Search on spoken language content inside YouTube videos

by

Tigran Mamikonyan

Bachelor of Science, Yerevan State University, 2016

A thesis submitted in partial satisfaction of

the requirements for the degree of

Master of Science

in

Computer & Information Science

in the

COLLEGE OF SCIENCE AND ENGINEERING

of the

AMERICAN UNIVERSITY OF ARMENIA

Supervisor: ___________________________

Signature: ___________________ Date:_____________

Committee Member: _____________________

Signature: ___________________ Date:_____________

Committee Member: _____________________

Signature: ___________________ Date:_____________

Committee Member: _____________________

Signature: ___________________ Date:_____________

Abstract

Search on spoken language content inside YouTube videos

Author: Tigran Mamikonyan

Every day, millions of people search across YouTube's billions of videos. The search index is built upon metadata like titles, descriptions and even the content of pages with inbound links. The videos' image content is also classified and labelled [2].

However, the videos' audio content itself is not indexed, and thus not used in matching or ranking. Users cannot perform text queries for audio content across videos, nor within a given video. In 2008, YouTube's parent company Google itself launched a similar project for limited number of political videos [1], but it is not available anymore.

This work is an attempt to start solving this issue. The technology makes YouTube videos searchable, building on the recent advances in speech recognition and continued advances in network bandwidth, storage and processing power.

Keywords: youtube, search, speech recognition, closed captions, subtitles, spoken content

Licenses for Software and Content

Software Copyright License (to be distributed with software developed for masters project)

Content Copyright License (to be included with Technical Report)

Table of contents

Literature Review

"In their own words": political videos meet Google speech-to-text technology [1]

In 2008, a team at Google made an iGoogle gadget which allowed people to search spoken content inside YouTube videos related to elections. It also indicated the parts of the videos where the mentioned words were spoken. The limited number of the searchable videos shows that even Google's resources were not sufficient for making large areas of YouTube searchable. The gadget is currently unavailable and there is not a substitute for it until now.
This work attempts to fill that gap and create a similar tool, making some Youtube videos searchable.

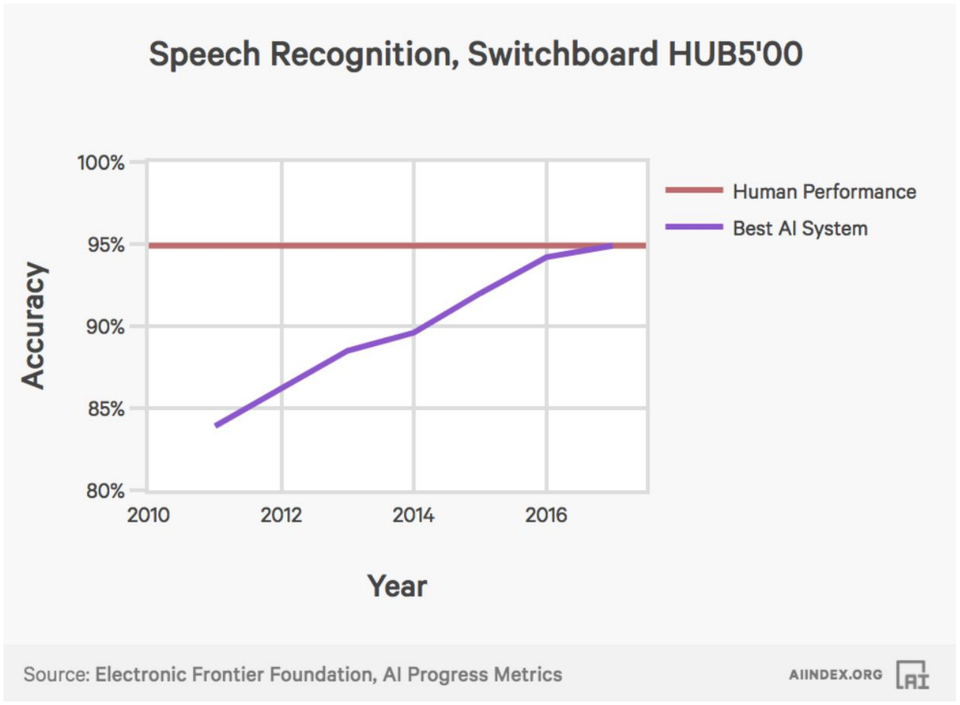
Beyond Short Snippets: Deep Networks for Video Classification [2]

In this article researchers show how effective classification of videos can be based on the images inside videos and the optical flow. They use deep neural networks - specifically, feature pooling networks and recurrent neural networks, for classification. These methods have good accuracy in classification, and are top performers in sports videos.
The search index of Youtube is based on metadata including, titles, descriptions, and the techniques described in the article. However, the audio content of the video is not indexed, which sometimes contains the main information. Based on this work people can classify and search among specific videos.

