Whatcha Watchin'?

Kuan-Yu Chen, Cheng Chen, Zhiheng Wang

The objective of this project is to build an advanced information retrieval system to support multiple search methods for people who are interested in TED Talks. The TED Talks is popular by its highly reputed speakers and their inspiring speeches. It is not only a shortcut for people who are interested in learning how to deliver a speech but also an educational material which encourage people to think out of the box. It has high impacts on people from all over the world, and many TED Talks are translated into multiple languages.

However, we notice that *TED* does not support content based search and this may make people who are interested in a specific talk struggling. This project uses TED Talks to build a high performance information retrieval system with multiple features to search, such as free text search which supports title, transcript, and description with doubled weight on title, search by speakers, and search by talks' length.

This system supports a tf-idf relevance based recommendation function which shows users the similar TED Talks on the result pages. The main relevance calculation for results' ranking is based on elasticSearch's default method of BM25.

Requirements

To run all the functions of the system, you should have the following:

General Dependency:

- Python 3.5+ (required) : used to run the code
- Elasticsearch 6.7 (required) : used to implement the search engine
- Google cloud account (required) : used to scrape data from youtube
- Youtube data V3 API : used to scrape data from youtube
- Word cloud API : used to implement the word relevance map
- Bootstrap : used to implement the web

Python Libraries Dependency:

ElasticSearch

```
pip install elasticsearch
```

Flask

```
pip install -U Flask
```

Elasticsearch-dsl

```
pip install elasticsearch-dsl
```

Wordcloud

```
pip install wordcloud
```

Googleapiclient

```
pip install google-api-python-client
```

Google-auth-oauthlib

```
pip install google-auth google-auth-oauthlib google-auth-httplib2
```

Beautifulsoup4

```
pip install beautifulsoup4
```

Requests

```
pip install requests
```

Scikit-learn

```
pip install scikit-learn
```

Json

Re

The following tools are optional, but can improve the efficiency of work:

- GitHub

General Layout

The code is divided into three parts, scraper, search engine and frontend.

1. Scraper:

TedTalks_scrape.py: scrape the data from TED
YouTube_scrape.py: scrape the data from YouTube and merge it with the data from TED to build the full corpus

The following data are scraped for building this information retrieval system:
"Title": the title of this TED Talk
"Description": the short description of this TED Talk
"Date": the date of this TED Talk being published
"Thumbnails": a image of the speaker of this TED Talk
"Tags": a set of descriptive tags of this TED Talk
"Num_views": number of views of this TED Talk from *TED* and *YouTube*
"Num_comments": number of comments of this TED Talk from *TED* and *YouTube*
"num_transcripts": number of languages available for transcripts/subtitles in this TED Talk
“Categories”: feedback of users on *TED*

“Transcript”: English transcript of this TED Talk
“YouTube_likeCount”: number of likes from *YouTube*
“YouTube_dislikeCount”: number of dislikes from *YouTube*
“YouTube_favoriteCount”: number of favorite from *YouTube*
“Comments”: comments from *YouTube*

get_authenticated_service(): used for accessing to the API
search_by_channelId(service, **kwargs): run API to query over YouTube by TED Talks channel id
write_video_id_list_to_file(outfile, service, **kwargs): call API to write the YouTube video id defined by YouTube into the outfile separated by \n
write_videos_statistics_to_file(id_list, outfile, service, **kwargs): call API to write the videos statistics (viewCount, likeCount, dislikeCount, favoriteCount, commentCount) to file
get_video_comments() call API to get comments based on YouTube video Id

2. Search Engine:

/ ted_index.py
Build index on title, speaker, description, transcript, duration using ES.
Normalization: text_analyzer (stop, lowercase, porter_stem), nnp_analyzer (lowercase)
Calculate TF-IDF between all documents for recommendation system.

```
from sklearn.feature_extraction.text import TfidfVectorizer

docs = [ v["transcript"]+v["description"]+v["title"]+' '.join(v["tags"]) for k, v in talks.items() ]
vect = TfidfVectorizer(min_df=1)
tfidf = vect.fit_transform(docs)
comp = (tfidf * tfidf.T).A

for k, v in talks.items():
    arr = comp[int(k)-1]
    rec = arr.argsort()[-5:][::-1]
    talks[k]["rec"] = np.array2string(rec[1:])[1:-1]
```

/ ted_query.py
Allocate different weight to the fields: 2x for title and 1x for description and transcript.
Highlighting keywords of query and return parts of the results.

