

Youtube 크롤링과 텍스트 검색 엔진 비교 분석

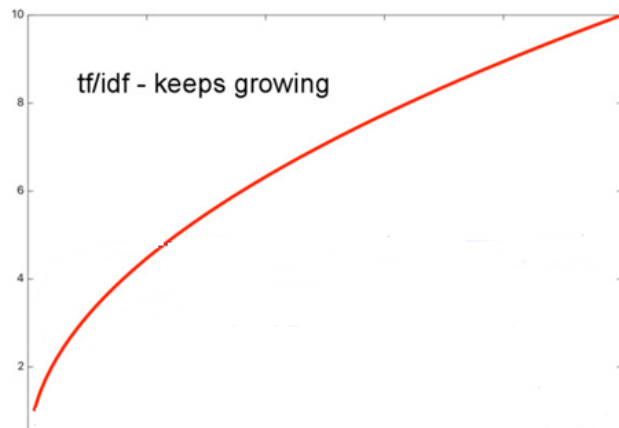
- Youtube Crawling
 - TFIDF vs BM25

텍스트 검색 원리

$$\text{score}(\mathbf{document}) = \sum_{\mathbf{term\ in\ query}} \text{score}(\mathbf{term\ in\ document})$$

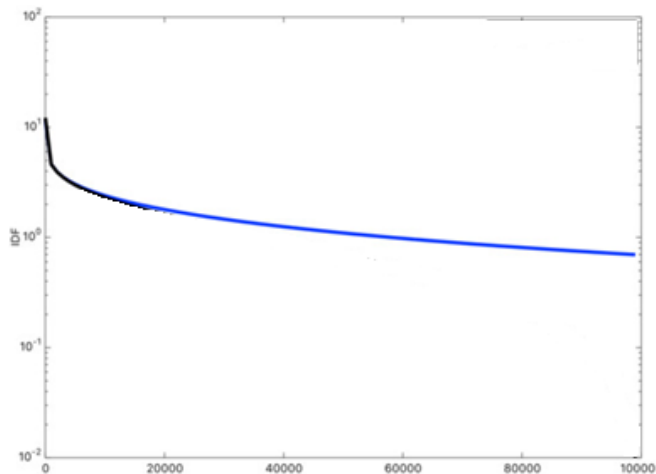
- 사용자가 입력한 쿼리(Query)에 대해 관련 있는 문서를 매칭한 후, 관련성에 따라 정렬하는 과정. 관련성(relevance)은 주어진 쿼리와 매칭되는 문서를 관련된 정도에 따라 수치화한 값으로, 검색 점수(score)라고 부른다.
- 한 단어 뿐만 아니라 여러 단어의 경우도 가능하다 각 단어들에 대한 검색 점수를 더해야 하며 이러한 각 단어를 **term**이라고 한다.

텍스트 검색 3가지 Key



(1) TF(Term Frequency): 단어빈도

많이 등장할수록 점수는 높아진다.



(2) IDF(Inverse Document Frequency): 문서 역빈도

$IDF = 1/DF$ (문서빈도)로 한 단어가 여러 문서에 등장할수록 IDF 값은 줄어든다. 불용어("a", "the", "is")와 같이 거의 항상 등장하는 term이 결과에 영향을 미치지 않게 하는 효과도 가지고 있다.

텍스트 검색 3가지 Key

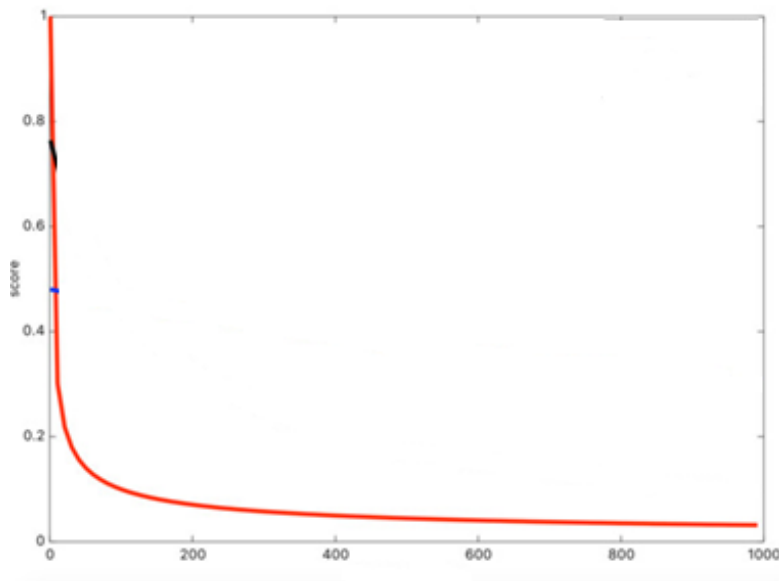
(3) norm: 문서 길이 가중치

문서의 길이가 길수록 점수가 낮아진다.

Ex)

- A가 과일 가게에서 사과와 배를 구매했다.
- B가 과일가게에서 사과를 구매했다.

라는 2개의 문장이 있고 사과를 검색했을 때 위 두 문서의 검색 점수가 같으면 불공평하기 때문에 첫번째 문장에 좀 더 가중치를 줘야하기 때문에 존대한다.



TFIDF vs BM25 - Key 계산 알고리즘 (1)

$$score_{q,d} = norm(q) \times \sum_{t \text{ in } q} \sqrt{tf_{t,d}} \times idf_t^2 \times norm(d, field) \times boost(t)$$

$$bm25(d) = \sum_{t \in q, f_{t,d} > 0} \log \left(1 + \frac{N - df_t + 0.5}{df_t + 0.5} \right) \cdot \frac{f_{t,d}}{f_{t,d} + k} \cdot \left(1 - b + b \frac{l(d)}{avgdl} \right)$$

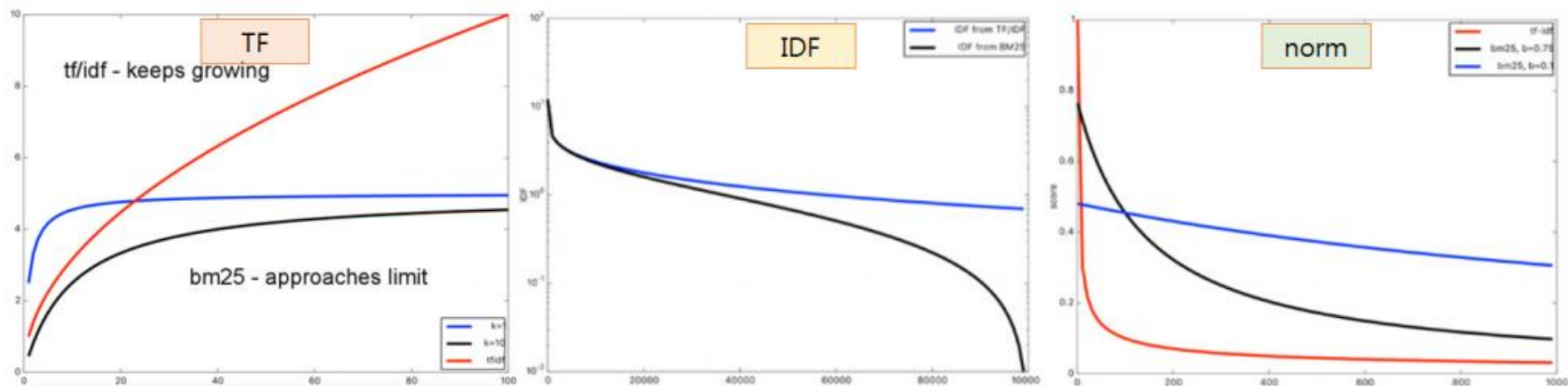


그림. 항목별 TF/IDF와 BM25 비교(출처: [Elasticsearch, Improved Text Scoring with BM25](#))

