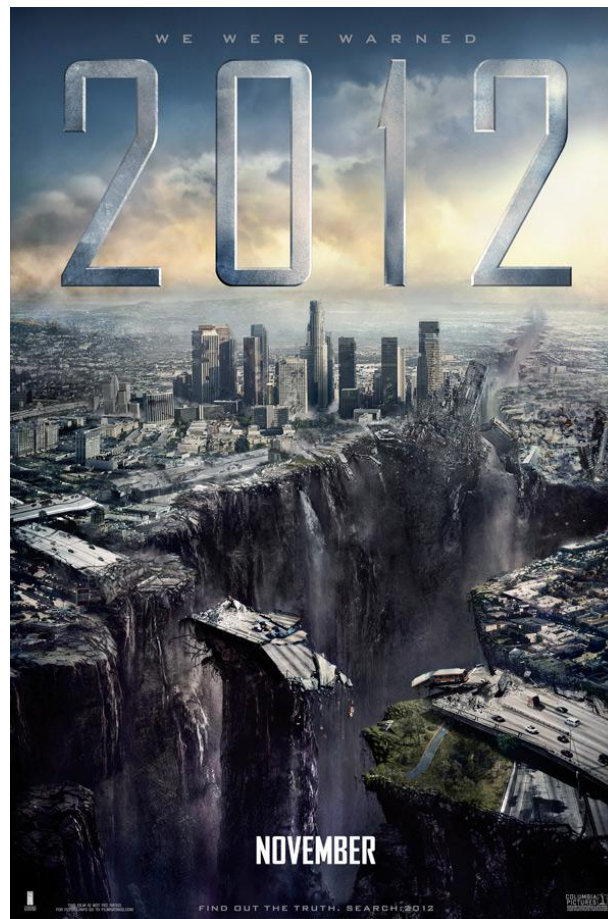


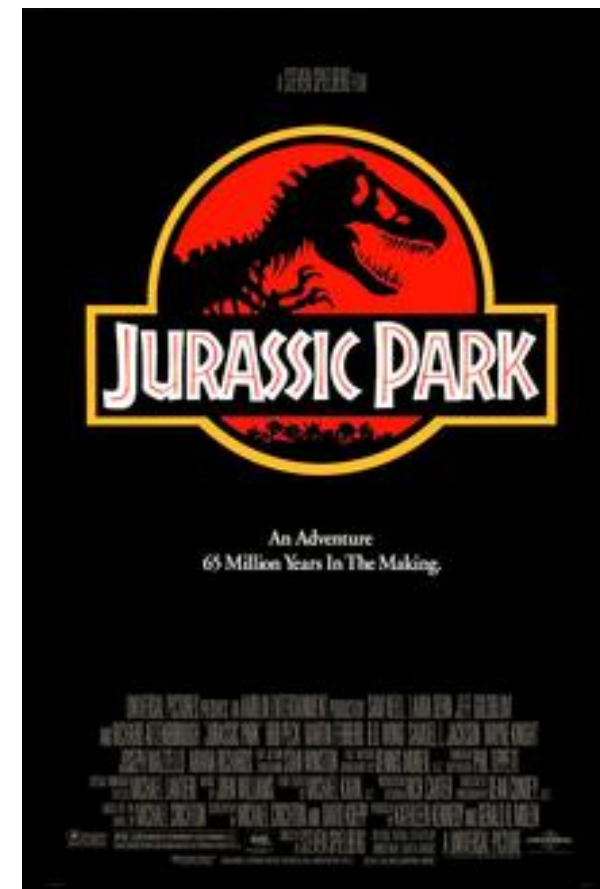
Movie script Text Analysis(2)

20186663 전찬웅

What kind of movies?



Title : **2012**
(Roland Emmerich)



Title : **Jurassic Park**
(Steven Allan Spielberg)

classifier

- I use TF-IDF

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{\text{df}_x} \right)$$

TF-IDF

Term x within document y

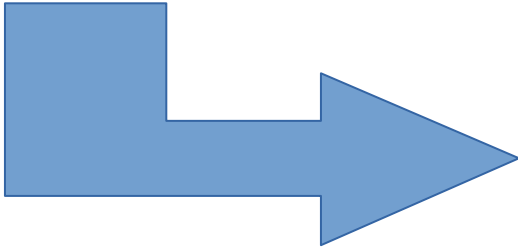
$\text{tf}_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Code Explanation – word tokenize