Calculate relevance and popularity for display.

```
result['relevance'] = '%.2f'%((hit.meta.score / response.hits.max_score)*100)

for key, result in resultList.items():
    pop = 1/2 * (result['num_views']/max_view) + 1/2 * (result['num_comments']/max_comment)
    pop = 100 * pop
    result['popularity'] = '%.2f'%pop
```

Jinha Lee: Reach into the computer and grab a pixel

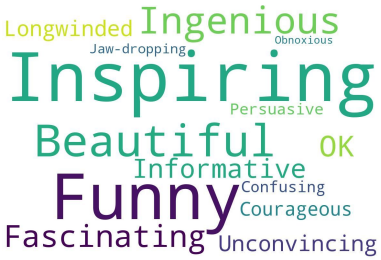
Relevance: 100.00% Popularity : 81.43%

As he demonstrates in this short, gasp-inducing talk, his ideas include a pen that penetrates into a screen to draw 3D models and SpaceTop, a computer desktop prototype that lets you reach through the screen to manipulate digital objects.

(Posted on 2013-02-27 | 6 mins)

https://www.ted.com/talks/jinha_lee_a_tool_that_lets_you_touch_pixels

Generate word cloud for selected video based on scraped data for better visualization.



Integrate relevant Youtube comments.

Ian Ritchie: The day I turned down Tim Berners-Lee

Well we all know the World Wide Web has absolutely transformed publishing, broadcasting, commerce and social connectivity, but where did it all come from? And I'll quote three people: Vannevar Bush, Doug Engelbart and Tim Berners-Lee. So let's just run through these guys. This is Vannevar Bush. Vannevar Bush was the U.S. government's chief scientific adviser during the war. And in 1945, he published an article in a magazine called Atlantic Monthly. And the article was called "As We May Think." And what Vannevar Bush was saying was the way we use information is broken. We don't work in terms of libraries and catalog systems and so forth. The brain works by association. With one item in its thought, it snaps instantly to the next item. And the way information is structured is totally incapable of keeping up with this process. And so he suggested a

Imagine it's late 1990, and you've just met a nice young man named Tim Berners-Lee, who starts telling you about his proposed system called the World Wide Web. Ian Ritchie was there. And ... he didn't buy it. A short story about information, connectivity and learning from mistakes.

What people say on Youtube:

That was a long buildup to a GREAT PUNCHLINE!!! :D ~RED ♥♫ -- blondwiththewind

The beginning noise is incredibly loud, obnoxious, and unnecessary. They really need to change it. -- Nick Abdallah

His joke at the end wasn't that funny, it was self-deprecating.... jeez you'd think he was Louis CK with the way you people comment. Everything the speaker says for the rest of his life will be dramatic or an inside-joke based on his having a complex about being such a failure of historical-proportions. -- ytubeanon

3. Frontend:

/templates/query.html
/templates/result.html
/templates/talk_page.html
/static/css/query.css
/static/css/result.css
/static/js/bootstrap.min.js
/static/js/circle-progress.min.js
/static/js/jquery-3.2.1.min.js
/static/js/main.js
/static/js/map.js
/static/js/owl.carousel.min.js
/static/img

Three html pages act as entrance of search, showing the structured and filtered data about the talks. List of the result is shown in the result page and users can see the details of each video in specific page corresponding to each talk.

Other css/js files used to build the pages.

4. Readme.md

Instruction of the program.

Design

1. Scrape:

- With the purpose of showing the best search results for users who are interested in TED Talks, we use both data from *TED* and *YouTube* to build our corpus to get a more comprehensive perspective of each TED Talk.
- *TED* has a total number of 3842 TED Talks while *YouTube* has only 2973 TED Talks.
- *YouTube* is much more popular than *TED* worldwidely, *YouTube* has more views and more comments with features of likes and dislikes.
- Through observation, almost all TEDers' comments on TED Talks are positive. The reason why this happens might be that only people who love watch TED Talks go to *TED.com*. In contrast, youtubers do not mind to leave negative comments and criticize the speakers' ideas. Therefore, we only take comments from *YouTube*.
- Originally, we also get the number of likes for the comments. However, most comments with high number of likes are not the review of the content of TED Talks, instead,

general they make jokes and have a good sense of humor. Therefore, we use the relevance between the comments and the TED Talks to rank the comments.