TFIDF vs BM25 - Key 계산 알고리즘 (2)

TF(Term Frequency): 단어 빈도

- TF: TF의 영향이 줄어든다.
- BM25: TF에서는 단어 빈도가 높아질수록 검색 점수도 지속적으로 높아지는 반면, BM25에서는 특정 값으로 수렴한다. 즉 검색 결과에 좀 더 노출되고 싶기 위해 `[아이폰][아이폰 중고][아이폰 싸게 팔아요]`와 같은 제목의 글을 올려 놓는 경우가 있는데 BM25에서는 이 가중치를 수렴하게 한다. 직관적으로도 `사과가 사과를 맛있게 사과했다.`같은 문장이 있다면 가중치를 줄여야 할 필요가 있어 보인다.

IDF(Inverse Document Frequency): 문서 역빈도

- TF: IDF의 영향이 커진다.
- BM25: BM25에서는 DF가 높아지면 검색 점수가 0으로 급격히 수렴하므로, 불용어가 검색 점수에 영향을 덜 미친다.

norm: 문서 길이 가중치

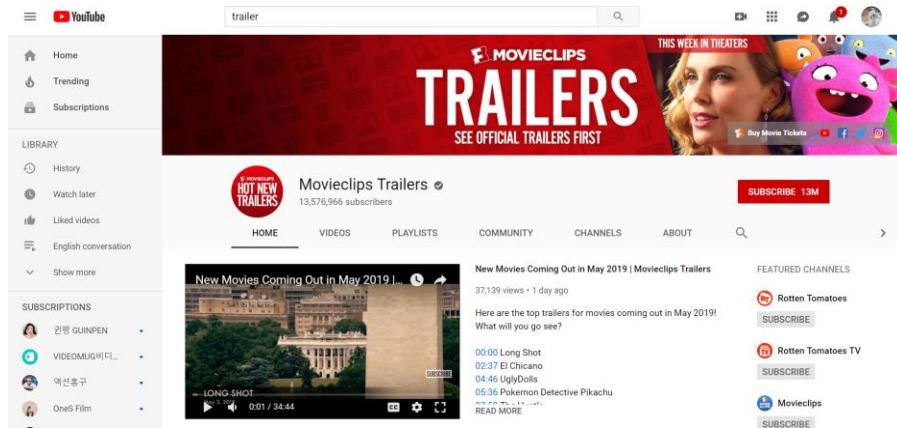
- TF: 문서 길이의 영향이 줄어든다.
- BM25: BM25에서는 문서의 평균 길이(avgdl)를 계산에 사용하며, 문서의 길이가 검색 점수에 영향을 덜 미친다.

모델링 (1) 데이터 구하기

Youtube 영화 예고편 자막 데이터 사용 이유

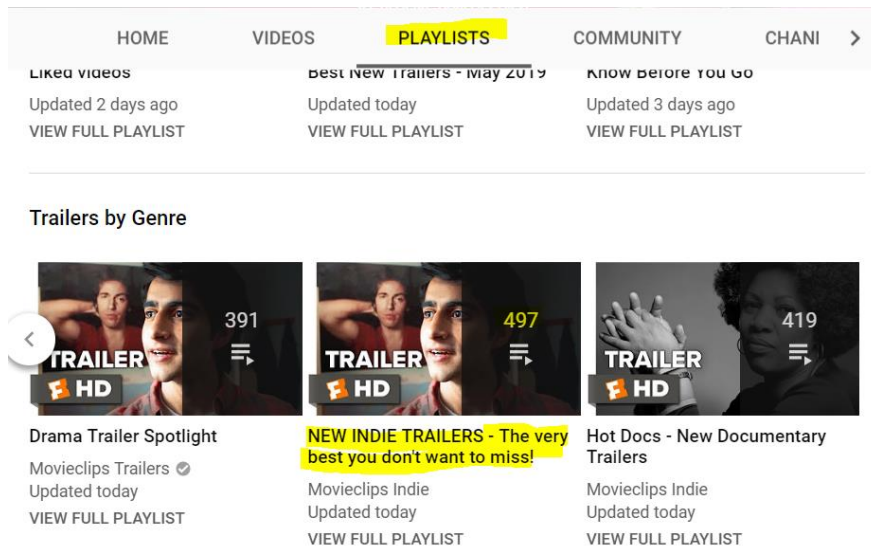
- 더빙 어플리케이션에 존재하는 script 파일과 가장 유사한 data를 찾으려 했습니다.
- 크롤링을 통해 데이터를 무한히 늘릴 수 있고 자동화를 통해 쉽게 얻고자 했습니다.
- 자막마다 start 와 duration 시간이 존재하여 후에 음성데이터에 적용하기 위해 사용했습니다.
- Mp3 파일을 start, duration을 통해 자막마다 분리하여 음성 데이터 feature 값인 mfcc 값을 얻을 수 있습니다.

모델링 (1) 데이터 구하기



<https://www.youtube.com/user/movieclipsTRAILERS>

Movieclips Trailer를 전문으로 제공하는 youtube 페이지에 접속합니다.



https://www.youtube.com/watch?v=ZOWfC09LnZM&list=PLauTLaPMBIIP3FFvODbIpF2bOD_7CSAaC

Playlist에 들어간 후 저는 가장 많은 (497개) video urls가 포함된 "NEW INDIE TRAILERS" 페이지에 접속하였습니다.

모델링 (1) 데이터 구하기

https://www.youtube.com/watch?v=ZOwfC09LnZM&list=PLauTLaPMBIIP3FFvODbIpF2bOD_7CSAaC

Williamsport Web Developer

Enter YouTube playlist id:

NOTE: The Google API for YouTube Data has been substantially changed. You will now need to enter the id of your playlist from the playlist URL. I can get less data about the videos.

NOTE: If this web application is not working for you, you can now buy my [Windows application](#) for only \$5.00. The Windows application has more features and can handle large playlists.

