코더의 소회 :

날씨 : 오늘의 학습 < 보통☁️>했습니다. [🌞🌤☁️🌧🌪☔]

- 배운점 : 진행하고자하는 훈련에 꼭! 기 원하는 옵션이 있다면 모델 설계때 모델 구성요소(ex.옵티마이저저 등)에 **해당 기능을 지원하는가**, 지원하지 않으면 다른 비슷한 대안이 있는가 알아보면 좋을 것 같다.
- 아쉬운점 : Matrix까지 실행하지 못하였던게 아쉽습니다. 엉뚱한 곳(파일 압축해제 함수화)에서 오전 시간을 다 사용한 것이 아쉽습니다. 영번역까지 해보고싶습니다.
- 느낀점 : 1) 간단할 것 같은데라고 생각해도 실제로 할때 그렇지 않을 확률이 높구나라는 것을 느꼈습니다. 2)금요일밤 미리 노드 11 코드를 클라우드에 가져오며 코드를 찬찬히 훑터보았던 것, 노트북 포멧을 만들어 놓는 것이 오늘 시간을 많이 벌어주었습니다.

(참고 : https://github.com/Annyong2/AIFFEL_quest_rs/tree/master/GoingDeeper/Gdr04 (https://github.com/Annyong2/AIFFEL_quest_rs/tree/master/GoingDeeper/Gdr04))

전체 코드 실행 플로우 (목차):

[TIP] STEP 4. 모델 설계

- 한국어를 영어로 잘 번역해 줄 멋진 Attention 기반 Seq2seq 모델을 설계하세요!
- 앞서 만든 모델에 Dropout 모듈을 추가하면 성능이 더 좋아집니다!
- Embedding Size와 Hidden Size는 실험을 통해 적당한 값을 맞춰 주도록 합니다!

[TIP] STEP 5. 모델 훈련

- traing loss 안정적으로 떨어지면서 학습 진행

[TIP] STEP 6. 모델 평가(테스트)

- 데스트용 디코더 모델의 출력 결과가 정답과 어느 정도 유사한 영번역 진행

STEP 1. 불러오기 : 라이브러리 & 데이터

1.1 라이브러리

In [81]:
```
import pandas
import tensorflow as tf
import numpy as np

import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm  # 폰트 설정
import matplotlib.ticker as ticker    # 눈금 설정
import konlpy
from konlpy.tag import Mecab
from tqdm import tqdm    # tqdm
import random

import tarfile
import time
import re
import os
import io

print(tf.__version__)
```

```
2.6.0
```

In [2]:
```
# 한글설정
%config InlineBackend.figure_format = 'retina'

fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager.findfont(font)

print("완료!")
```

```
완료!
```

1.2 데이터

데이터 출처 : https://github.com/jungyeul/korean-parallel-corpora/tree/master/korean-english-news-v1 (https://github.com/jungyeul/korean-parallel-corpora/tree/master/korean-english-news-v1)

1.2 데이터 : 다운로드

In [3]:
```
def download_dataset(url, dataset_path, zipfilename):
    # 데이터셋 폴더 생성
    if not os.path.exists(dataset_path):
        os.makedirs(dataset_path)

    # zip 파일 경로 지정
    data_path = os.path.join(dataset_path, zipfilename)

    # 데이터셋 다운로드
    path_to_zip = tf.keras.utils.get_file(
        zipfilename,
        origin=url,
        extract=True,
        cache_dir=dataset_path)

    return path_to_zip
```

In [4]:
```
# 데이터셋 저장경로
url = 'https://github.com/jungyeul/korean-parallel-corpora/tree/master/korean-english-news-v1'
dataset_train_path = os.getenv('HOME') + '/aiffel/s2s_translation/datasets/data_train'
dataset_test_path = os.getenv('HOME') + '/aiffel/s2s_translation/datasets/data_test'
```

In [5]:
```
# TRAIN
# path_to_zip = download_dataset(url, dataset_train_path, 'korean-english-park.train.tar.gz')
# path_to_file = os.path.join(os.path.dirname(path_to_zip), 'korean-english-park.train.tar.gz')
```

In [6]:
```
# TEST
# path_to_zip_test = download_dataset(url, dataset_test_path, 'korean-english-park.test.tar.gz')
# path_to_file = os.path.join(os.path.dirname(path_to_zip_test), 'korean-english-park.test.tar.gz')
```

1.2 데이터 : 압축해제

In [7]:
```
def extract_tar_file(tar_path, extract_path):
    try:
        # 압축 파일을 해제합니다.
        if not os.path.exists(extract_path):
            with tarfile.open(tar_path, 'r:gz') as tar:
                tar.extractall(path=extract_path)
                # 압축 파일 내부의 파일 목록을 출력합니다.
                print("압축 해제된 파일 목록:")
                for member in tar.getmembers():
                    print(member.name)
        else:
            print(f"경로 {extract_path} 가 이미 존재합니다.")
    except Exception as e:
        print(f"압축 해제 중 오류 발생: {e}")
```