Microsoft Garage releases "Video Breakdown," generates searchable transcripts and a lot more [3]

Microsoft's Video Breakdown includes several features, one of which is obtaining transcripts from audio. It also provides search through the video according to the transcript and the topic that the speaker is talking about.
Many tools that can make videos searchable are private and used only for specific list of videos defined by the creators. On the other hand, Video Breakdown [3] and Panopto [4] are tools that can help people make specific videos searchable. These tools have a common disadvantage of not providing developers with specific tools or libraries to use in their systems. In addition, they cannot work with YouTube videos directly, and some are paid.

Progress in artificial intelligence for speech recognition



The above graph, mentioned in AI Index 2017 Annual Report, shows the performance of AI systems on a task to recognize speech from phone call audio. [7]

We can see in the graph that in the last few years, speech recognition systems have improved significantly, reaching human performance. This suggests that it might be the right time for this project, as it is heavily based on speech recognition.
The graph also shows that in 2011, the best AI performance was less than 85%. Based on this data, we can imply that this project can succeed and work better than the iGoogle gadget built in 2008.

Goals and objectives

The goal of this work is three main deliverables:

1. ### Making specific videos searchable

   This part of the project is in form of a package. It is mainly written in Python. This is a complete module, and provides easy to use interface. The package will be public, and other developers can use it inside their own applications. Other parts of this project build upon this package.
   The package directly works with YouTube. It uses the audio recording of a YouTube video. Based on speech recognition, it transforms the audio recording into text.
   The package provides functionality to search inside the recognized text. Receiving a query in form of a string, it finds the places where the specific phrase is mentioned. The result is in form of a list of all occurrences of the query, which contains the parts of the text, and the time ranges in the audio where it was mentioned.

2. ### Providing searching tool for YouTube users

   Building on Deliverable 1, we provide a tool that helps to search spoken phrase inside a video, when the user is browsing the YouTube page of that video. It is a Google Chrome extension for YouTube.
   The extension provides a search bar in video pages. Receiving a word or a phrase as a query, it finds the places where the query appears and displays those points on the YouTube page.

3. ### Making portions of YouTube searchable

   This part of the project is built upon the first deliverable. It is part of a Python package. Developers can deploy and use this package inside their applications.

   The deliverable provides search over a set of YouTube videos. The search is similar to the one mentioned above, however the domain is a set of videos instead of a single video.
   The set can be the list of videos of YouTube channels, top 1000 videos of YouTube, or top trending videos. It can also be a set of educational videos (e.g. all Stanford videos) or news videos (e.g. recent EuroNews videos).

Building blocks

This project consists of four major parts.

1. YouTube page

The YouTube page is a part of the project which connects to the system through a Google Chrome extension. Ordinary users will mainly work with the page. A content script of the extension is responsible for working with the page, retrieving information from and making changes in it.
The information in the YouTube page mainly gets processed automatically, without human intervention. The content script finds the necessary information in the page and sends it to the extension. Depending on the type of a YouTube page, different actions may be required. The script will recognize the type of the page, and send the required information. If it is a page of a video then the system should search words or expressions inside the video's audio content. If it is a page of search results the system should search them inside the videos that are on the screen. All of that is done automatically.
When the search results are ready, the content script is responsible for making necessary changes in the YouTube page. Again, the changes depend on the type of the YouTube page. It also provides a way to jump to the part or parts of a video where the search results suggest.