Programming by [Robert S. Robbins](#). This web application backs up the information on your favorite YouTube videos to an Excel spreadsheet. You can then save the Excel file to your hard drive or other media. The reason you may want to export your favorites from the YouTube database is because YouTube has been deleting the accounts of content creators who violate their terms of service. Some accounts have been deleted without warning or explanation. If your account is deleted you will lose all your favorites and have trouble finding those videos again. However, if you have backed up the video information then you can recreate your favorites on a new account.

Instructions
Internet Explorer will pop up a File Download dialog box giving you the option to Open or Save the YouTube-Favorites.xls file. You will need Microsoft Office Excel or a program capable of opening Excel files like [Open Office](#).

Playlist에 주소에 있는 list key를 가져와 옆의 사이트에 접속하여 입력 해주면

<http://www.williamsportwebdeveloper.com/FavBackUp.aspx>

오른쪽과 같이 video url이 들어있는 csv 파일을 받을 수 있습니다. 총 497개의 url이 존재하고 저는 Video URL column만 사용할 것입니다.


A1		YouTube Playlist Export					
	A	B	C	D	E	F	G
1	YouTube Playlist Export						
2	Published	Video URL	Channel	Title	Description		
3	5/2/2019	https://www.youtube.cc	Movieclips	Blinded by	Check out the new trailer for Bl		
4	5/2/2019	https://www.youtube.cc	Movieclips	Toni Morri	Check out the new trailer for To		
5	5/2/2019	https://www.youtube.cc	Movieclips	Ophelia Tr	Check out the new trailer for O		
6	5/2/2019	https://www.youtube.cc	Movieclips	Perfect Tra	Check out the new trailer for Pe		
7	4/25/2019	https://www.youtube.cc	Movieclips	The Third	Check out the new trailer for Th		
8	4/18/2019	https://www.youtube.cc	Movieclips	Skin Trailer	Check out the new trailer for Sk		
9	4/17/2019	https://www.youtube.cc	Movieclips	We Have	Check out the new Trailer for W		
10	4/15/2019	https://www.youtube.cc	Movieclips	Shadow Tr	Check out the new trailer for Sh		
11	4/15/2019	https://www.youtube.cc	Movieclips	Girls of the	Check out the new trailer for Gi		


모델링 (1) 데이터 구하기

Branch: master ▾


New pull request

Create new


 zlsisp54 Add files via upload

 __pycache__


Add files via upload

 data


Add files via upload

 jupyter


Add files via upload

 README.md


Add files via upload

 database_handler_mysql.py


Add files via upload

 database_handler_sqlite.py


Add files via upload

 get_information.py


Add files via upload

 get_urls.ipynb


Add files via upload

 main.py

Add files via upload

 requirements.txt

Add files via upload

 url_handler.py

Add files via upload

<https://github.com/zlsisp54/Youtube-Crawler>

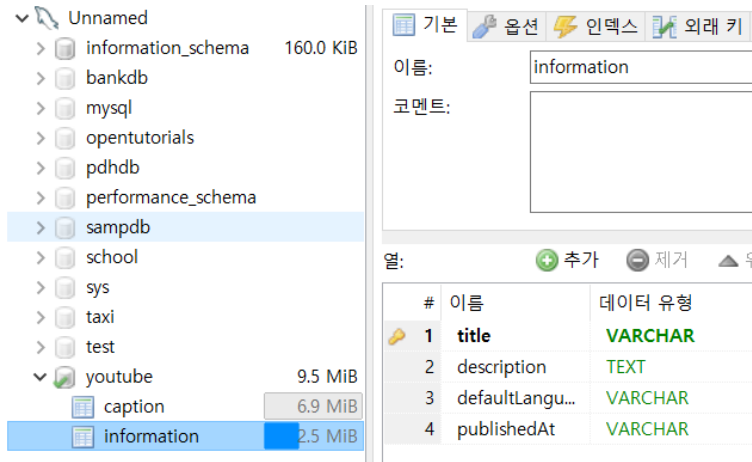
Github에 미리 만들어 놓은 youtube_crawling 모듈을 사용하여 Database에 크롤링한 정보들을 저장합니다.

모델링 (1) 데이터 구하기

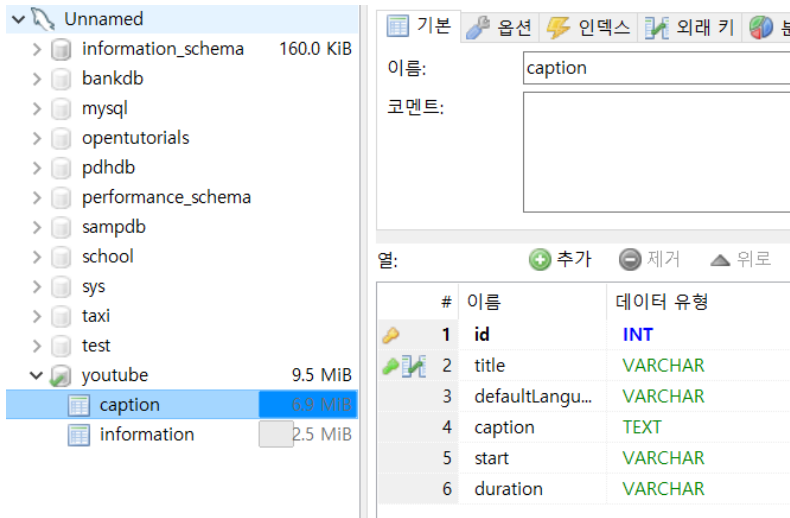
Github에 만들어 놓은 모듈은 1개의 url만을 받아 Database에 저장하는 코드입니다. 그래서 전에 만들어 놓은 csv 파일에서 여러 개의 url들을 for문으로 받아 DB에 저장하는 코드를 따로 만들었습니다.
(Github jupyter 폴더에 get_urls.ipynb 이름으로 저장해 놓았습니다.)

```
1 from url_handler import get_api_information
2 from get_information import download_info, download_caption
3 # from database_handler_mysql import insert_info, insert_caption, close_connection
4 import pymysql.cursors
5
6 import pandas as pd
7 df = pd.read_csv('./data/youtube_urls_indie.csv', header=1)
8 urls = df['Video URL'].tolist()
9
10
11 conn = pymysql.connect(host='localhost',
12                        user='root',
13                        password='111111',
14                        db = 'youtube',
15                        charset='utf8mb4')
16
17 cursor = conn.cursor()
18
19 #from database_handler import close_connection
20
21 for i in urls:
22     url = i
23     apiKey = 'AIzaSyBt3aInFJz9zvryJq-tHe3wq7hNzjJtjK0'
24
25     try:
26         video_id, items = get_api_information(url, apiKey)
27     except:
28         print('Error at get_api_information')
29
30     try:
31         videos_info = download_info(items)
32         videos_caption = download_caption(video_id)
33     except:
34         print('Error at downloading from items(json format)')
35
36     # videos_caption db에 title, defaultlanguage 추가
37     # you can handle which parameters will be included in caption table
38     for x in videos_caption:
39         x.insert(0, videos_info[0])
40         x.insert(1, videos_info[2])
41
42
43
44 with conn.cursor() as cursor:
45     sql = '''
46     INSERT INTO information (title, description, defaultlanguage, publishedAt)
47     VALUES (%s, %s, %s, %s)
48     '''
49     try:
50         cursor.execute(sql, videos_info)
51     except:
52         print('duplicate')
53     conn.commit()
54
55
56 try:
57     with conn.cursor() as cursor:
58         sql = '''
59         INSERT INTO caption (title, defaultlanguage, caption, start, duration)
60         VALUES (%s, %s, %s, %s, %s)
61         '''
62         cursor.executemany(sql, videos_caption)
63         conn.commit()
64     except:
65         print("error")
66
67     print(1)
68     conn.close()
```