In [8]:
```
# TRAIN
# 1. 압축 해제
train_tar_path = os.path.join(dataset_train_path, 'korean-english-park.train.tar.gz')
extract_tar_file(train_tar_path, dataset_train_path)

# 2. 파일 경로 설정
train_korean_file = os.path.join(dataset_train_path, 'korean-english-park.train.ko')
train_english_file = os.path.join(dataset_train_path, 'korean-english-park.train.en')
```

```
경로 /aiffel/aiffel/s2s_translation/datasets/data_train 가 이미 존재합니다.
```

In [9]:
```
# TEST
# 1. 압축 해제
test_tar_path = os.path.join(dataset_test_path, 'korean-english-park.test.tar.gz')
extract_tar_file(test_tar_path, dataset_test_path)

# 2. 파일 경로 설정
test_korean_file = os.path.join(dataset_test_path, 'korean-english-park.test.ko')
test_english_file = os.path.join(dataset_test_path, 'korean-english-park.test.en')
```

```
경로 /aiffel/aiffel/s2s_translation/datasets/data_test 가 이미 존재합니다.
```

코멘트:

korean-english-park.test.tar.gz
왜인지 모르겠는데 해당 파일 압축해제가 안되어서 시간을 50분이나 허비했다...

korean-english-park.test.ko, korean-english-park.test.en
위 두 파일 로컬에서 풀어서 올렸다.

1.3 데이터 확인

1.3 데이터 확인 : TEST

In [18]:
```
# KOREAN
# 데이터 형태 확인
with open(test_korean_file, "r") as f:
    raw_test_ko = f.read().splitlines()

print("Data Size:", len(raw_test_ko))
print("Example:")

for sen_test_ko in raw_test_ko[0:100][::20]: print(">>", sen_test_ko)
```

```
Data Size: 2000
Example:
>> 토론에 참여한 사람들은 법 집행과 국가 안전보장에 대한 우려를 표명해야 할 필요성을 진지하게 받아 들이고 있습니다.
>> 비록 그 위협(의 가능성)은 적지만, 그 잠재적인 영향력은 가히 파괴적인 것이다.
>> 한국은 세계 에서 네번째로 원유를 많이 수입하는 나라이며, 전적으로 원유 수입에 의존하고 있다.
>> 불과 1,379년 전이다.
>> 대신, 그들은 "안정"이라는 면에서 서방 세계와 똑같은 이해 관계를 갖고 있는 각 지역의 소수 엘리트를 대표하고 있다.
```

In [19]:
```
# ENGLISH
# 데이터 형태 확인
with open(test_english_file, "r") as f:
    raw_test_en = f.read().splitlines()

print("Data Size:", len(raw_test_en))
print("Example:")

for sen_test_en in raw_test_en[0:100][::20]: print(">>", sen_test_en)
```

```
Data Size: 2000
Example:
>> Those involved in the discussions do take seriously the need to address concerns of law enforcement and national security.
>> Even though the threat is small, the potential effects are devastating.
>> South Korea is the world's fourth largest oil importer and wholly depends on imports of crude.
>> That is only 1,379 years ago.
>> Instead, they represent a small elite in each place that has a common interest with the West in “stability.”
```

1.3 데이터 확인 : TRAIN

In [10]:
```
# KOREAN
# 데이터 형태 확인
with open(train_korean_file, "r") as f:
    raw_ko = f.read().splitlines()

print("Data Size:", len(raw_ko))
print("Example:")

for sen_ko in raw_ko[0:100][::20]: print(">>", sen_ko)
```

```
Data Size: 94123
Example:
>> 개인용 컴퓨터 사용의 상당 부분은 "이것보다 뛰어날 수 있느냐?"
>> 북한의 핵무기 계획을 포기하도록 하려는 압력이 거세지고 있는 가운데, 일본과 북한의 외교관들이 외교 관계를 정상화하려는 회담을 재개했다.
>> "경호 로보트가 침입자나 화재를 탐지하기 위해서 개인적으로, 그리고 전문적으로 사용되고 있습니다."
>> 수자원부 당국은 논란이 되고 있고, 막대한 비용이 드는 이 사업에 대해 내년에 건설을 시작할 계획이다.
>> 또한 근력 운동은 활발하게 걷는 것이나 최소한 20분 동안 뛰는 것과 같은 유산소 활동에서 얻는 운동 효과를 심장과 폐에 주지 않기 때문에, 연구학자들은 근력 운동이 심장에 큰 영향을 미치는지 여부에 대해 논쟁을 해왔다.
```