From Assignment 1



0. make word-tokenizer

```
stop_word = ["a", "about", "above", "across", "after", "afterwards", "again", "against", "all", "almost", "alone", "along", "already", "also", "among", "and", "an", "another", "any", "anybody", "anything", "anywhere", "are", "as", "at", "because", "been", "but", "by", "can", "cannot", "could", "couldn't", "did", "didn't", "do", "does", "doesn't", "doing", "don't", "down", "during", "each", "few", "for", "from", "further", "had", "hadn't", "has", "hasn't", "have", "haven't", "he", "her", "hers", "him", "his", "how", "however", "i", "if", "in", "into", "is", "isn't", "it", "its", "it's", "just", "knew", "know", "known", "knows", "last", "less", "lest", "like", "liked", "may", "maybe", "might", "mightn't", "must", "mustn't", "my", "myself", "never", "nevertheless", "no", "nobody", "none", "not", "noted", "nothing", "now", "nowhere", "of", "off", "on", "once", "only", "or", "other", "others", "otherwise", "out", "over", "own", "part", "per", "perhaps", "should", "shouldn't", "so", "some", "somebody", "something", "somewhere", "than", "that", "the", "there", "they", "they're", "this", "those", "through", "thru", "thus", "to", "too", "toward", "towards", "under", "until", "up", "upon", "us", "was", "wasn't", "we", "weren't", "were", "what", "whatever", "when", "whenever", "where", "wherever", "whether", "while", "whither", "who", "whoever", "whose", "why", "will", "won't", "with", "within", "without", "would", "wouldn't", "you", "you're", "your", "yours", "yourself", "yourselves"]

import re
import string
fre_2012 = {}
doc_txt = open('2012.txt', 'r')
text_string = doc_txt.read().lower()
match = re.findall(r'\b[a-z]{3,15}\b', text_string)
for word in match:
    count = fre_2012.get(word, 0)
    fre_2012[word] = count + 1

fre_list = fre_2012.keys()
lists=[]
for i in range(len(stop_word)):
    for words in fre_list:
        if stop_word[i] == words:
            lists.append(words)
for j in range(len(lists)):
    del fre_2012[lists[j]]

fre_jurassic = {}
doc_txt = open('jurassic park.txt', 'r')
text_string = doc_txt.read().lower()
match = re.findall(r'\b[a-z]{3,15}\b', text_string)
for word in match:
    count = fre_jurassic.get(word, 0)
    fre_jurassic[word] = count + 1

fre_list = fre_jurassic.keys()
lists=[]
for i in range(len(stop_word)):
    for words in fre_list:
        if stop_word[i] == words:
            lists.append(words)
for j in range(len(lists)):
    del fre_jurassic[lists[j]]
```

Code Explanation – parameter's meaning

- num_2012 is frequency of 2012 words in Movie 2012.
- anti_num_2012 is frequency of 2012 words in Movie Jurassic Park.
- num_jurassic is frequency of Jurassic Park words in Movie Jurassic Park.
- anti_num_jurassic is frequency of Jurassic Park words in Movie 2012.
- count_2012 is number of document included 2012 words.
- count_jurassic is number of document included Jurassic Park words.



```
list_2012 = list(fre_2012)
list_jurassic = list(fre_jurassic)
num_2012 = list(fre_2012.values())
anti_num_2012 = []
num_jurassic = list(fre_jurassic.values())
anti_num_jurassic = []
count_2012 = []
count_jurassic = []
```

Code Explanation – parameter's meaning

- num_2012 is frequency of 2012 words in Movie 2012.
- anti_num_2012 is frequency of 2012 words in Movie Jurassic Park.
- num_jurassic is frequency of Jurassic Park words in Movie Jurassic Park.
- anti_num_jurassic is frequency of Jurassic Park words in Movie 2012.



```
for i in range(len(list_2012)):
    a = 0
    for j in range(len(list_jurassic)):
        if list_2012[i] == list_jurassic[j]:
            anti_num_2012.append(num_jurassic[j])
            a = 1
    if a == 0:
        anti_num_2012.append(0)
for i in range(len(list_jurassic)):
    a = 0
    for j in range(len(list_2012)):
        if list_jurassic[i] == list_2012[j]:
            anti_num_jurassic.append(num_2012[j])
            a = 1
    if a == 0:
        anti_num_jurassic.append(0)
```

Code Explanation – parameter's meaning

- count_2012 is number of document included 2012 words.
- count_jurassic is number of document included Jurassic Park words.



```
for i in range(len(list_2012)):
    if (num_2012[i] != 0) and (anti_num_2012[i] != 0):
        count_2012.append(2)
    elif (num_2012[i] == 0) and (anti_num_2012[i] == 0):
        count_2012.append(0)
    else:
        count_2012.append(1)

for i in range(len(list_jurassic)):
    if (num_jurassic[i] != 0) and (anti_num_jurassic[i] != 0):
        count_jurassic.append(2)
    elif (num_jurassic[i] == 0) and (anti_num_jurassic[i] == 0):
        count_jurassic.append(0)
    else:
        count_jurassic.append(1)
```


Evaluate

- Before F-measure, I assume that I find **movie 2012** using both scripts.
so **relevent document is 2012** and **non-relevent document is Jurassic Park**.

F-measure

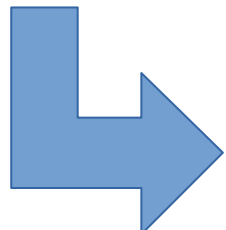
- By above code, now we know TF, N, DF.
- -> so we can calculate TF-IDF like $TF * \log(N/DF)$.
- And then we can calculate F-measure like this.

```
recall = tp/(tp+fn)
precision = tp/(tp+fp)
F_measure = 2*recall*precision/(recall+precision)
print("recall : ",recall,"precision : ",precision)
print("F-measure : ",F_measure)
```

```
recall : 0.5336819440172255 precision : 0.5336819440172255
F-measure : 0.5336819440172255
```

Conclusion

- First, I use all words in both movie script. So F-measure isn't good. When you only use words of high frequency, then you can measure high F-measure.
- And then, I use only two movie script. So when there some words in both script, their tf-idf will be 0.



If $N > 2$

```
recall = tp/(tp+fn)
precision = tp/(tp+fp)
F_measure = 2*recall*precision/(recall+precision)
print("recall : ",recall,"precision : ",precision)
print("F-measure : ",F_measure)
```

```
recall : 0.6847123961857889 precision : 0.7229620006495615
F-measure : 0.7033175355450236
```

It means when I use $N+1$, instead N , then tf-idf won't 0, so I can get more better F-measure.

thanks