모델링 (1) 데이터 구하기

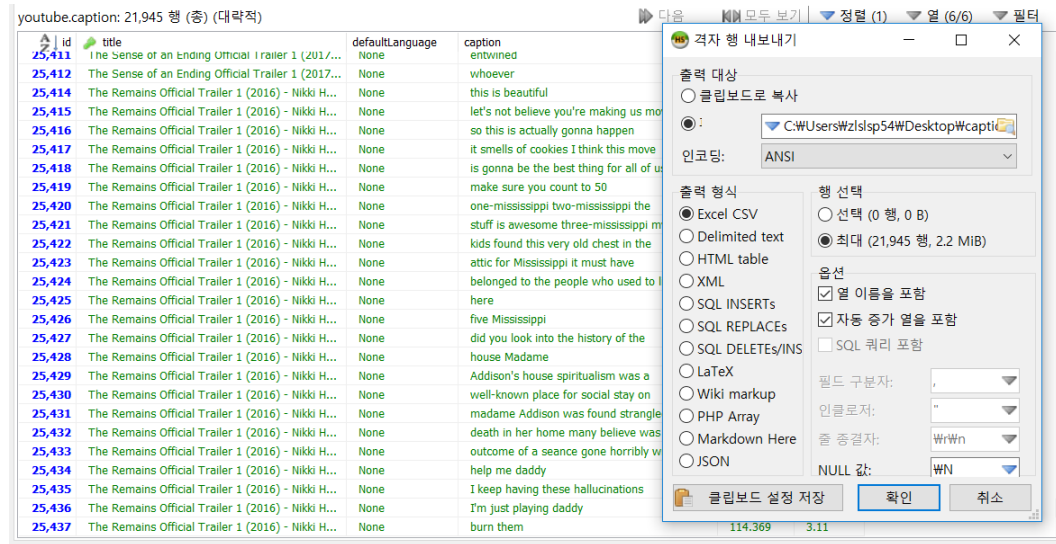


- Database로는 Mysql – HEIDI SQL을 사용했습니다.
- 데이터는 두 개의 Table에 저장했습니다.
- Information 테이블에는 title, description, defaultLanguage, publishedAt 컬럼을 지정했습니다.



- Caption 테이블에는 id, title, defaultLanguage, caption, start, duration 컬럼을 지정했습니다.
- Caption테이블의 title을 외래키로 information의 title과 연동하였습니다.

모델링 (1) 데이터 구하기



The screenshot shows a web browser window displaying a table of YouTube captions. The table has columns for 'id', 'title', 'defaultLanguage', and 'caption'. The 'caption' column contains various subtitles in English. Overlaid on the table is a dialog box titled '격자 행 내보내기' (Export Grid Rows). The dialog box has options for '출력 대상' (Output Target) set to '클립보로 복사' (Copy to Clipboard), '인코딩' (Encoding) set to 'ANSI', and '출력 형식' (Output Format) set to 'Excel CSV'. There are also checkboxes for '열 이름을 포함' (Include column names) and '자동 증가 열을 포함' (Include auto-increment columns), both of which are checked. The dialog box also shows the number of rows to export as '최대 (21,945 행, 2.2 MiB)' (Maximum (21,945 rows, 2.2 MiB)).

id	title	defaultLanguage	caption
25,411	The Sense of an Ending Official Trailer 1 (2017...	None	entwined
25,412	The Sense of an Ending Official Trailer 1 (2017...	None	whoever
25,414	The Remains Official Trailer 1 (2016) - Nikki H...	None	this is beautiful
25,415	The Remains Official Trailer 1 (2016) - Nikki H...	None	let's not believe you're making us mo
25,416	The Remains Official Trailer 1 (2016) - Nikki H...	None	so this is actually gonna happen
25,417	The Remains Official Trailer 1 (2016) - Nikki H...	None	it smells of cookies I think this move
25,418	The Remains Official Trailer 1 (2016) - Nikki H...	None	is gonna be the best thing for all of us
25,419	The Remains Official Trailer 1 (2016) - Nikki H...	None	make sure you count to 50
25,420	The Remains Official Trailer 1 (2016) - Nikki H...	None	one-mississippi two-mississippi the
25,421	The Remains Official Trailer 1 (2016) - Nikki H...	None	stuff is awesome three-mississippi m
25,422	The Remains Official Trailer 1 (2016) - Nikki H...	None	kids found this very old chest in the
25,423	The Remains Official Trailer 1 (2016) - Nikki H...	None	attic for Mississippi it must have
25,424	The Remains Official Trailer 1 (2016) - Nikki H...	None	belonged to the people who used to l
25,425	The Remains Official Trailer 1 (2016) - Nikki H...	None	here
25,426	The Remains Official Trailer 1 (2016) - Nikki H...	None	five Mississippi
25,427	The Remains Official Trailer 1 (2016) - Nikki H...	None	did you look into the history of the
25,428	The Remains Official Trailer 1 (2016) - Nikki H...	None	house Madame
25,429	The Remains Official Trailer 1 (2016) - Nikki H...	None	Addison's house spiritualism was a
25,430	The Remains Official Trailer 1 (2016) - Nikki H...	None	well-known place for social stay on
25,431	The Remains Official Trailer 1 (2016) - Nikki H...	None	madame Addison was found strangle
25,432	The Remains Official Trailer 1 (2016) - Nikki H...	None	death in her home many believe was
25,433	The Remains Official Trailer 1 (2016) - Nikki H...	None	outcome of a seance gone horribly w
25,434	The Remains Official Trailer 1 (2016) - Nikki H...	None	help me daddy
25,435	The Remains Official Trailer 1 (2016) - Nikki H...	None	I keep having these hallucinations
25,436	The Remains Official Trailer 1 (2016) - Nikki H...	None	I'm just playing daddy
25,437	The Remains Official Trailer 1 (2016) - Nikki H...	None	burn them

- Caption 컬럼을 클릭하여 공백, 영어가 아닌 문자 열, [Music] 같은 필요 없는 자막들을 삭제해 주었습니다.
- Caption 테이블의 caption 컬럼에 총 23,437개의 자막 데이터가 저장되어 있습니다. 이미 저장해 놓은 1000개와 INDIE playlist로 크롤링 해온 약 10000개가 더해져 약 25000개가 저장되었습니다.
- 약 500 video에 1000개의 자막이 저장되었으니 한 비디오 당 20개 정도의 자막이 존재하는 걸 알 수 있습니다.
- 화면을 오른쪽 클릭하여 격자 행 내보내기를 선택하고 caption table을 csv파일로 export 해줍니다.