In [11]:
```
# ENGLISH
# 데이터 형태 확인
with open(train_english_file, "r") as f:
    raw_en = f.read().splitlines()

print("Data Size:", len(raw_en))
print("Example:")

for sen_en in raw_en[0:100][::20]: print(">>", sen_en)
```

```
Data Size: 94123
Example:
>> Much of personal computing is about "can you top this?"
>> Amid mounting pressure on North Korea to abandon its nuclear weapons program Japanese and North Korean diplomats have resumed talks on normalizing diplomatic relations.
>> “Guard robots are used privately and professionally to detect intruders or fire,” Karlsson said.
>> Authorities from the Water Resources Ministry plan to begin construction next year on the controversial and hugely expensive project.
>> Researchers also have debated whether weight-training has a big impact on the heart, since it does not give the heart and lungs the kind of workout they get from aerobic activities such as brisk walking or running for at least 20 minutes.
```

코멘트:

한국어데이터와 영어데이터 크기가 동일하다.
두 파일로 나눠져있는 데이터들은 순서(idx)도 매핑되어있다.(raw_ko[1] = raw_en[1])

그렇다면...
-> 두 데이터는 각각 따로 중복 제거해주면 되지 않을까?

In [12]:
```
print(type(raw_ko))
print(type(raw_en))
```

```
<class 'list'>
<class 'list'>
```

In [13]:
```
# 중복된 항목 확인 및 인덱스 출력
seen_ko = set()
duplicated_ko = []
for idx, item in enumerate(raw_ko):
    if item in seen_ko:
        print(f"중복 발견: {item.strip()} (인덱스: {idx})")
        duplicated_ko.append([item.strip(),"Wt",{idx}])
    else:
        seen_ko.add(item)
```

```
중복 발견: 번역 : (인덱스: 4833)
중복 발견: 번역 : (인덱스: 5034)
중복 발견: 번역 : (인덱스: 5065)
중복 발견: 어휘 : (인덱스: 5218)
중복 발견: 번역 : (인덱스: 5220)
중복 발견: 어휘 : (인덱스: 5278)
중복 발견: 세계 해운업 신문인 Lloyd's List는 이제부터 선박은 여성으로서의 성(性)을 잃게 되고, "그녀"가 아니라 "그것"으로 언급될 것이라고 결정했다. (인덱스: 5337)
중복 발견: 선박을 여성으로 취급하는 관습은 주로 영국의 지배를 받던 국가들에서 사용된 관례이기는 하지만, 그것이 어떻게 시작되었는지는 알려지지 않고 있다. (인덱스: 5338)
중복 발견: 어휘 : (인덱스: 5346)
중복 발견: 번역 : (인덱스: 5375)
중복 발견: 어휘 : (인덱스: 5494)
중복 발견: 번역 : (인덱스: 5495)
중복 발견: 어휘 : (인덱스: 5514)
중복 발견: 번역 : (인덱스: 5596)
중복 발견: 어휘 : (인덱스: 5742)
중복 발견: 번역 : (인덱스: 5778)
중복 발견: 어휘 : (인덱스: 5782)
중복 발견: 번역 : (인덱스: 5791)
```

In [14]:
```
print("Data Size:", len(duplicated_ko))
print()
for sen_ko_duplicated in duplicated_ko[0:200][::50]: print(">>", sen_ko_duplicated)
```

```
Data Size: 16532

>> ['어휘 :', 'Wt', {30}]
>> ['열렬한 오바마 지지자들인 패트릭 리히 상원의원과 크리스토퍼 도드 상원의원은 28일 클린턴이 중도 사퇴해야 한다는 주장을 펼쳤다.', 'Wt', {14057}]
>> ['교훈:', 'Wt', {32802}]
>> ['function pop_open(url){win = window.open(url,W'movieW',"scrollbars=no,width=770,height=570,left=50,top=50");}', 'Wt', {42276}]
```

In [15]:
```
# 중복된 항목 확인 및 인덱스 출력
seen_en = set()
duplicated_en = []
for idx, item in enumerate(raw_en):
    if item in seen_en:
        print(f"중복 발견: {item.strip()} (인덱스: {idx})")
        duplicated_en.append([item.strip(),"Wt",{idx}])
    else:
        seen_en.add(item)
```

```
red by Australia and the United States," Lyn Allison, leader of the Australian Democrats minor opposition party, said in a statement. (인덱스: 49639)
중복 발견: She said she was disappointed that all but one member of the government refused to sign. (인덱스: 49640)
중복 발견: "It would have been a more powerful letter had they signed, of course, but it's nonetheless a very significant demonstration of the depth of feeling in the Australian Parliament on this issue," Allison later told Australian Broadcasting Corp. radio. (인덱스: 49641)
중복 발견: The Democrats co-wrote the letter with the major opposition Labor Party. (인덱스: 49642)
중복 발견: The letter was also supported by the Greens party and independents. (인덱스: 49643)
중복 발견: Congress had confirmed Pelosi's receipt of the letter, a Labor official said Friday. (인덱스: 49644)
중복 발견: The letter suggests Congress pass a resolution insisting that the 31-year-old alleged Taliban fighter be immediately repatriated to stand trial in Australia. (인덱스: 49645)
중복 발견: Failing repatriation, the lawmakers request Hicks "be immediately put to trial before a properly constituted U.S. criminal court. (인덱스: 49646)
중복 발견: Hicks has been held at the U.S. military prison in Guantanamo since January 2002, a month after he was detained in Afghanistan. (인덱스: 49647)
중복 발견: His lawyers say he is suffering from depression and ill health because of the conditions of his incarceration. (인덱스: 49648)
중복 발견: He was originally charged with attempted murder, conspiracy to commit war crimes and aiding the enemy, and was selected to face a U.S. military tribunal. (인덱스: 49649)
중복 발견: But his case was thrown into limbo when the U.S. Supreme Court declared the commissions illegal in June. (인덱스: 49650)
중복 발견: Howard, a staunch U.S. ally in the war on terror, says his government is unhappy about the delay in bringing Hicks to trial but has resisted pressure to request his repatriation. (인덱스: 49651)
```