2. Search Engine:

- We use about eighteen fields to help us make an overall search about ted talk from many aspects, like score, speaker, number of views, number of comments, posted date and ratings.
- Firstly we directly show the fields about ted talks that related to the query words. Each video in the result list has its own fields, like title, speaker, views, posted date etc. and parts of the description of the video and the link of source page are also displayed in the result page.
- Also, we use some raw data to do some manipulation and then analyze whether the elements we get is related or typical to the video. For example, when we do some preprocessing. we use views, scores and the number of comments to calculate the relevance and popularity as additional elements of the videos we need to take into consideration.
- We build a recommendation system according to the titles, topics, description and transcript using tf-idf model. Initially, we tried to allocate different weights to them and compare the results, and the best choice is to connect them as a string before calculating the score.
- Extract ten related talks and transfer the title, pic, link that connected to the talk_page to frontend.
- Word cloud by using the feedback from users of TED to count the audiences reaction to the TED Talks.

3. Frontend:

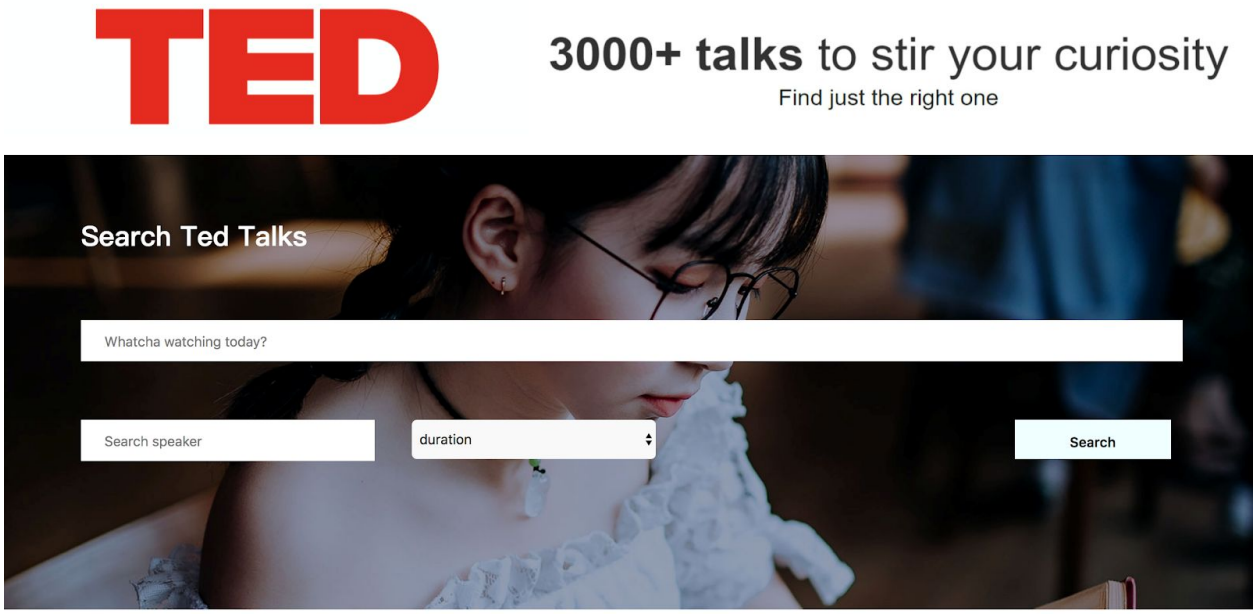
- Users can search ted talks by different fields, free text search is necessary. They can also search the talks by specific speaker. Another constraint is duration, which means how long the talk is.
- In the result page, we need to show the information related to each video. Basically, title, speaker, views and other information like this are shown just besides the talk, and users can directly go to the source page of the ted talk or click the title to see the details of the talk.
- In the talk page, details of the talk are shown, such as the word cloud, transcript and recommended talks. Users can also watch the video in this page. Related talks are list in this page and user can enter the talk_page by clicking the pic of each video.