모델링 (2) TFIDF 구현

zlsisp54 Add files via upload		
__pycache__		Add files via upload
2DUB_tfidf.ipynb		Add files via upload
Extract_top.py		Add files via upload
Get_data.py		Add files via upload
PreProcessing.py		Add files via upload
TFIDF_Hardcoding.py		Add files via upload
main.py		Add files via upload

https://github.com/zlsisp54/Searching_TFIDF

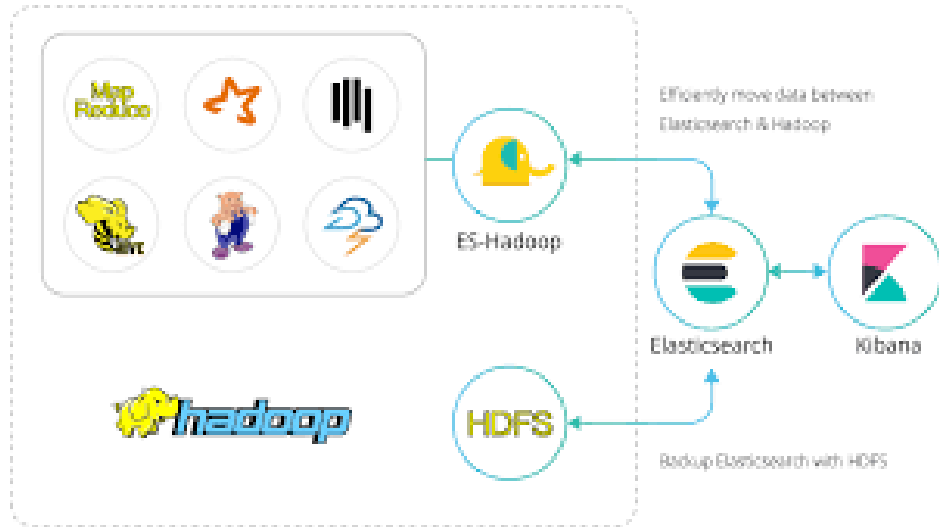
- Github에 TFIDF 모델을 구현하여 올려 두었습니다.
- TFIDF로 벡터화 하여 Cosine similarity를 통해 유사한 단어, 문장을 찾아줍니다.
- Top5 문장을 도출해줍니다.

모델링 (2) TFIDF 구현

```
(tensorflow) C:\Users\zls\sp54\sean\TFIDF>python main.py
Input sentence: I love you so much
1. so much | similarity: 0.7812243268483877
2. so much | similarity: 0.7812243268483877
3. do you know how much I love you I love | similarity: 0.7228954630084238
4. I love you guys so much being when | similarity: 0.67888164807046
5. so much I liked you so much I'm always | similarity: 0.665047479967823
```

- "I love you so much"를 검색해 봅니다.
- "So much"가 가장 유사한 단어로 인식되어 보여집니다.
- "I love you"가 중요한데.. 결과가 별로 좋아 보이지 않습니다.

모델링 (3) BM25 구현 - Elasticsearch



- BM25 알고리즘을 사용하여 유사한 문장을 찾아봅니다.
- 현재 아파치 루신의 BM25 검색 엔진 알고리즘을 사용하고 있는 Elasticsearch를 사용합니다.
- 엘라스틱서치란 아파치 루신을 기반으로 개발된 오픈소스 분산 검색 엔진(서버)입니다.
- 특징으로는 분산+확장성,고가용성, 전문 검색, RESTful api가 존재합니다.

모델링 (3) BM25 구현 - Elasticsearch

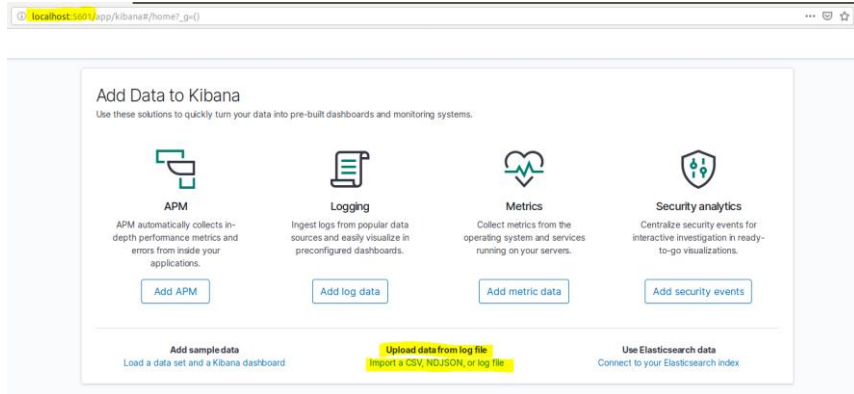
```
zls1sp54@ubuntu:~/Elastic$ ls  
elasticsearch kibana logstash
```

```
:~/Elastic/elasticsearch$ bin/elasticsearch
```

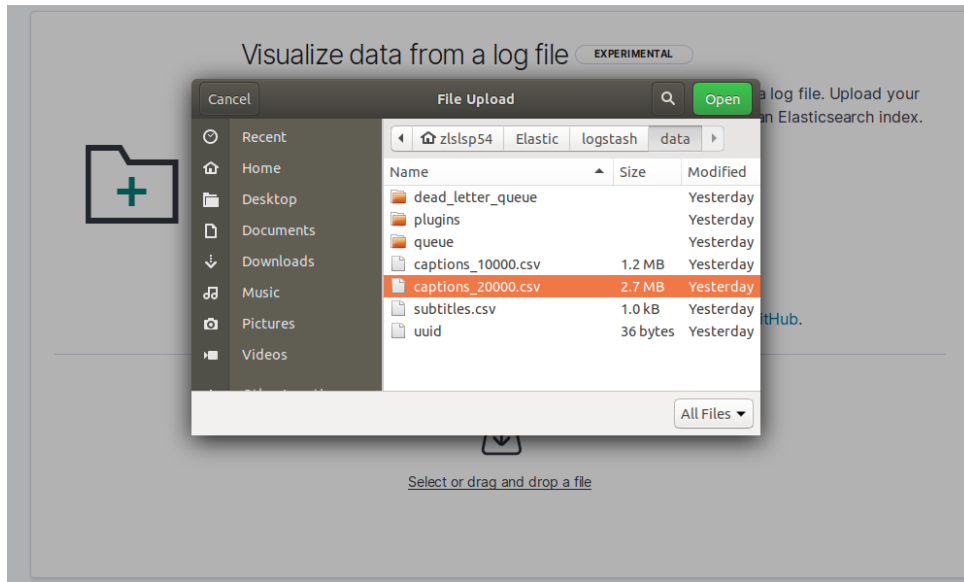
```
:~/Elastic/kibana/kibana-7.0.1-linux-x86_64$ bin/kibana
```

- 리눅스 환경에서 Elasticsearch, Kibana, Logstash를 기본적으로 설치해줍니다.
- Elasticsearch를 실행 해줍니다.
- Kibana도 실행해 줍니다.