2. Google Chrome extension

In order to provide a searching tool to ordinary YouTube users, we have decided to create a Google Chrome extension. Its main purpose is to connect YouTube page to a server, which will do the main work. It is easier to do the main part in a remote server, rather than in the scope of an extension. Therefore, the extension has a role of intermediary.
The extension is connected both to a YouTube page and to a remote server. It interacts with all the YouTube pages in the browser, and maintains a single connection with the remote server. A single connection for a browser will result in great speed, and later allow to calculate the number of users easier and more accurately.
The extension processes messages from both sides and sends corresponding messages to the other side, thus, maintaining connection between the YouTube page and the server. It is mainly written in javascript, with a small exception of HTML code.
Additionally, the extension provides a searching tool to the users as a text field, where they can type the query. It also shows the status about the current state of the system.

3. Node.js server

The server part is written in Node.js. Its main functions include processing incoming messages, running the python package, and controlling the server.
The server maintains a connection with the extension, and can work simultaneously for thousands of users. The connection is based on Socket.IO library. The data that should be transferred gets compressed, which makes the data transfer faster.
During the whole running time, the server keeps track of the number of active users. It saves the statistics for later analysis.
The server must process incoming messages, and send appropriate responses to the extension. This process usually involves the Python package. Processing the data to send to the package, running the package, receiving and processing the output are all assigned to the server.
The server also plays a role of a web server. It supports the web page of the system with download link and important announcements.

4. Python package

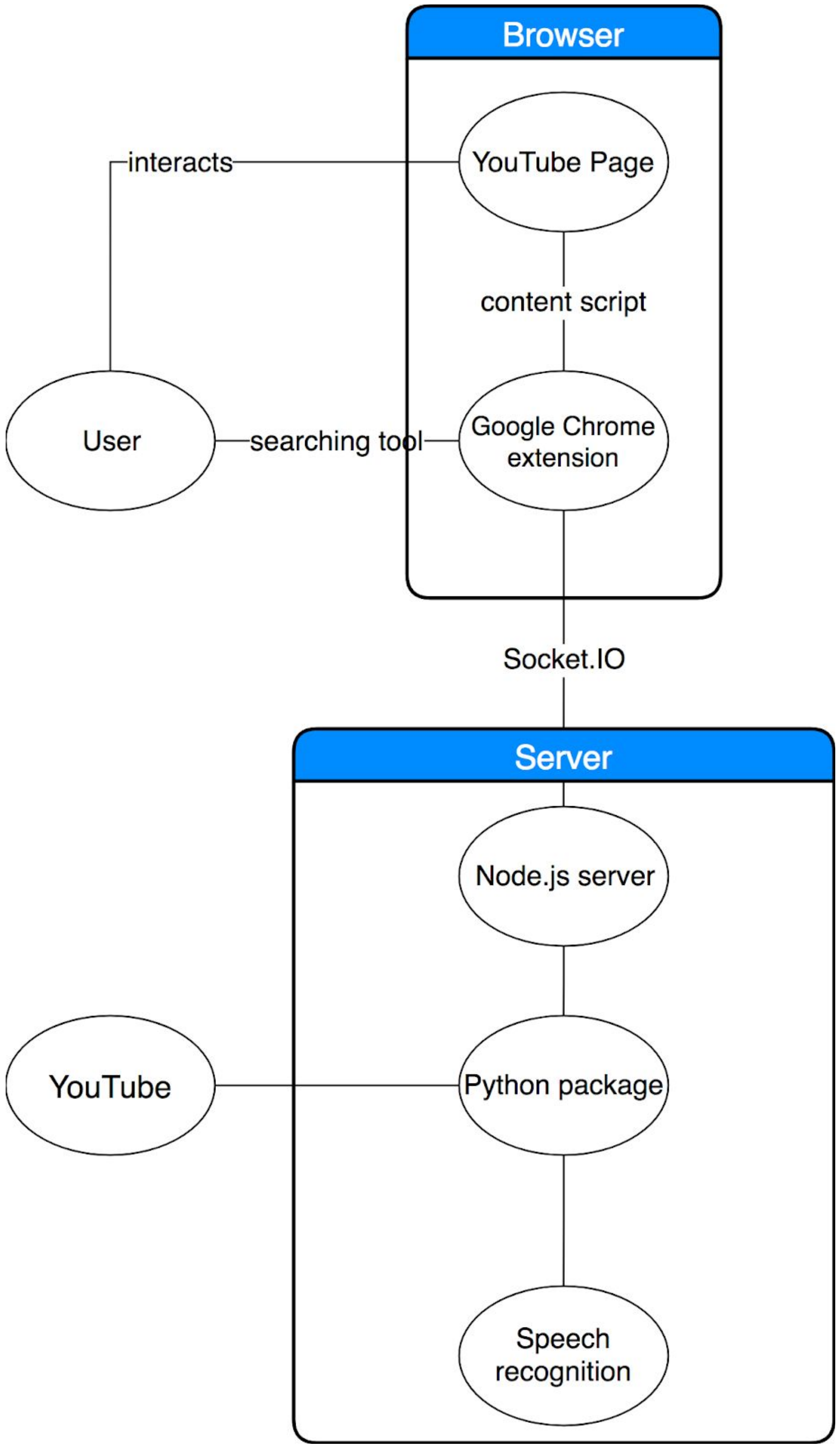
The main work is done inside the Python package. The other parts are for connecting it to the YouTube page and for some additional functions.
The package will receive a YouTube video or a set of YouTube videos to process. It tries to find closed captions from YouTube for the given videos. If there are captions then it will download them in WebVTT format.
In case the captions are missing, the package will try to generate the spoken text itself. First, it downloads the audio content of the video, as downloading the whole video and then subtracting the audio would cost more traffic data. Later, it uses speech recognition on the audio to turn it into text.
We currently use CMU Sphinx speech recognition toolkit [8] for processing the audio and generating the spoken text. It is an open source tool and is easy to use. In some cases, the audio may require preprocessing, such as changing the format.
The resulting recognized text and the downloaded captions are in different forms, hence they are processed separately.
The speech recognition process takes much longer time than processing already existing captions, thus using captions is preferable. In addition, the existing captions usually have better accuracy than the ones generated by the Sphinx toolkit even when they are auto-generated by YouTube.