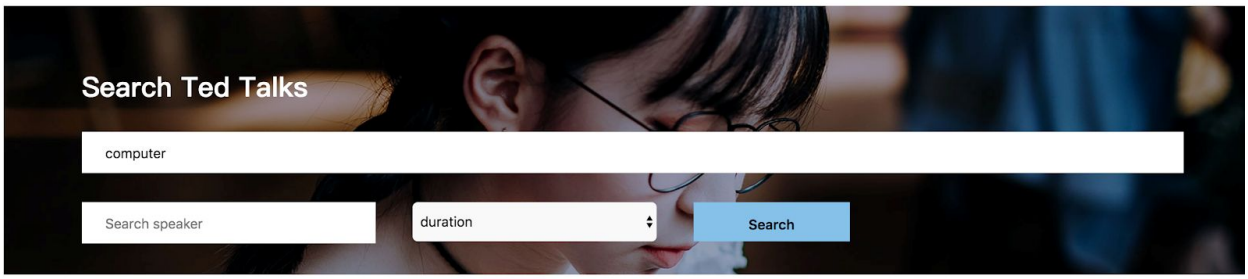
Compiling and Running Program

- Start elasticsearch service before building corpus.
- Create corpus needed as structured data, run ted_index.py to build structured corpus. (~30 seconds)
- Run ted_query.py, and open the search page http://localhost:5000/, shown in screenshot below.

Test

1. Using free text to search.

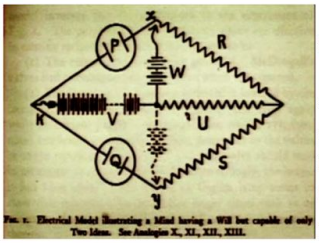




To get more accurate result, you can search with more constraints

Found 653 results. Showing 1 – 10

Next



George Dyson: The birth of the computer

Relevance: 100.00% Popularity : 36.33%

Historian George Dyson tells stories from the birth of the modern computer -- from its 17th-century origins to the hilarious notebooks of some early computer engineers.

(Posted on 2003-03-03 | 18 mins)

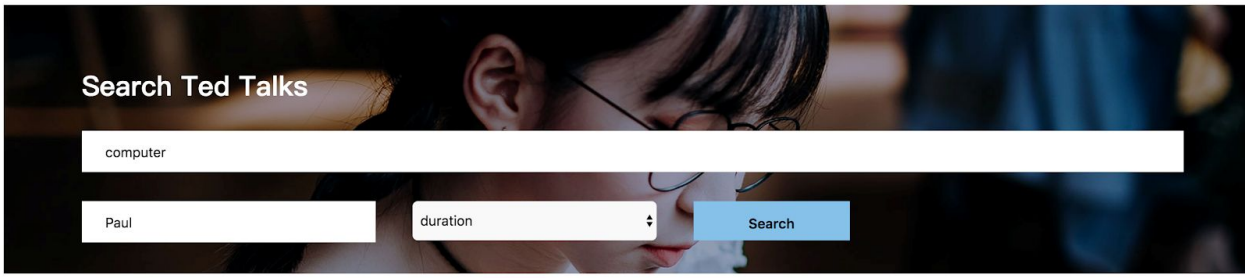
https://www.ted.com/talks/george_dyson_at_the_birth_of_the_computer

Shimon Schocken: The self-organizing computer course

Relevance: 95.71% Popularity : 39.48%

Shimon Schocken and Noam Nisan developed a curriculum for their students to build a computer, piece by piece. When