모델링 (3) BM25 구현 - Elasticsearch



- Localhost:5601에 접속하면 Kibana에 접속할 수 있습니다.
- Upload data from log file – csv를 클릭하여 들어갑니다.
- 전에 만들어 놓은 caption 테이블의 csv파일을 import 해줍니다.



모델링 (3) BM25 구현 - Elasticsearch

```
6 ^ }  
7  
8 GET /_cat/indices?v  
9
```

	health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
2	yellow	open	subtitles	9H6kCSeDSH-fPq4aaJwBRw	1	1	9	0	6.3kb	6.3kb
3	yellow	open	customer	Xu8MOVY-Qgib5AE2dJK9kg	1	1	0	0	283b	283b
4	yellow	open	subtitles_20000	ereVzlsxTUqM4Lu7feMMrg	1	1	21945	0	3.2mb	3.2mb
5	yellow	open	movie_subtitles	o9pe1IWOSLuiPy2r3bpRmg	1	1	10896	0	1.6mb	1.6mb
6	green	open	.kibana_task_manager	bYWSV336QA-H5cCekQ0VRg	1	0	2	0	29.5kb	29.5kb
7	green	open	.kibana_1	_PTosAQMRvi8fKzLNj-kAg	1	0	7	1	39.3kb	39.3kb

- Dev Tools 메뉴에서 Consol로 들어옵니다.
- Get /_cat/indices?v 명령어로 index들을 불러옵니다.
- 저는 subtitles_20000이라는 이름으로 저장해 놓았습니다. Index 이름에서 확인할 수 있습니다.
- Docs.count를 통해 doc 양을 확인할 수 있습니다. 21945개가 존재합니다.

모델링 (3) BM25 구현 - Elasticsearch

```
GET movie_subtitles/_search
{
  "query": {
    "match": {
      "caption": "I love you so much"
    }
  },
  "explain": true
}
```

```
{
  "title": "Beautiful Boy Trailer #1 (2018) | Movieclips Trailers",
  "defaultlanguage": "None",
  "caption": "do you know how much I love you I love",
  "start": "123.75",
  "duration": "6.39"
},
"_explanation": {
  "match": {
    "caption": "do you know how much I love you I love"
  }
},
{
  "_shard": "[movie_subtitles][0]",
  "_node": "5HEwRRdDSNKVCKQXwitt4A",
  "_index": "movie_subtitles",
  "_type": "_doc",
  "_id": "r04zfGoBdVPoAHid-_0H",
  "_score": 11.66747,
  "_source": {
    "id": "3213",
    "title": "Head Full of Honey Trailer #1 (2018) | Movieclips Trailers",
    "defaultlanguage": "None",
    "caption": "did I ever tell you how much I love you",
    "start": "14.66",
    "duration": "10.85"
  },
  "_explanation": {
    "match": {
      "caption": "did I ever tell you how much I love you"
    }
  }
}
```

- Match api를 이용해 query를 보냅니다.
 - TFIDF에서와 같이 "I love you so much"를 주어 BM25알고리즘을 사용한 가장 비슷한 문자열을 찾아줍니다.
 - 결과로 top2를 확인해 봅니다.
1. "do you know how much I love you I love"
 2. "did I ever tell you how much I love you"

결과

```
(tensorflow) C:\Users\zlsisp54\sean\TFIDF>python main.py
Input sentence: I love you so much
1. so much | similarity: 0.7812243268483877
2. so much | similarity: 0.7812243268483877
```

```
{
  "caption": "do you know how much I love you I love",
  "start": "123.75",
  "duration": "6.39",
},
 "_explanation": {
},
},
},
 "_shard": "[movie_subtitles][0]",
 "_node": "SHEwRRdDSNKVCKQXwitt4A",
 "_index": "movie_subtitles",
 "_type": "doc",
 "_id": "r04zfGoBdVPoAHID-0H",
 "_score": 11.66747,
 "_source": {
  "id": "3213",
  "title": "Head Full of Honey Trailer #1 (2018) | Movieclip",
  "defaultLanguage": "None",
  "caption": "did I ever tell you how much I love you",
  "start": "14.66"
```

- TFIDF에서는 “I love you so much”와 가장 비슷한 문자열로 “so much”를 도출했는데 이는 I와 love라는 문자가 다큐먼트 전체에 많이 존재하여 IDF값이 높아져 불용어 처리가 되어 불용어가 검색 점수에 많은 영향을 끼쳐 I, love라는 단어가 없는 문자열을 나타내 준 것 같습니다.
- BM25에서는 DF가 높아지면 검색 점수가 0으로 급격히 수렴하므로 불용어가 검색 점수에 영향을 덜 미칩니다.
- 또한 BM25에서 TF 점수에서도 단어 빈도가 높아지면 특정 값으로 수렴하고 norm을 문서의 평균 길이를 계산에 사용하여 문서의 길이가 검색 점수에 영향을 덜 미치는 장점도 가지고 있습니다. (이 알고리즘들은 한 다큐먼트에 엄청 많은 문자열들이 저장되었을 때 효과가 있습니다. 지금 같은 한 문장 단위에서는 효과가 없습니다.)
- 직관적으로 봐도 BM25를 사용한 문장들이 더 높은 상관관계가 있는 것 같고 알고리즘 적으로도 더 좋은 성능을 보입니다.