In [16]:
```
print("Data Size:", len(duplicated_en))
print()
for sen_en_duplicated in duplicated_en[0:200][::40]: print(">>", sen_en_duplicated)
```

```
Data Size: 18525

>> ['Authorities said that torrential rains are expected to pound the region for another 10 days raising water levels further, according to the news agency Xinhua.', 'Wt', {7218}]
>> ['Mike Huckabee dropped out of the Republican race after the results came in.', 'Wt', {16552}]
>> ['For men, kissing is more often used as a means to an end namely, to gain sexual access.', 'Wt', {24482}]
>> ['JOHANSSON:', 'Wt', {33486}]
>> ['The cars, packed with fuel and nails, could have killed hundreds if they had been set off.', 'Wt', {38371}]
```

단계 전체 코멘트 : 생각치도 못한 부분

- len(duplicated_ko) = 16532
- len(duplicated_en) = 18525

영어 한국어 데이터, 중복 개수가 다르며
(아래 코드블럭 참고) 중복이 시작되는 인덱스도 다르다.

=> LMS 노드11에서 pandas 사용하지 않아서, 사용하지 않고 병렬구조로 엮어서 Set 넣어보려 했다.
=> 아래 **시도01(Step 2. 데이터 정제)**에서 실패하면 pandas 바로 쓸 것이다.

In [47]:
```
# 중복 발견: Good to know: (인덱스: 10070)
# 중복 발견: Good to know: (인덱스: 10074)
# '열렬한 오바마 지지자들인 패트릭 리히 상원의원과 크리스토퍼 도드 상원의원은 28일 클린턴이 중도 사퇴해야 한다는 주장을 펼쳤다.', 'Wt', {14057}

print("< 중복 count : 한국어 , 영어 >")
print("한국어 중복 개수 : ",len(duplicated_ko)) # 16532
print("영  어 중복 개수 : ",len(duplicated_en)) # 18525
print()
print("< 동일 [index] : 한국어 >> 영어 >")
print(f"[10070]: {raw_en[10070]}  >>  {raw_ko[10070]}")
print(f"[10074]: {raw_en[10074]}  >>  {raw_ko[10074]}")
print(f"[14057]: {raw_en[14057]}  >>  {raw_ko[14057]}")
print()
print("< index : 한국어[0:4] , 영어[0:4] >")
for sen_ko_duplicated in duplicated_ko[0:4]: print(">>", sen_ko_duplicated)
for sen_en_duplicated in duplicated_en[0:4]: print(">>", sen_en_duplicated)
```

```
< 동일 [index] : 한국어 >> 영어 >
[10070]: Good to know:  >>  * 알아두면 좋은 것:
[10074]: Good to know:  >>  * 알아두면 좋은 것:
[14057]: Christopher Dodd and Patrick Leahy, said Friday that Clinton should rethink her chances of overcoming that deficit and consider folding her campaign.  >>  열렬한 오바마 지지자들인 패트릭 리히 상원의원과 크리스토퍼 도드 상원의원은 28일 클린턴이 중도 사퇴해야 한다는 주장을 펼쳤다.

< index : 한국어[0:4] , 영어[0:4] >
>> ['어휘 :', 'Wt', {30}]
>> ['어휘 :', 'Wt', {282}]
>> ['어휘 :', 'Wt', {1456}]
>> ['어휘 :', 'Wt', {2226}]
>> ['Authorities said that torrential rains are expected to pound the region for another 10 days raising water levels further, according to the news agency Xinhua.', 'Wt', {7218}]
>> ['The flood has hit nine provinces the hardest &#8212; including Sichuan, which is still reeling from a 7.9-magnitude earthquake that struck southwestern China on May 12.', 'Wt', {7219}]
>> ["One of the lawmakers said Clinton's husband, former President Bill Clinton, has been pushing the idea privately for several weeks.", 'Wt', {7789}]
>> ["Myanmar's government has put the death toll at 78,000.", 'Wt', {8713}]
```

