequencies(dict(noun_list))
    plt.figure(figsize=(10,10)) #이미지 사이즈 지정
    plt.suptitle("Word Cloud", fontsize=40)
```

```
    plt.title(title, fontsize=20)
    plt.imshow(wc, interpolation='lanczos') #이미지의 부드럽기 정도
    plt.axis('off') #x y 축 숫자 제거
    plt.show()  # 워드클라우드 이미지 확인
    return wc

total_reviews = visualize(noun_list,'total') # 워드클라우드 시각화
```

많이 등장한 단어들 중 우리, 정말과 같은 대화 주제별로 다른 단어가 아니라 일반적으로 대화에 많이 쓰는 단어들이 많이 등장하여 인사이트를 얻기가 조금 힘듭니다. 이런 경우를 방지하려고 TF-IDF(Term Frequency - Inverse Document Frequency) 라는 방법을 씁니다. 이는 어떤 단어가 특정 문서 내에서 얼마나 중요한 것인지를 나타내는 수치인데, 이 방법을 사용하게 되면 여러 문서에서 공통적으로 자주 등장하는 단어보다는 특정 문서 내에서 더 중요하게 판단되는 단어를 추출해줍니다.

5. 개인정보 마스킹 부분 확인

해당 데이터셋에는 개인정보가 포함되어 있었어서, 이 개인정보들을 마스킹 하여 제공합니다.

```
###
전화번호 → #PhoneNumber#
주소 → #Address#
생년월일 → #DateOfBirth#
여권번호 → #PassportNumber#
사회보장번호 → #SSN#
```

```
신용카드 번호 → #CardNumber#
차량 번호 → #CarNumber#
이메일 주소 → #Email
####
```

이렇게 8가지의 정보들을 마스킹 해두었는데, 두 개의 # 사이에 어떤 정보가 마스킹 되어있는지를 표시합니다.

이런 패턴을 가지고 있는 값들을 추출하기 위해서는 정규표현식을 사용할 수 있습니다.

```
# 정규표현식 사용하기
import re
def reg_masking(text):
  pattern = r"#Ww+#"  # ## 사이의 값을 추출하는 정규식 패턴
  masked = re.findall(pattern, text)
  return masked

train_set = train['dialogue'].apply(lambda x:str(set(reg_masking(x))))
```

이렇게 정규표현식을 적용한 결과를 확인해보면 다음과 같습니다.

```
           {'#Person2#', '#Person3#', '#Person1#'}
       {'#Person2#', '#PhoneNumber#', '#Person1#'}
{'#Person2#', '#PhoneNumber#', '#Address#', '#...
   {'#Person2#', '#PassportNumber#', '#Person1#'}
       {'#Person2#', '#PhoneNumber#', '#Person1#'}
                          ...
       {'#Person2#', '#PhoneNumber#', '#Person1#'}
       {'#Person2#', '#PhoneNumber#', '#Person1#'}
           {'#Person2#', '#Person3#', '#Person1#'}
       {'#Person2#', '#DateOfBirth#', '#Person1#'}
           {'#Person2#', '#Person3#', '#Person1#'}
```

발화자를 구분하는 토큰도 #PersonN# 두 개의 # 사이에 Person 번호를 넣어 구성하고 있네요. 두 개의 # 사이에 숫자로 끝나는 값이 들어있으면 추출하는 명령어를 주면 대화에 포함된 발화자가 몇 명인지 확인할 수 있겠습니다.

```
import re
def reg_person(text):
  pattern = r"#Ww+Wd#"  # ## 사이의 값을 추출하는 정규식 패턴 > special token 으로 tokenizer에 추가
  masked = re.findall(pattern, text)
  return masked

train_person = train['dialogue'].apply(lambda x:set(reg_person(x)))
```

```
[{'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
 {'#Person1#', '#Person2#'},
```

이렇게 정규표현식을 잘 활용하면 원하는 텍스트들의 패턴을 확인하여 추출할 수 있습니다. 이렇게 마스킹 한 값들을 tokenizer 에 special token으로 포함시키려면 다음과 같은 방법을 사용할 수 있습니다.

```
# special token 추가
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("gogamza/kobart-base-v2")
special_tokens_dict = {'additional_special_tokens': unique_token}
tokenizer.add_special_tokens(special_tokens_dict)
tokenizer.additional_special_tokens, tokenizer.additional_special_tokens_ids
```

추가된 토큰들의 index 를 확인하면 다음과 같이 잘 추가되었음을 확인할 수 있네요!

```
#Email# : 30000
#Person5# : 30001
#DateOfBirth# : 30002
#SSN# : 30003
#CarNumber# : 30004
#Person# : 30005
#PassportNumber# : 30006
#Person2# : 30007
#Person7# : 30008
#Address# : 30009
#Person1# : 30010
#Person6# : 30011
#CardNumber# : 30012
#Person3# : 30013
#PhoneNumber# : 30014
#Person4# : 30015
```

6. 토론

이렇게 텍스트 데이터에서는 다양한 정보를 추출할 수 있습니다. 어떤 특정한 단어나 패턴을 파악하여 정제하는 등의 다양한 시도를 해보면서 모델의 성능을 높일 수 있는 방법들을 생각해봅시다!

댓글