결과 - 다른 예제

"Never took it done"

"I have no idea"

"I would like to"

```
1 GET _search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
8 GET _cat/indices?v
9
10 GET movie_subtitles/_search
11 {
12   "query": {
13     "match": {
14       "caption": "never took it done"
15     }
16   },
17   "explain": true
18 }
19
20 GET /subtitles/ search
```

```

21     "type": "doc",
22     "id": "tE4zfGoBdVPoAHd-vLw",
23     "score": 2.249528,
24     "source": " ",
25     "id": "5",
26     "title": "Triple Frontier Trailer #2 (2019) | Movieclips
27     "defaultLanguage": "en",
28     "caption": "never took it done",
29     "start": "20.46",
30     "duration": "4.079"
31 },
32     "_explanation": "( )
33 },
294 {
295     "shard": "[movie_subtitles][0]",
296     "node": "5HEwRRdDSNvCKQXwitt4A",
297     "index": "movie_subtitles",
298     "type": "doc",
299     "id": "8U8zfGoBdVPoAHd-wss",
300     "score": 8.978611,
301     "source": " ",
302     "id": "7398",
303     "title": "Hearts Beat Loud Trailer #1 | Movieclips Indie
304     "defaultLanguage": "None",
305     "caption": "never done this before this is just what",
306     "start": "124.479",
307     "duration": "3.87"
308 },
309     "_explanation": "( )
310 },
444 {
445     "shard": "[movie_subtitles][0]",
446     "node": "5HEwRRdDSNvCKQXwitt4A",
447     "index": "movie_subtitles",
448     "type": "doc",
449     "id": "Dk8OfGoBdVPoAHdCRqJ",
450     "score": 8.744799,
451     "source": " ",
452     "id": "11464",
453     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
454     "defaultLanguage": "None",
455     "caption": "it will never end",
456     "start": "10.46",
457     "duration": "3.87"
458 },
459     "_explanation": "( )
460 },
461     "shard": "[movie_subtitles][0]",
462     "node": "5HEwRRdDSNvCKQXwitt4A",
463     "index": "movie_subtitles",
464     "type": "doc",
465     "id": "Dk8OfGoBdVPoAHdCRqJ",
466     "score": 8.744799,
467     "source": " ",
468     "id": "11464",
469     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
470     "defaultLanguage": "None",
471     "caption": "it will never end",
472     "start": "10.46",
473     "duration": "3.87"
474 },
475     "_explanation": "( )
476 },
477     "shard": "[movie_subtitles][0]",
478     "node": "5HEwRRdDSNvCKQXwitt4A",
479     "index": "movie_subtitles",
480     "type": "doc",
481     "id": "Dk8OfGoBdVPoAHdCRqJ",
482     "score": 8.744799,
483     "source": " ",
484     "id": "11464",
485     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
486     "defaultLanguage": "None",
487     "caption": "it will never end",
488     "start": "10.46",
489     "duration": "3.87"
490 },
491     "_explanation": "( )
492 },
493     "shard": "[movie_subtitles][0]",
494     "node": "5HEwRRdDSNvCKQXwitt4A",
495     "index": "movie_subtitles",
496     "type": "doc",
497     "id": "Dk8OfGoBdVPoAHdCRqJ",
498     "score": 8.744799,
499     "source": " ",
500     "id": "11464",
501     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
502     "defaultLanguage": "None",
503     "caption": "it will never end",
504     "start": "10.46",
505     "duration": "3.87"
506 },
507     "_explanation": "( )
508 },
509     "shard": "[movie_subtitles][0]",
510     "node": "5HEwRRdDSNvCKQXwitt4A",
511     "index": "movie_subtitles",
512     "type": "doc",
513     "id": "Dk8OfGoBdVPoAHdCRqJ",
514     "score": 8.744799,
515     "source": " ",
516     "id": "11464",
517     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
518     "defaultLanguage": "None",
519     "caption": "it will never end",
520     "start": "10.46",
521     "duration": "3.87"
522 },
523     "_explanation": "( )
524 },
525     "shard": "[movie_subtitles][0]",
526     "node": "5HEwRRdDSNvCKQXwitt4A",
527     "index": "movie_subtitles",
528     "type": "doc",
529     "id": "Dk8OfGoBdVPoAHdCRqJ",
530     "score": 8.744799,
531     "source": " ",
532     "id": "11464",
533     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
534     "defaultLanguage": "None",
535     "caption": "it will never end",
536     "start": "10.46",
537     "duration": "3.87"
538 },
539     "_explanation": "( )
540 },
541     "shard": "[movie_subtitles][0]",
542     "node": "5HEwRRdDSNvCKQXwitt4A",
543     "index": "movie_subtitles",
544     "type": "doc",
545     "id": "Dk8OfGoBdVPoAHdCRqJ",
546     "score": 8.744799,
547     "source": " ",
548     "id": "11464",
549     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
550     "defaultLanguage": "None",
551     "caption": "it will never end",
552     "start": "10.46",
553     "duration": "3.87"
554 },
555     "_explanation": "( )
556 },
557     "shard": "[movie_subtitles][0]",
558     "node": "5HEwRRdDSNvCKQXwitt4A",
559     "index": "movie_subtitles",
560     "type": "doc",
561     "id": "Dk8OfGoBdVPoAHdCRqJ",
562     "score": 8.744799,
563     "source": " ",
564     "id": "11464",
565     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
566     "defaultLanguage": "None",
567     "caption": "it will never end",
568     "start": "10.46",
569     "duration": "3.87"
570 },
571     "_explanation": "( )
572 },
573     "shard": "[movie_subtitles][0]",
574     "node": "5HEwRRdDSNvCKQXwitt4A",
575     "index": "movie_subtitles",
576     "type": "doc",
577     "id": "Dk8OfGoBdVPoAHdCRqJ",
578     "score": 8.744799,
579     "source": " ",
580     "id": "11464",
581     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
582     "defaultLanguage": "None",
583     "caption": "it will never end",
584     "start": "10.46",
585     "duration": "3.87"
586 },
587     "_explanation": "( )
588 },
589     "shard": "[movie_subtitles][0]",
590     "node": "5HEwRRdDSNvCKQXwitt4A",
591     "index": "movie_subtitles",
592     "type": "doc",
593     "id": "Dk8OfGoBdVPoAHdCRqJ",
594     "score": 8.744799,
595     "source": " ",
596     "id": "11464",
597     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
598     "defaultLanguage": "None",
599     "caption": "it will never end",
600     "start": "10.46",
601     "duration": "3.87"
602 },
603     "_explanation": "( )
604 },
605     "shard": "[movie_subtitles][0]",
606     "node": "5HEwRRdDSNvCKQXwitt4A",
607     "index": "movie_subtitles",
608     "type": "doc",
609     "id": "Dk8OfGoBdVPoAHdCRqJ",
610     "score": 8.744799,
611     "source": " ",
612     "id": "11464",
613     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
614     "defaultLanguage": "None",
615     "caption": "it will never end",
616     "start": "10.46",
617     "duration": "3.87"
618 },
619     "_explanation": "( )
620 },
621     "shard": "[movie_subtitles][0]",
622     "node": "5HEwRRdDSNvCKQXwitt4A",
623     "index": "movie_subtitles",
624     "type": "doc",
625     "id": "Dk8OfGoBdVPoAHdCRqJ",
626     "score": 8.744799,
627     "source": " ",
628     "id": "11464",
629     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
630     "defaultLanguage": "None",
631     "caption": "it will never end",
632     "start": "10.46",
633     "duration": "3.87"
634 },
635     "_explanation": "( )
636 },
637     "shard": "[movie_subtitles][0]",
638     "node": "5HEwRRdDSNvCKQXwitt4A",
639     "index": "movie_subtitles",
640     "type": "doc",
641     "id": "Dk8OfGoBdVPoAHdCRqJ",
642     "score": 8.744799,
643     "source": " ",
644     "id": "11464",
645     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
646     "defaultLanguage": "None",
647     "caption": "it will never end",
648     "start": "10.46",
649     "duration": "3.87"
650 },
651     "_explanation": "( )
652 },
653     "shard": "[movie_subtitles][0]",
654     "node": "5HEwRRdDSNvCKQXwitt4A",
655     "index": "movie_subtitles",
656     "type": "doc",
657     "id": "Dk8OfGoBdVPoAHdCRqJ",
658     "score": 8.744799,
659     "source": " ",
660     "id": "11464",
661     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
662     "defaultLanguage": "None",
663     "caption": "it will never end",
664     "start": "10.46",
665     "duration": "3.87"
666 },
667     "_explanation": "( )
668 },
669     "shard": "[movie_subtitles][0]",
670     "node": "5HEwRRdDSNvCKQXwitt4A",
671     "index": "movie_subtitles",
672     "type": "doc",
673     "id": "Dk8OfGoBdVPoAHdCRqJ",
674     "score": 8.744799,
675     "source": " ",
676     "id": "11464",
677     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
678     "defaultLanguage": "None",
679     "caption": "it will never end",
680     "start": "10.46",
681     "duration": "3.87"
682 },
683     "_explanation": "( )
684 },
685     "shard": "[movie_subtitles][0]",
686     "node": "5HEwRRdDSNvCKQXwitt4A",
687     "index": "movie_subtitles",
688     "type": "doc",
689     "id": "Dk8OfGoBdVPoAHdCRqJ",
690     "score": 8.744799,
691     "source": " ",
692     "id": "11464",
693     "title": "Shot Caller Trailer #1 (2017) | Movieclips Indi
694     "defaultLanguage": "None",
695     "caption": "it will never end",
696     "start": "10
```

```
1 GET _search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
8 GET _cat/indices?v
9
10 GET movie_subtitles/_search
11 {
12   "query": {
13     "match": {
14       "caption": "I have no idea"
15     },
16   },
17   "explain": true
18 }
19
20 GET /subtitles/_search
```

```

21     "type": "doc",
22     "id": "t04zfG0dB/POA#ID_-4P",
23     "score": 12.20661,
24     "source": {
25       "id": "3513",
26       "title": "Destroyer Trailer #1 (2018) | Movieclips Trailers",
27       "defaultLanguage": "None",
28       "caption": "No ID, no idea",
29       "start": "37.34",
30       "duration": "1.7",
31     },
32     "explanation": { [REDACTED] },
33   },
34   {
35     "shard": "[movie_subtitles][0]",
36     "node": "SHEwRdDSMKVCKQXwitt4A",
37     "index": "movie_subtitles",
38     "type": "doc",
39     "id": "uU4zfG0dB/POA#ID_vLM",
40     "score": 12.15511,
41     "source": {
42       "id": "54",
43       "title": "The Hustle Trailer #1 (2019) | Movieclips Trailers",
44       "defaultLanguage": "en",
45       "caption": "sisters-in-arms I have no idea how small",
46       "start": "29.519",
47       "duration": "6.79",
48     },
49     "explanation": { [REDACTED] },
50   },
51   {
52     "shard": "[movie_subtitles][0]",
53     "node": "SHEwRdDSMKVCKQXwitt4A",
54     "index": "movie_subtitles",
55     "type": "doc",
56     "id": "eU4zfG0dB/POA#ID_-kA",
57     "score": 12.15511,
58     "source": {
59       "id": "1974",
60       "title": "Fighting With My Family Final Trailer (2019) | Mo",
61       "defaultLanguage": "None",
62       "caption": "I have no idea who I'm supposed to be",
63       "start": "78.28",
64     },
65   },
66 ]

