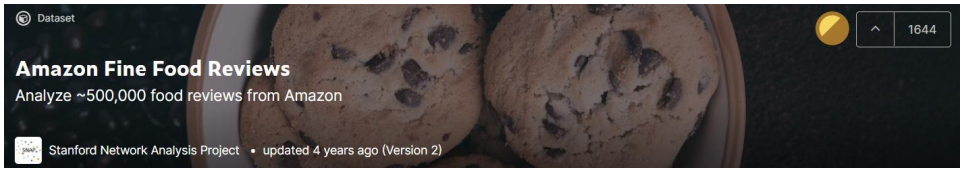
https://www.kaggle.com/snap/amazon-fine-food-reviews



이 데이터셋을 이용하여 텍스트 요약을 해볼 것이다. 이번주는 일단 데이터 전처리를 하였고 어떤 모델을 쓸지 생각해보았다.

1. 데이터 확인

```
print("전체 학습 데이터 개수: {}".format(len(data)))
```

```
전체 학습 데이터 개수: 568454
```

```
text_length=data['Text'].apply(len)
print('리뷰 길이 최댓값: {}'.format(np.max(text_length)))
print('리뷰 길이 최솟값: {}'.format(np.min(text_length)))
print('리뷰 길이 평균값: {:.2f}'.format(np.mean(text_length)))
print('리뷰 길이 표준편차: {:.2f}'.format(np.std(text_length)))
print('리뷰 길이 중간값: {}'.format(np.median(text_length)))
print('리뷰 길이 제1사분위: {}'.form
      at(np.percentile(text_length, 25)))
print('리뷰 길이 제3사분위: {}'.format(np.percentile(text_length, 75)))
```

```
리뷰 길이 최댓값: 21409
리뷰 길이 최솟값: 12
리뷰 길이 평균값: 436.22
리뷰 길이 표준편차: 445.34
리뷰 길이 중간값: 302.0
리뷰 길이 제1사분위: 179.0
리뷰 길이 제3사분위: 527.0
```

2. 중복 데이터 제거

```
sorted_data = data.sort_values('ProductId',axis=0,ascending=True,kind='quicksort',na_position='last')
non_redundant = sorted_data.drop_duplicates(subset={"UserId","Time","ProfileName","Text"},keep='first')
non_redundant = non_redundant[non_redundant['HelpfulnessNumerator']<=non_redundant['HelpfulnessDenominator']]
```

```
print(non_redundant.shape[1])
```

```
10
```

```
print("Percentage of data remained = ",(non_redundant["Id"].size*1.0/data["Id"].size*1.0)*100)
```

```
Percentage of data remained =  69.29865917031105
```

```
print(non_redundant.isna().any())
non_redundant.dropna(axis=0,inplace=True)
```

```
Id                      False
ProductId               False
UserId                  False
ProfileName              True
HelpfulnessNumerator    False
HelpfulnessDenominator  False
Score                   False
Time                    False
Summary                  True
Text                    False
dtype: bool
```

3. 데이터 전처리

(1) 줄임 문자 변경

```
def decontraction(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can₩'t", "can not", phrase)

    # general
    phrase = re.sub(r"n₩'t", " not", phrase)
    phrase = re.sub(r"₩'re", " are", phrase)
    phrase = re.sub(r"₩'s", " is", phrase)
    phrase = re.sub(r"₩'d", " would", phrase)
    phrase = re.sub(r"₩'ll", " will", phrase)
    phrase = re.sub(r"₩'t", " not", phrase)
    phrase = re.sub(r"₩'ve", " have", phrase)
    phrase = re.sub(r"₩'m", " am", phrase)
    return phrase
```

(2) 대문자를 소문자로 변경

```
def lower_case(sentence):
    sentence = ' '.join(e.lower() for e in sentence.split())
    return sentence

print(lower_case(non_redundant['Text'].values[900]))
```

```
this is a wonderful color assortment for any decorator. the colors blend in well and can be easily combined to create more variations.
```