STEP 2. 데이터 정제

시도01_set()사용하여 중복제거

In [60]:
```
# 병렬 데이터 합치기 및 중복 제거

# 병렬 데이터셋 생성
dataset = list(zip(raw_ko, raw_en))


print("Type(dataset) : ", type(dataset),"₩n")
print()
print("제거전dataset : ","₩n")
for i, (ko, en) in enumerate(dataset[:5]):
    print(f"Index {i}:")
    print(f"  한국어: {ko}")
    print(f"  영어: {en}")
    print()
print()

# 병렬 데이터 중복 제거
unique_dataset = list(set(dataset))
print()

# 한국어와 영어 문장을 다시 분리
unique_ko, unique_en = zip(*unique_dataset)

per = round(len(unique_dataset)/len(dataset),1)
print(f"중복 제거 전 데이터 크기: {len(dataset)}")
print(f"중복 제거 후 데이터 크기: {len(unique_dataset)} , {(per)*100}%")
```

```
Type(dataset) :  <class 'list'>


제거전dataset :

Index 0:
  한국어: 개인용 컴퓨터 사용의 상당 부분은 "이것보다 뛰어날 수 있느냐?"
  영어: Much of personal computing is about "can you top this?"

Index 1:
  한국어: 모든 광마우스와 마찬가지 로 이 광마우스도 책상 위에 놓는 마우스 패드를 필요로 하지 않는다.
  영어: so a mention a few weeks ago about a rechargeable wireless optical mouse brought in another rechargeable, wireless mouse.

Index 2:
  한국어: 그러나 이것은 또한 책상도 필요로 하지 않는다.
  영어: Like all optical mice, But it also doesn't need a desk.

Index 3:
  한국어: 79.95달러하는 이 최첨단 무선 광마우스는 허공에서 팔목, 팔, 그외에 어떤 부분이든 그 움직임에따라 커서의 움직임을 조절하는 회전 운동 센서를 사용하고 있다.
  영어: uses gyroscopic sensors to control the cursor movement as you move your wrist, arm, whatever through the air.

Index 4:
  한국어: 정보 관리들은 동남 아시아에서의 선박들에 대한 많은 (테러) 계획들이 실패로 돌아갔음을 밝혔으며, 세계 해상 교역량의 거의 3분의 1을 운송하는 좁은 해로인 말라카 해협이 테러 공격을 당하기 쉽다고 경고하고 있다.
  영어: Intelligence officials have revealed a spate of foiled plots on ships in Southeast Asia and are warning that a narrow stretch of water carrying almost one third of the world's maritime trade is vulnerable to a terror attack.



중복 제거 전 데이터 크기: 94123
중복 제거 후 데이터 크기: 78968 , 80.0%
```

In [66]:
```
cleaned_corpus = unique_dataset
print("cleaned_corpus = unique_dataset")
print(" len(cleaned_corpus) : ", len(cleaned_corpus))
print("Type(cleaned_corpus) : ", type(cleaned_corpus))
print()
print("list_cleaned_corpus : ","₩n")
print(cleaned_corpus[:2])
```

```
cleaned_corpus = unique_dataset
 len(cleaned_corpus) :  78968
Type(cleaned_corpus) :  <class 'list'>

list_cleaned_corpus :

[('가장 더운 기온을 기록했던 올해 날씨와 산타아나에서 부는 바람으로 이번주 산불 발생은 더욱 잦아졌다.', 'Hot weather and Santa Ana winds marked the height of traditional wildfire season this weekend, after one of the driest years on record.'), ('하루에 1시간만이라도 전자 기기의 전원을 제거한다.', 'Give yourself at least an hour a day when you completely unplug from electronic devices.')]
```

중복제거 후 코멘트 :

- len(duplicated_ko) = 16532
- len(duplicated_en) = 18525

영어 한국어 데이터, *중복 개수*가 다르며 중복이 *시작되는 인덱스*도 달랐다.
그래서 각 영한 데이터를 병렬구조로 만들고, set()매써드 사용하여 중복제거를 하였다.

데이터 중복 제거후, 데이터는 이전의 80%정도 남았다.

In [68]:
```
# 병렬 데이터셋 전처리 및 코퍼스 생성
kor_corpus = []
eng_corpus = []
```

In [69]:
```
# 데이터 전처리 : 불용어 정제 및 토큰화 함수 정의
def preprocess_korean_sentence(sentence):
    mecab = Mecab()
    stopwords = ['을', '를', '이', '가', '은', '는', '에', '의', '와', '과', '도', '로', '으로']

    # 소문자 변환 및 공백 제거
    sentence = sentence.lower().strip()

    # 특수 문자 제거
    sentence = re.sub(r"[?.!,]", "", sentence)

    # 형태소 분석 및 불용어 제거
    tokens = [token for token in mecab.morphs(sentence) if token not in stopwords]

    return " ".join(tokens)

def preprocess_english_sentence(sentence):
    sentence = sentence.lower().strip()

    # 특수 문자 제거
    sentence = re.sub(r"[?.!,]", r"", sentence)

    # 토큰화 및 시작/종료 토큰 추가
    sentence = '<start> ' + sentence + ' <end>'

    return sentence
```