```

```

1 GET _search
2 {
3   "query": {
4     | "match_all": {}
5   }
6 }
7
8 GET _cat/indices?v
9
10 GET movie_subtitles/_search
11 {
12   "query": {
13     "match": {
14       | "caption": "I would like to"
15     }
16   },
17   "explain": true
18 }
19
20 GET /subtitles/_search

```

```

21     "type": "doc",
22     "id": "rk8zfG0dBVPoAHD-wYw",
23     "score": 10.399589,
24     "source": {
25       "id": "5852",
26       "title": "The Children Act Trailer #1 (2010) | Movieclips Trailers",
27       "defaultlanguage": "None",
28       "caption": "case i would like to hear from Adam",
29       "start": "45.84",
30       "duration": "6.03"
31     },
32     "explanation": {
33       "shard": "[movie_subtitles][0]",
34       "node": "SHEwRRdDSNKVCQXwitt4A",
35       "index": "movie_subtitles",
36       "type": "doc",
37       "id": "NU8zfG0dBVPoAHD-wg4",
38       "score": 10.399589,
39       "source": {
40         "id": "6295",
41         "title": "The Children Act International Trailer #1 (2010) | Movieclips Trailers",
42         "defaultlanguage": "None",
43         "caption": "into their sister i would like to hear",
44         "start": "50.3",
45         "duration": "3.84"
46       },
47       "explanation": {
48         "shard": "[movie_subtitles][0]",
49         "node": "SHEwRRdDSNKVCQXwitt4A",
50         "index": "movie_subtitles",
51         "type": "doc",
52         "id": "AK8zfG0dBVPoAHD-xM-",
53         "score": 10.399589,
54         "source": {
55           "id": "9376",
56           "title": "Roman J. Israel, Esq. Trailer #1 (2017) | Movieclips Trailers",
57           "defaultlanguage": "en",
58           "caption": "try though i would like to address it",
59           "start": "6.25",
60           "duration": "4.17"
61         }
62       }
63     }
64   }
65 }

```

1. never took it done
2. never done this before this is just what
3. It will never end

1. No ID, no idea
2. Sisters-in-arms I have no idea how small
3. I have no idea who I'm supposed to be

1. Case I would like to hear from Adam
2. Into their sister I would like to hear
3. Try though I would like to address it

결과 - 요약

TFIDF

- 동일한 단어가 계속해서 score에 영향을 끼쳐 tfidf가 계속해서 증가할 수 있다.
- 짧은 fields(문장)의 score가 높게 측정된다.

BM25

- term frequency의 영향을 제한하여 동일한 단어가 계속해서 영향을 끼치는 데에 limit을 건다.
- Norm을 통해 scale field를 평균으로 계산하여 짧은 문장의 score가 높게 측정되는 것을 막아준다.

Elasticsearch 추후 개발 목표

- 한 단어 데이터셋에 한 단어 이상 검색 결과를 가져오는 법
- 두 단어 이상 데이터 셋에서 문장으로 검색하는 법

Git Blog에 더 자세한 요약 정리

<https://2meu.github.io/2dub/2DUB/#>