2. Adding speaker constraint.



To get more accurate result, you can search with more constraints

Found 1 results. Showing 1 – 1

Paul Rothemund: Playing with DNA that self-assembles

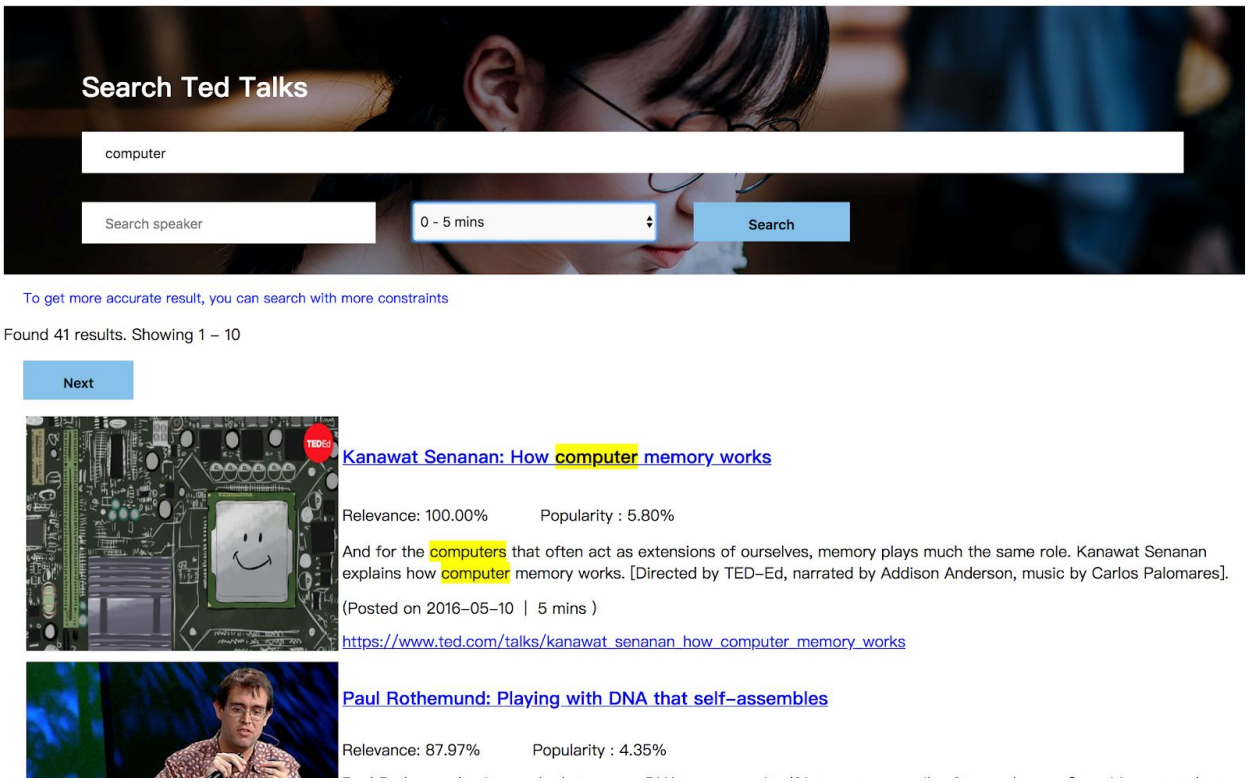
Relevance: 100.00% Popularity : 100.00%

Paul Rothemund writes code that causes DNA to arrange itself into a star, a smiley face and more. Sure, it's a stunt, but it's also a demonstration of self-assembly at the smallest of scales -- with vast implications for the future of making things.

