TubeExtractor: A Crawler and Converter for Generating Research DataSet from YouTube Videos

Pooja Ajwani
Assistant Professor
Faculty of Computer Technology
GLS University, Ahmedabad, Gujarat - 380006

Harshal Arolkar
Professor
Faculty of Computer Technology
GLS University, Ahmedabad, Gujarat - 380006

ABSTRACT
With the advent of the internet and e-resources, there has been an exponential growth of data available to the users. Amongst many content providers, YouTube succeeds in securing the second most popular website in the world. The data from YouTube is easily available to the users, due to which many researchers gather YouTube videos as their dataset for research. Searching the required video for data analysis from YouTube is a cumbersome task as YouTube is overloaded with trillions of videos. Researchers thus need to spend a huge amount of time to get required dataset. To save the time taken by researchers for accumulating dataset, an open source application "TubeExtractor" is proposed in this paper. The TubeExtractor application will allow researchers to download the videos and its metadata from YouTube based on the desired parameters provided by the researcher. The TubeExtractor will also provide as an output a plain text file of the downloaded video. This file can be used by the researchers to do additional processing of their choice if required. The keywords to download the videos are provided to the crawler in the form of a document, generated using a keyphrase extractor algorithm. If the vtt (Video Text Tracks) file of the video to be downloaded is available then a plain text file is created using a two-step parser. This TubeExtractor can save enough time of researchers.

General Terms
Dataset, video, data analysis, extraction, translator.

Keywords
Crawler, Keyphrase extractor, parser, youtube-dl, vtt, RAKE.

1. INTRODUCTION
Since its inception, YouTube has become one of the most popular user content based website which allows the user to perform various activities with video files like upload, view, rate, and share, add to favorites, report, comment and subscribe to channels/videos. As of February 2017, more than 400 hours of video content is uploaded per minute and more than 1 billion hours of content is being watched every day on YouTube [15]. As of August 2018, YouTube website has become the second most popular site in the world [16].

Due to its popularity and ease of availability, numerous scholars are doing research on YouTube's context data and metadata [8] [2] [4]. A Survey done by Kayvan Kousha et. al. [6] indicates that the YouTube videos are used as dataset by numerous researchers in the field of research on Political science, Public Health, Media and Communication, Medicine, Management and Marketing, Intelligence and Security and many more.

Finding data in the huge repository of YouTube is like finding a needle in the pile of a haystack. The difficulty arises when there is a need of crawling videos from YouTube with respect to the desired research topic. For crawling the video when scholar enters the keyword or group of keywords, the endless list of videos appears, some of which are useful some are not. This process of generating dataset takes a lot of efforts and time. In this paper, an open source application **"TubeExtractor"** is proposed to save the time taken by researchers in data collection.

2. RELATED WORK
The web crawler also termed as Spider or Spiderbot or Internet bot is a software program that systematically browses the World Wide Web for the purpose of web indexing [18]. The crawler is a script which when run will automatically download the web pages from the Internet. Researchers have done a lot of research in this area and have developed customized crawlers.

Wang Bingwei et. al [13], classifies the web crawler into four major categories according to system structure and implementation technologies. They have been categorized as General Purpose web crawler, Focused crawlers, Incremental web crawler and Deep web crawler. They developed a Topic web crawler, which does not download all web pages from the Internet, rather crawls the web links of the pages that are related to the topic.

Yuhao Fan [14], designed and developed a distributed web crawler. He developed this using Scrapy framework and used Redis for storage. He also used the Bloom Filter algorithm to avoid repeated crawling if an URL is already present in the Redis.

Egor Lakomin et. al [4], designed and developed a KT-Speech crawler for automatic dataset construction for Speech Recognition from YouTube videos. The crawler proposed can obtain around 150 hours of transcribed speech within a day, considering an estimate of 3.5% word error rate in the transcriptions. They developed the crawler using YouTube video API and then did the cleaning, post-processing and transcription.