In [70]:
```
# 병렬 데이터셋 전처리 및 코퍼스 생성 (상위 3만 개 데이터만 사용)
for ko_sentence, en_sentence in cleaned_corpus[:30000]:
    kor_corpus.append(preprocess_korean_sentence(ko_sentence))
    eng_corpus.append(preprocess_english_sentence(en_sentence))

# 확인을 위해 일부 출력
print("Korean Corpus Sample:")
print(kor_corpus[:5])
print()
print("English Corpus Sample:")
print(eng_corpus[:5])
```

```
Korean Corpus Sample:
['가장 더운 기온 기록 했 던 올해 날씨 산타아나 에서 부 바람 이번 주 산불 발생 더욱 잦아졌 다', '하루 1 시간 만 라도 전자 기기 전원 제거 한다', '단체 공동 대표 톰 데 슐 전 민주당 상원 의원 다', '영상 에서 반복 적 들리 기타 음악 배경 크루즈 목소리 만 들리 면담 자 질문 답 하 고 있 었 다', '로그인 korea 사전']

English Corpus Sample:
['<start> hot weather and santa ana winds marked the height of traditional wildfire season this weekend after one of the driest years on record <end>', '<start> give yourself at least an hour a day when you completely unplug from electronic devices <end>', '<start> the other co-chair also is a former senate majority leader democrat tom daschle of south dakota <end>', '<start> "it shows tom cruise with all the wide-eyed fervor that he brings to the promotion of a movie making the argument for scientology" which it calls "the bizarre 20th-century religion <end>', '<start> they landed in north korean waters <end>']
```

👀 Data Review :

STEP 3. 데이터 토큰화

In [71]:
```
def tokenize(corpus):
    tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
    tokenizer.fit_on_texts(corpus)

    tensor = tokenizer.texts_to_sequences(corpus)

    tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor, padding='post')

    return tensor, tokenizer
```

In [75]:
```
kor_train, kor_tokenizer = tokenize(kor_corpus)
eng_train, eng_tokenizer = tokenize(eng_corpus)

print(len(kor_train))
print(kor_train,"₩n")
print(len(eng_train))
print(eng_train,"₩n")
```

```
30000
[[  146  9153  3396 ...     0     0     0]
 [  659    53    40 ...     0     0     0]
 [  175   711   276 ...     0     0     0]
 ...
 [ 1260   479  1001 ...     0     0     0]
 [   61   153    16 ...     0     0     0]
 [  148 20151 36196 ...     0     0     0]] 

30000
[[    2  2079  1055 ...     0     0     0]
 [    2   433  5682 ...     0     0     0]
 [    2     1    67 ...     0     0     0]
 ...
 [    2     1 17307 ...     0     0     0]
 [    2     1 10506 ...     0     0     0]
 [    2   892 45768 ...     0     0     0]] 
```

STEP 4. 모델 설계

In [86]:
```
# Attention 기반 Seq2seq 모델
class BahdanauAttention(tf.keras.layers.Layer):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.w_dec = tf.keras.layers.Dense(units)
        self.w_enc = tf.keras.layers.Dense(units)
        self.w_com = tf.keras.layers.Dense(1)

    def call(self, h_enc, h_dec):
        h_enc = self.w_enc(h_enc)
        h_dec = tf.expand_dims(h_dec, 1)
        h_dec = self.w_dec(h_dec)

        score = self.w_com(tf.nn.tanh(h_dec + h_enc))
        attn = tf.nn.softmax(score, axis=1)
        context_vec = attn * h_enc
        context_vec = tf.reduce_sum(context_vec, axis=1)

        return context_vec, attn

class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units):
        super(Encoder, self).__init__()
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.enc_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')
        # Dropout layer added to prevent overfitting
        self.dropout = tf.keras.layers.Dropout(0.5)

    def call(self, x):
        x = self.embedding(x)
        # Applying dropout after embedding to prevent overfitting
        x = self.dropout(x)
        output, state = self.gru(x)
        return output , state

class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units):
        super(Decoder, self).__init__()
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.dec_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')
        self.fc = tf.keras.layers.Dense(vocab_size)
        self.attention = BahdanauAttention(self.dec_units)
        self.dropout = tf.keras.layers.Dropout(0.5)

    def call(self, x, hidden, enc_output):
        context_vector, attention_weights = self.attention(enc_output, hidden)
        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
        x = self.dropout(x)
        output, state = self.gru(x)
        output = tf.reshape(output, (-1, output.shape[2]))
        x = self.fc(output)

        return x, state, attention_weights
```