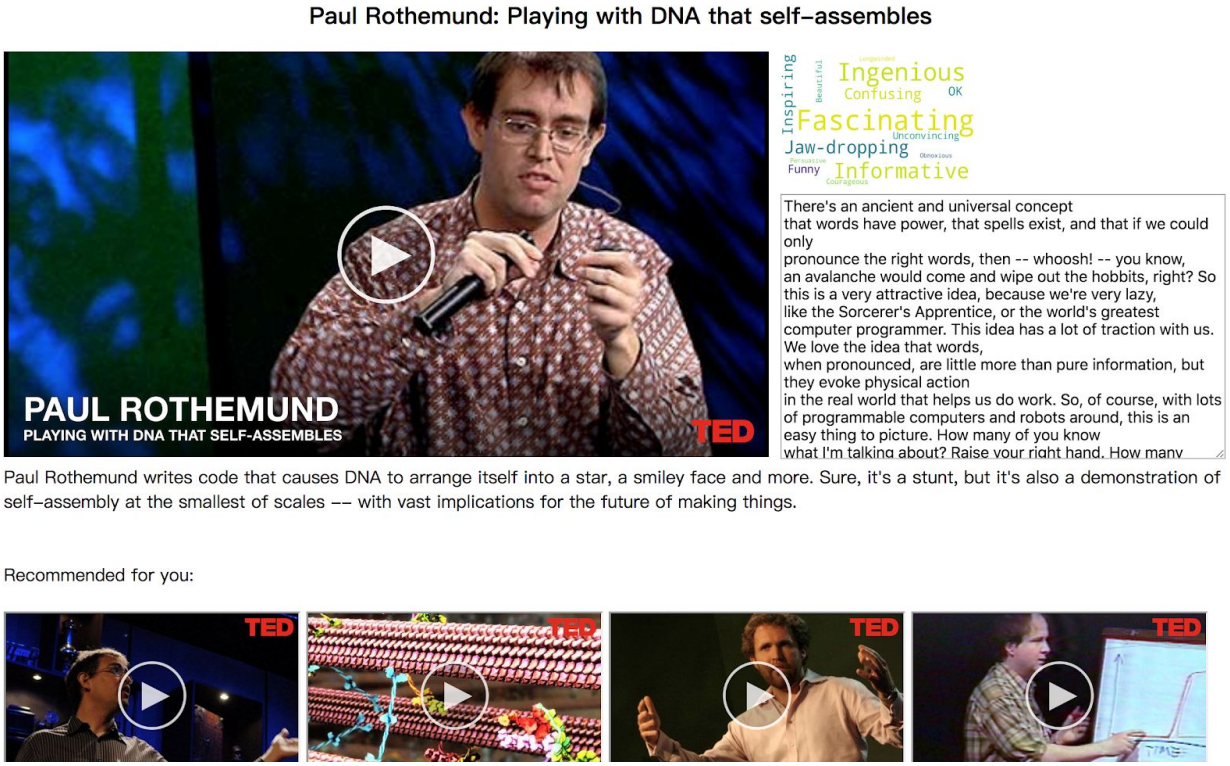
(Posted on 2007-03-03 | 5 mins)

https://www.ted.com/talks/paul_rothemund_casts_a_spell_with_dna

3. Adding duration constraint.



Example of individual result page.



Challenges

During the development, our team faces a series of challenges of building a high performance information retrieval system. The first challenge is evaluation. Our information retrieval system is built offline and it is extremely difficult to find a good way to evaluate the performance of our system. As a team with only three members, it is nearly impossible to perform a high quality test between different system settings. The state of the art information retrieval system like Google uses A/B testing which can get more compelling results. The second challenge is to develop a beautiful UI. None of our team members is expertised in UI design. The third one is that the time is limited and our teams cannot get many feedbacks from peers. Besides, Google Cloud Youtube Data V3 API has a lot of constraints to prevent developers to get the data. For example, one can only query over YouTube through programming languages 100 times per day. Also, the search results have the maximum number of results of 500 which is the way below our required corpus size.

Future Work

We have come up with two ways to further improve our information retrieval system.
The first one is to do a query correction which acts like what Google does. However, we do not have a large database and server to record the users habits.
The second way is to apply natural language process (NLP) knowledge of sentiment analysis to rank the comments from youtubers and get the most representative two with one expresses a positive review and another expresses a negative review. However, it is not easy since this task need to classify not only sentiment but also the quality of the reviews.

Team Member Contributions

Kuan-Yu Chen (Leader) (chenky@brandeis.edu)

I was primarily responsible for ted_index.py and ted_query.py but also helped with frontend development and putting everything together. Beside building the search engine using elasticSearch, I also implemented the recommendation system based on TF-IDF. I generated word clouds and help with data formatting/visualization on result.html and talk_page.html.

Cheng Chen (xiaoxiaochen@brandeis.edu)

What I primarily responsible for is UI development. The code I mainly write is query.html, result.html, talk_page.html and their corresponding css files. The pages used to display the query entrance and the result of query, at the same time, many details of the results based on list of results or some specific talk pages are shown in different ways in each page. And the design of frontend affects the user experience and stable of the system.

Zhiheng Wang (zhihengwang@brandeis.edu)

Code submitted by

Kuan-Yu Chen

GitHub

https://github.com/chenky0401/COSI132A_IR_ted