3. PROPOSED CRAWLER
In this paper, a crawler "TubeExtractor" for video web crawling of YouTube videos is proposed. The crawler accepts parameter provided by the scholar and generates a repository of plain text files. The TubeExtractor consist of keyphrase extractor, downloader and converter. The keyphrase extractor identifies the keyphrases from the input file provided by the user. These keyphrases are then accepted as input by downloader to generate the repository. The repository consists of video, it's metadata file and vtt file. The web vtt files are the timed text tracks added to the videos [12]. These vtt files are provided as an input to converter resulting into a plain text file.

The web crawler is used to extract the YouTube videos, their context data and metadata. Scholars at times are also interested in extracting the attributes of videos such as comments, tags, average ratings, description, title, likes, dislikes and more. Such information can be extracted using tools like 'wget' or 'Heritrix' from YouTube. One major drawback of these tools is the change in the site and page structure of YouTube [3]. The dataset can be categorized as static metadata – which is static data about the video like its title, description, posted date, the name of owner and more, another category of the dataset is contextual data which is time-dependent information like likes, average ratings, comments, view count and more.

3.1 Working Flow of TubeExtractor

Figure 1 shows the working flow of the TubeExtractor. The working of figure 1 is as mentioned:

1. A user provides the desired parameter in the form of a file/keyword along with a timeline for the purpose of crawling through a user interface.
2. This parameter is then passed to the crawler. If the parameter passed is a content file then the keyphrase extractor will extract the keyphrases from it and store it in keyphrase file. If it is a keyword then it is directly passed to the downloader.
3. If a keyphrase file in step-2 is generated, then it is passed to the downloader.
4. The downloader downloads the videos and metadata related to keyphrases stored in the file, generated in step -2.
5. The converter extracts the vtt (Video Text Tracks) of the video files from the place where downloader stores the videos and metadata.

The converter then translates the vtt file into the plain text and stores at the path specified by the user

4. COMPONENTS OF TUBEEXTRACTOR

The TubeExtractor has been created as an open source application by us, for the researchers. It helps the researchers in data collection from YouTube. It consists of three components namely keyphrase extractor, downloader and a converter. This section describes these components in detail.

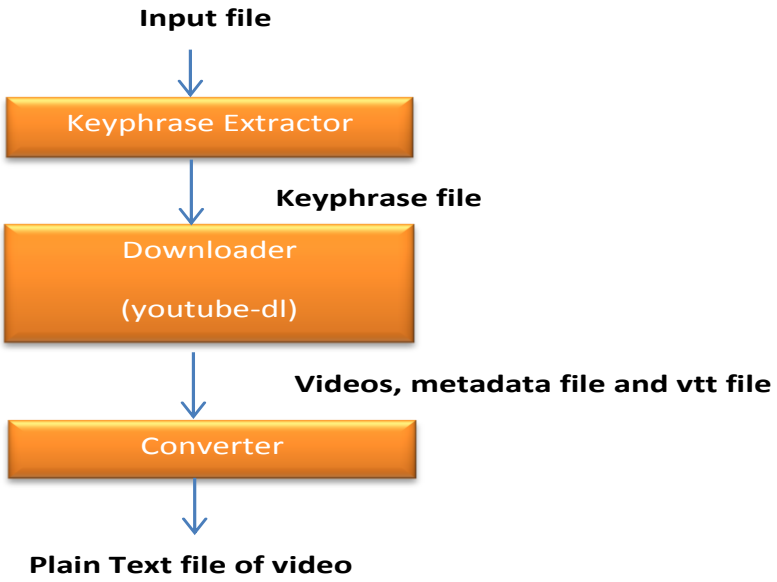

Fig 1: Work Flow of TubeExtractor

4.1 Keyphrase Extractor

Keyphrases are the sequences of words which the researcher intends to find in a document. These keyphrases also give relevance with respect to the topic of a document. This keyphrase extractor takes the document from the user and generates the file which only contains the keyphrases. Many algorithms exist for extracting the keyphrases from a document. Keyphrase extraction can be divided into supervised and unsupervised approach [9].

4.1.1 Supervised Approach

This approach requires a predefined repository of documents. These documents are used as patterns for extracting keyphrases. Many algorithms exist in this category. In this paper, three such algorithms are discussed.

KEA (Keyphrase Extraction Algorithm) is based on features like term frequency, inverse document frequency and the first occurrence of the term. Ian H. Witten et. al. [5] described the KEA algorithm and determines that training document size and global corpus affects the results of the algorithm.