With our current hardware, the speech recognition takes longer time than the length of the video. However, with the help of GPUs we can increase the recognition speed multiple times.

As the Sphinx toolkit does not deliver high accuracy recognition compared to the best speech recognition systems, we consider replacing it with another system in the future. The replacement process will not take much time as the system is designed to work with any type of a speech recognition tool.

The package is also responsible for processing the captions and searching inside them for the requested query. It finds all the occurrences of the query in the videos and returns the times of these parts.

High level structure of the system



Conclusion and future work

Building on the recent advances in speech recognition and continued advances in network bandwidth, storage and processing power, this project makes it possible to search for audio content inside a YouTube video or across a subset of YouTube videos.

A library for indexing and searching videos' audio content is a building block technology with many interesting potential extensions and applications.

1. Extend Youtube search results

Building on Deliverable 1, this tool will help to filter or rank Youtube search results, videos of channels, or other sets of videos based on audio content. It can also be added to the existing Google Chrome extension.

2. Real-time alerts

This tool will be a service running on a server. It will listen for and notify on the mention of certain words or phrases. Users can indicate a phrase and request notification when the phrase appears in a group of videos (e.g. in videos of a YouTube channel). As soon as a new uploaded video in the group mentions the phrase, the user will receive a notification with the link of the video. The notifications can be sent via email.

3. Monitoring trends

Based on the deliverables, it is possible to create a tool to monitor trends. It will monitor a set of videos and report how the frequency of certain words or phrases changes over time.

4. Training language models

Based on the deliverables, we can gain useful data about YouTube videos. Based on that data, we can train language models.
In particular, they can learn unsupervised [5] or supervised [6] word embedding technique to:

- compare videos

  Decide whether two videos are related to each other, similar, or about the same topic
- cluster videos

Break a list of videos into subsets of videos, which are related to each other. Decide whether the video is restricted and should be blocked

- make predictions

  Predict a label for a video, or for example, the number of upvotes and views that the video will receive

References

1 - googleblog.blogspot.am/2008/07/in-their-own-words-political-videos.html
2 - research.googleblog.com/2015/04/beyond-short-snippets-deep-networks-for.html
3 - onmsft.com/news/microsoft-garage-releases-video-breakdown-generates
-searchable-transcripts-and-a-lot-more
4 - panopto.com
5 - P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, Enriching Word Vectors with Subword Information (arxiv.org/abs/1607.04606)
6 - A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of Tricks for Efficient Text Classification (arxiv.org/abs/1607.01759)
7 - AI Index 2017 Annual Report (aiindex.org/2017-report.pdf)
8 - CMU Sphinx speech recognition toolkit (cmusphinx.github.io)