In [94]:
```
BATCH_SIZE     = 32  # 기존 64에서 감소
SRC_VOCAB_SIZE = len(kor_tokenizer.index_word) + 1
TGT_VOCAB_SIZE = len(eng_tokenizer.index_word) + 1

units         = 512  # 기존 1024에서 감소
embedding_dim = 256  # 기존 512에서 감소

encoder = Encoder(SRC_VOCAB_SIZE, embedding_dim, units)
decoder = Decoder(TGT_VOCAB_SIZE, embedding_dim, units)

# sample input
sequence_len = 30

sample_enc = tf.random.uniform((BATCH_SIZE, sequence_len))
sample_output, sample_state = encoder(sample_enc)

print ('Encoder Output:', sample_output.shape)

sample_state = tf.random.uniform((BATCH_SIZE, units))

sample_logits, h_dec, attn = decoder(tf.random.uniform((BATCH_SIZE, 1)),
                                     sample_state, sample_output)

print ('Decoder Output:', sample_logits.shape)
print ('Decoder Hidden State:', h_dec.shape)
print ('Attention:', attn.shape)
```

```
Encoder Output: (32, 30, 512)
Decoder Output: (32, 45774)
Decoder Hidden State: (32, 512)
Attention: (32, 30, 1)
```

STEP 5. 모델 훈련 및 시각화

In [95]:
```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True, reduction='none')
```

In [99]:
```
def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss = loss_object(real, pred)

    mask = tf.cast(mask, dtype=loss.dtype)
    loss *= mask

    return tf.reduce_mean(loss)

import tensorflow as tf

run_opts = tf.compat.v1.RunOptions(report_tensor_allocations_upon_oom=True)

# @tf.function
def train_step(src, tgt, encoder, decoder, optimizer, eng_tok):
    bsz = src.shape[0]
    loss = 0

    with tf.GradientTape() as tape:
        kor_out, h_dec = encoder(src)

        eng_src = tf.expand_dims([eng_tok.word_index['<start>']] * bsz, 1)

        for t in range(1, tgt.shape[1]):
            pred, h_dec, _ = decoder(eng_src, h_dec, kor_out)
            loss += loss_function(tgt[:, t], pred)
            eng_src = tf.expand_dims(tgt[:, t], 1)

    batch_loss = (loss / int(tgt.shape[1]))

    variables = encoder.trainable_variables + decoder.trainable_variables
    gradients = tape.gradient(loss, variables)
    optimizer.apply_gradients(zip(gradients, variables))

    return batch_loss


'''run_opts추가
@tf.function
def train_step(src, tgt, encoder, decoder, optimizer, eng_tok):
    bsz = src.shape[0]
    loss = 0

    with tf.GradientTape() as tape:
        kor_out, h_dec = encoder(src)

        eng_src = tf.expand_dims([eng_tok.word_index['<start>']] * bsz, 1)

        for t in range(1, tgt.shape[1]):
            pred, h_dec, _ = decoder(eng_src, h_dec, kor_out)
            loss += loss_function(tgt[:, t], pred)
            eng_src = tf.expand_dims(tgt[:, t], 1)

    batch_loss = (loss / int(tgt.shape[1]))

    variables = encoder.trainable_variables + decoder.trainable_variables
    gradients = tape.gradient(loss, variables)
    optimizer.apply_gradients(zip(gradients, variables), options=run_opts) #HERE

    return batch_loss
'''

'''원본
@tf.function

def train_step(src, tgt, encoder, decoder, optimizer, eng_tok):
    bsz = src.shape[0]
    loss = 0

    with tf.GradientTape() as tape:
        # 인코더의 두 개의 출력값을 각각 받음
        kor_out, h_dec = encoder(src)

        eng_src = tf.expand_dims([eng_tok.word_index['<start>']] * bsz, 1)

        for t in range(1, tgt.shape[1]):
            pred, h_dec, _ = decoder(eng_src, h_dec, kor_out)
            loss += loss_function(tgt[:, t], pred)
            eng_src = tf.expand_dims(tgt[:, t], 1)

    batch_loss = (loss / int(tgt.shape[1]))

    variables = encoder.trainable_variables + decoder.trainable_variables
    gradients = tape.gradient(loss, variables)
    optimizer.apply_gradients(zip(gradients, variables))

    return batch_loss'''
```

Out[99]:
```
"₩n@tf.function₩n₩ndef train_step(src, tgt, encoder, decoder, optimizer, eng_tok):₩n    bsz = src.shape[0]₩n    loss = 0₩n₩n    with tf.GradientTape() as tape:₩n        # 인코더의 두 개의 출력값을 각각 받음₩n        kor_out, h_dec = encoder(src)₩n        ₩n        eng_src = tf.expand_dims([eng_tok.word_index['<start>']] * bsz, 1)₩n₩n        for t in range(1, tgt.shape[1]):₩n            pred, h_dec, _ = decoder(eng_src, h_dec, kor_out)₩n            loss += loss_function(tgt[:, t], pred)₩n            eng_src = tf.expand_dims(tgt[:, t], 1)₩n        ₩n    batch_loss = (loss / int(tgt.shape[1]))₩n₩n    variables = encoder.trainable_variables + decoder.trainable_variables₩n    gradients = tape.gradient(loss, variables)₩n    optimizer.apply_gradients(zip(gradients, variables))₩n    ₩n    return batch_loss"
```