Nirmala Pudota et.al [9] derived a new hybrid algorithm, DIKpE (Domain Independent Keyphrase Extraction), which is an extended form of KEA. It works without any training document as it automatically generates the content based tag.

LetianWang et.al [7] proposed a chunk based method for keyphrase algorithm, which identifies the chunk of keyphrase having length less than 5 words. They showed the results with higher precision as compared to other supervised methods and unsupervised method to rank the candidate keyphrase.

4.1.2 Unsupervised Approach

This approach eliminates the need to train a document. This approach is based on first selecting the candidate phrases from the document and then applies a ranking strategy to find the actual candidate keyphrases. One such algorithm in this category, proposed by Aliaa A.A. Youssif [1] is an automatic Keyphrase extraction algorithm (KPE). This algorithm is based on n-gram filtration technique. The ranking method used in this algorithm is based on position, frequency, and $\chi 2$-measure (chi-square) for testing the bias of co-occurrence distribution between frequent terms. The algorithm showed much higher precision then KEA (supervised) and Ngram filtration technique (Unsupervised).

TextRank algorithm, proposed by Rada Mihalcea et.al [10], is another unsupervised algorithm based on the graph-based ranking model. This algorithm first tokenizes the document as a part of preprocessing. To avoid the massive growth of graph size by adding all possible combinations of more than one lexical unit, they consider every single word as a candidate for addition to the graph. All the lexical units are added to the graph as vertices and edges are drawn between the vertices that co-occur within the window of N words. Once the graph constructed, the score of each vertex initialized with the value 1. Then the ranking algorithm calculates the final score of each vertex and then sorted. Then the top T vertices are considered to be keyphrases and are taken for further processing.

Another important algorithm in this category is RAKE (Rapid Automatic keyword Extraction)[11]. This algorithm first identifies the candidate keyphrases by removing all the stop words and punctuations. Then it identifies the keyword score (based on word frequency, word degree and ration of word frequency to word degree). Based on the keyword score of the candidate keyphrases the final keyphrases are extracted.

RAKE algorithm is an unsupervised, language independent, and domain independent algorithm. It requires no efforts for training dataset and is independent of the size of the document. Owing to this reason, RAKE algorithm has been used in TubeExtractor for extracting the keyphrases.

4.2 Downloader

This module of TubeExtractor is used for the purpose of crawling the YouTube videos based on the search criteria provided by a user from their official website [19] and download them in the local repository along with metadata. The module downloads the requested video only if it is not available in the local repository created before. If the video exists in the repository then the metadata is updated. This module is developed using an open source tool "youtube-dl" [17].

The working of this module is based on the algorithm given here:

1. Read the file generated by the keyphrase extractor.
2. Extract each keyphrase from the file.
3. Read the other parameters provided by the user.
4. Search matching video on YouTube.
5. Download the video as per the parameters and keyphrase identified.
6. Repeat Step-2 to Step-5 for all the keyphrases extracted.
7. Stop

4.3 Converter

This module of TubeExtractor converts the vtt(Video Text Tracks) file of the videos available in the repository into the plain text file. These plain text file can be further used by scholars for processing. The converter is a two-step parser. In the first phase, it does the ETL process that is it extracts the files, cleans all the unnecessary data and loads them into the new temporary file. In the next phase, this temporary file is extracted and all the repeated data is removed which generates the cleaned plain text file. This file is the final required output and can be used for analysis by the researchers. The working of the converter follows the algorithm given here:

4.3.1 Data Cleaning Phase-I

1. Search for the data source in the repository.
2. If the data source is unavailable then wait for the data source to be downloaded by downloader module else search for the .vtt file and proceed to step-3.
3. Open the .vtt file, ignore the header if available.
4. Extract a line.
5. Ignore the line if starting with (00:) and go to Step-4.
6. Identify the substring present within '<' and '>' tags. If the substring is present than ignore and extract the line.
7. Store the line into an intermediate file.
8. Check for the end of the file otherwise go to Step-4.
9. Go to data cleaning Phase II.