(3) Html 태그 제거

```
from bs4 import BeautifulSoup


print(non_redundant['Text'].values[0])
print("₩n",'='*50,'₩n')

print(BeautifulSoup(non_redundant['Text'].values[0]))
print("₩n",'='*50,'₩n')

text = BeautifulSoup(non_redundant['Text'].values[0])
text = text.get_text() # Extracting text from the html object.
print(text)
```

```
In June<br />I saw a charming group<br />of roses all begin<br />to droop<br />I pepped them up<br />with chicken soup!<br />Sprinkle once

 ================================================

<html><body><p>In June<br/>I saw a charming group<br/>of roses all begin<br/>to droop<br/>I pepped them up<br/>with chicken soup!<br/>Spri

 ================================================

In JuneI saw a charming groupof roses all beginto droopI pepped them upwith chicken soup!Sprinkle oncesprinkle twicesprinkle chicken soupv
```

(4) 불용어 제거

```
import nltk
from nltk.corpus import stopwords
stopwords = set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",₩
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', ₩
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',₩
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', ₩
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', ₩
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', ₩
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',₩
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',₩
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',₩
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', ₩
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', ₩
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',₩
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',₩
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", ₩
            'won', "won't", 'wouldn', "wouldn't"])
# Creating sets of stop keywords
print(stopwords)

def rm_stop_words(sentence):
    sentence = ' '.join(e for e in sentence.split() if e not in stopwords)
    return sentence

print(non_redundant['Text'].values[900])
print("₩n",'='*50,'₩n')
print(rm_stop_words(non_redundant['Text'].values[900]))
```

```
{'being', 'me', 'shouldn', 'so', 'here', 'our', "mustn't", "wouldn't", 'while', 'ma', "shouldn't", 'didn', "hadn't", 'out', 'who', "aren'
This is a wonderful color assortment for any decorator.  The colors blend in well and can be easily combined to create more variations.

 ==================================================

This wonderful color assortment decorator. The colors blend well easily combined create variations.
```

(5) Html 태그 제거, 구두점 제거, 특수문자 제거, 최종 전처리

```
from tqdm import tqdm
import pdb
index,reviews = [],[]
preprocessed_review = []
for review in tqdm(non_redundant['Text'].values):
    review = re.sub(r'http₩S+','',review)
    review = BeautifulSoup(review).get_text()
    review = re.sub('[^A-Za-z0-9]+',' ',review)
    review = re.sub('₩S*₩d₩S*','',review)
    review = ' '.join(e.lower() for e in review.split() if e.lower() not in stopwords)
    preprocessed_review.append(decontraction(review))
```

```
print(preprocessed_review[1500])
print('₩n',len(preprocessed_review))
```

```
cat loves bubbles say want bubbles comes running wherever may hiding meows pounces bubbles giving rating though smell horrible drip anything white turn green easily cleaned

 393917
```

4. 사용할 모델 조사

(1) 시퀀스 투 시퀀스

https://brunch.co.kr/@kakao-it/139

인코더 RNN, 디코더 RNN, 주의 분포와 문맥 벡터, 어휘 분포

포인터 생성 네트워크

- Seq2Seq + Attention
- Pointer-Gen
- Pointer-Gen + Coverage

(2) TextRank

https://lovit.github.io/nlp/2019/04/30/textrank/

TextRank를 이용하여 핵심 문장을 추출하기 위해서는 문장 그래프를 만들어야 함. Consine similarity는 길이가 짧은 문장에 민감하게 반응할 수 있어서(16개의 단어로 구성된 문장에서는 3,4개의 단어가 공통으로 등장하면 중요한 단어이지만 2개의 단어로 구성된 문장에서는 하나의 단어만 함께 등장해도 절반이 넘는 단어가 공통으로 등장한 것임.) TextRank는 이러한 문제를 해결하기 위해 문장 간 유사도를 재정의하였다. TF-IDF + Consine Similarity 또는 Gensim의 summarize 함수에서는 BM25 함수 사용하였음. TextRank는 긴 문장에 높은 유사도를 부여하고 자주 등장하는 단어들을 다수 포함하는 문장을 핵심 문장으로 선택한다.