오류기록

OOM(Out of Memory) 오류

- 발생 원인
  - 배치 크기 및 모델 크기: GPU 메모리가 부족할 때 발생하는 OOM 오류의 원인은 주로 배치 크기, 임베딩 크기, 또는 GRU 유닛 수가 GPU의 메모리 용량보다 너무 크기 때문입니다. 이로 인해 학습 중에 GPU 메모리를 초과하게 되어 발생했습니다.
- 해결 시도(successful)
  - **배치 크기 줄이기**: BATCH_SIZE를 기존 64에서 32로 줄이는 시도를 했습니다. 배치 크기를 줄이면 각 배치에서 처리해야 하는 데이터 양이 줄어들기 때문에 GPU 메모리의 사용량이 감소합니다.
  - **Embedding 및 Hidden Layer 크기 줄이기**: 임베딩 크기 (embedding_dim)와 GRU 유닛 수 (units)를 줄여 모델의 크기를 줄였습니다. 이를 통해 메모리 사용량을 줄여서 OOM 오류를 해결하려고 했습니다.
  - **tf.function 데코레이터 제거**: train_step 함수에서 @tf.function 데코레이터를 제거하여 텐서 생성 시 유연하게 처리될 수 있도록 했습니다. 이를 통해 메모리 관리가 더 유연하게 가능하도록 했습니다.
- 해결 시도(unsuccessful)
  - 메모리 부족의 원인을 파악을 위해 현재 할당된 텐서의 목록을 확인할 수 있는 옵션(tf.compat.v1의 RunOptions)를 추가하여 메모리 할당 정보를 출력하려고 했으나 optimizer가 해당 옵션 지원하지 않아서 사용할 수 없었다.

In [100]:
```
EPOCHS = 3 # 10

for epoch in range(EPOCHS):
    total_loss = 0

    idx_list = list(range(0, kor_train.shape[0], BATCH_SIZE))
    random.shuffle(idx_list)
    t = tqdm(idx_list)    # tqdm

    for (batch, idx) in enumerate(t):
        batch_loss = train_step(kor_train[idx:idx+BATCH_SIZE],
                                eng_train[idx:idx+BATCH_SIZE],
                                encoder,
                                decoder,
                                optimizer,
                                eng_tokenizer)

        total_loss += batch_loss

        t.set_description_str('Epoch %2d' % (epoch + 1))    # tqdm
        t.set_postfix_str('Loss %.4f' % (total_loss.numpy() / (batch + 1)))    # tqdm
```

```
Epoch  1: 100%|██████████| 938/938 [20:22<00:00,  1.30s/it, Loss 2.2024]
Epoch  2: 100%|██████████| 938/938 [20:50<00:00,  1.33s/it, Loss 2.1699]
Epoch  3: 100%|██████████| 938/938 [20:48<00:00,  1.33s/it, Loss 2.1721]
```

STEP 6. 모델 평가(테스트)

In [101]:
```
def evaluate(sentence, encoder, decoder):
    attention = np.zeros((eng_train.shape[-1], kor_train.shape[-1]))

    sentence = preprocess_korean_sentence(sentence)
    inputs = kor_tokenizer.texts_to_sequences([sentence.split()])
    inputs = tf.keras.preprocessing.sequence.pad_sequences(inputs,
                                                           maxlen=kor_train.shape[-1],
                                                           padding='post')

    result = ''

    kor_out, eng_hidden = encoder(inputs)
    eng_input = tf.expand_dims([eng_tokenizer.word_index['<start>']], 0)

    for t in range(eng_train.shape[-1]):
        predictions, eng_hidden, attention_weights = decoder(eng_input,
                                                             eng_hidden,
                                                             kor_out)

        attention_weights = tf.reshape(attention_weights, (-1, ))
        attention[t] = attention_weights.numpy()

        predicted_id = tf.argmax(tf.math.softmax(predictions, axis=-1)[0]).numpy()

        result += eng_tokenizer.index_word[predicted_id] + ' '

        if eng_tokenizer.index_word[predicted_id] == '<end>':
            return result, sentence, attention

        eng_input = tf.expand_dims([predicted_id], 0)

    return result, sentence, attention
```

In [104]:
```
def plot_attention(attention, sentence, predicted_sentence):
    fig = plt.figure(figsize=(10,10))
    ax = fig.add_subplot(1, 1, 1)
    ax.matshow(attention, cmap='viridis')

    fontdict = {'fontsize': 14}

    return ax
```

In [ ]:
```

```