4.3.2 Data Cleaning Phase-II

1. Search for the data source in the repository.
2. If the data source is unavailable then wait for the data source to be generated by Phase-I module of the converter.
3. If the data source is available then search for the .txt file.
4. Opens the .txt file.
5. Extract a line.
6. Check if the line is redundant, then discard the line otherwise append the line in final output file.
7. Check for the end of the file otherwise go to Step-5.
8. Stop and generate the plain text file of the video.

5. CONCLUSION

In this paper, a TubeExtractor application is presented that can be used by a researcher for data collection from YouTube. This application helps the scholars in saving their time and efforts required to collect the dataset. The analysis of 1000 videos has shown that dataset collected is of good quality and can be easily used for performing the analysis. Hence the accuracy of the plain text file matched with that of the video is almost 95%.

6. REFERENCES

[1] Aliaa A.A. Youssif, Atef Z.Ghalwash, Islam A.Amer, "KPE: An Automatic Keyphrase Extraction Algorithm" , International Conference on Information Systems and Computational Intelligence (ICISCI 2011), 2011.

[2] Ashish Sureka, Ponnurangam Kumaraguru, Atul Goyal, and Sidharth Chhabra, "Mining YouTube to Discover Extremist Videos, Users and Hidden Communities", Information Retrieval Technology, 6458,13-24.

[3] Chirag Shah, "Supporting Research Data Collection from YouTube with TubeKit" Journal of Information Technology & Politics (JITP), 7(2-3), 226-240 [DOI].

[4] Egor Lakomkin Sven Magg CorneliusWeber Stefan Wermter, "KT-Speech-Crawler: Automatic Dataset Construction for Speech Recognition from YouTube Videos", Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 90–95.

[5] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin and Craig G. Nevill-Manning, "KEA: Practical Automatic Keyphrase Extraction" in Proceeding of DL '99 Proceedings of the fourth ACM conference on Digital libraries, Pages 254-255 , Berkeley, California, USA — August 11 - 14, 1999.

[6] Kayvan Kousha, Mike Thelwall, Mahshid Abdoli, " The role of online videos in research communication: A content analysis of YouTube videos cited in academic publications", Journal of the American Society for Information Science and Technology 63(9):1710-1727 · September 2012.

[7] LetianWang, Fang Li, "SJTULTLAB: Chunk Based Method for Keyphrase Extraction", Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, pages 158–161.

[8] Luca Rossetto and Heiko Schuldt, "Web Video in Numbers An Analysis of Web-Video Metadata", arXiv preprint arXiv:1707.01340 (2017).

[9] Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, Felice Ferrara, Carlo Tasso, "Automatic keyphrase extraction and ontology mining for content-based tag recommendation" , International Journal of Intelligent Systems - New Trends for Ontology-Based Knowledge Discovery, Volume 25 Issue 12, December 2010 , Pages 1158-1186 .

[10] Rada Mihalcea and Paul Tarau, "TextRank: Bringing Order into Texts", Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004.

[11] Stuart Rose, Dave Engel, Nick Cramer and Wendy Cowley, Automatic keyword extraction from individual documents", Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, pages 158–161, Uppsala, Sweden, 15-16 July 2010. 2010 Association for Computational Linguistics.

[12] Thomas Steiner, Hannes Mühleisen, Ruben Verborgh, Pierre-Antoine Champin, Benoît Encelle, Yannick Prié , "Weaving the Web(VTT) of Data", LDO16014 (7th International Workshop about Linked Data on the Web), April 8, 2014, Seoul, Korea.

[13] Wang Bingwei1, Yu Su2, "The Research on Related Technologies of Web Crawler", International Refereed Journal of Engineering and Science (IRJES), ISSN (Online) 2319-183X, (Print) 2319-1821, Volume 6, Issue 4 (April 2017), PP.16-19.

[14] Yuhao Fan, "Design and Implementation of Distributed Crawler System Based on Scrapy", IOP Conf. Series: Earth and Environmental Science 108 (2018) 042086 doi :10.1088/1755-1315/108/4/042086

[15] www.wikipedia.org

[16] www.alexa.com/siteinfo/youtube.com

[17] https://github.com/rg3/youtube-dl/blob/master/README.md

[18] https://en.wikipedia.org/wiki/Web_crawler

[19] https://www.youtube.com/