2021年度 卒業論文 2022/01/13 12:29

YouTubeクリップ自動生成システムの開発

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

18H110 吉田悠真

YouTube クリップ自動生成システムの開発

18H110 吉田悠真

1　はじめに

近年、若年層の youtube 利用率が高くなっている。youtube とは、誰もが無料で自由に動画を投稿、生配信したり、投稿された動画を視聴することができるサービスである。総務省の「令和元年度情報通信メディアの利用時間と情報行動に関する調査報告書概要」[1] から、10 代が利用率が 93.7 %、20 代が 91.5 %、30 代で 85.4 %となっていることからこれは明らかだ。

2　目的

現状、youtube の視聴時間が若い世代を中心に長くなってきている。youtube で配信しているストリーマーの動画は生配信をそのままアーカイブとして残したものが多い。それらを全部見ようとすると膨大な時間が必要になる。YouTube と TV の若年層の一日あたりの平均視聴時間を示したグラフ [2] がある。本研究の目的は、長時間配信アーカイブ中の内容を素早く把握することである。そのために毎日更新される配信アーカイブのチャットリプレイを参考にクリップを自動的に見つけ、生成するプログラムを開発した。

3　YouTube クリップ自動生成システム

YouTube の生配信のアーカイブは、ライブ配信をそのまま動画としてアップロードするため、全容を把握するためには長時間視聴する必要がある。アーカイブは編集された動画と違い、配信者がリアルタイムで配信するため無駄な時間がある。そこで動画内容をすばやく把握するために動画の一部を抜擢しようと考えた。そのなかで YouTube の機能であるライブ中にコメントを入力し、チャット欄に掲載することができる機能を利用して、より多くの人がチャットを入力した時間に多くの情報が詰まっているのではないかと考え、YouTube のアーカイブをより見やすくするために開発したソフトウェアである。

4　検証

主に youtube で活動している”三人称”というチャンネルの動画 [3] を使わせていただき検証する。本システムで一分単位でチャットの総量を計り、どこが一番チャットの振れ幅が大きくなっているか検証する。比較対象として、この動画の切り抜きを行っている”お試し三人称～毎日切り抜き投稿中～”というチャンネルが切り抜いた動画 [4] と比較する。

5　まとめ

本研究では、youtube の長時間にわたる配信アーカイブのチャットリプレイを利用し、配信アーカイブのクリップを自動的に見つけ、再生するシステムの開発を行った。その結果、配信アーカイブからクリップを見つけ、PC のデフォルトのブラウザで youtube を開き再生することができるようになった。しかし動画一本の中に盛り上がった箇所が一か所とは限らないため見逃している場面もあるようだ。今後の課題として、”チャットリプレイだけに頼らず、動画内の音声等を利用した正確な抜粋の実現”、”チャットテキストからどのような内容で盛り上がったのかを識別する機能”の以上 2 点が上げられる。

参考文献

[1] 【2021 年度版】SNS の年代別、利用数・利用率や目的を徹底比較！ https://grove.tokyo/media/g0113/

[2] 大学生の TV 離れ”が顕著、YouTube は 3 時間以上視聴者が 3 割近くに【クロス・マーケティング調べ】 https://webtan.impress.co.jp/n/2020/11/16/38147

[3] 三人称 + 標準の【Battlefield 2042】発売記念配信～芸人&ストリーマー混合対決～ https://www.youtube.com/watch?v=H1ISkt6ZQUE

[4] 思考回路が狂ってしまい BF イベント配信に遅刻して現れる標準さん【三人称 +1/切り抜き/BF2042】 https://www.youtube.com/watch?v=xmjjkb594FI

目次

1　はじめに

近年、若年層の youtube 利用率が高くなっている。youtube とは、誰もが無料で自由に動画を投稿したり、投稿された動画を視聴することができるサービスである。[1] 総務省の「令和元年度情報通信メディアの利用時間と情報行動に関する調査報告書概要」（1）から、10 代が利用率が 93.7 %、20 代が 91.5 %、30 代で 85.4 %となっていることからこれは明らかだ。

第 2 章では本研究の目的について述べる。第 3 章では、youtube について述べる。第 4 章では、スクレイピングについて述べる。第 5 章では本研究で開発したシステムについてを述べる。第 6 章では、本研究の検証について述べる。第 7 章では検証の結果をその考察とともに述べる。第 8 章には研究の成果とともに今後の課題についてまとめる。

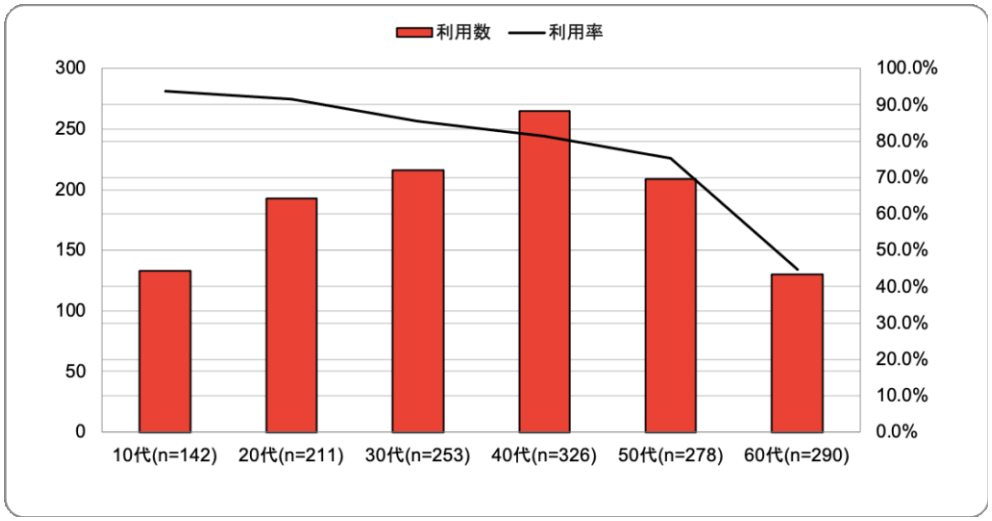

図 1　Youtube の年代別利用数・利用率

2　目的

現状、youtube の視聴時間が若い世代を中心に長くなってきている。youtuube で配信しているストリーマーの動画は生配信をそのままアーカイブとして残したものが多い。それらを全部見ようとすると膨大な時間が必要になる。YouTube と TV の若年層の一日あたりの平均視聴時間を示したグラフ [2] がある。これを見やすくしたものを図 2 と 3 で示す。本研究の目的は、長時間配信アーカイブ中の内容を素早く把握することである。そのために毎日更新される配信アーカイブのチャットリプレイを参考にクリップを自動的に見つけ、生成するプログラムを開発した。

TV 視聴時間（%）

			3 時間以上	30 分～3 時間未満	見ない/30 分未満
全体		1000	16.3	63.0	20.7
男性	13-15 歳	100	11.0	72.0	17.0
	16-18 歳	100	7.0	66.0	27.0
	19-22 歳	100	8.0	55.0	37.0
	23-29 歳	100	12.0	57.0	31.0
	30-34 歳	100	18.0	68.0	14.0
女性	13-15 歳	100	11.0	69.0	20.0
	16-18 歳	100	14.0	68.0	18.0
	19-22 歳	100	20.0	57.0	23.0
	23-29 歳	100	28.0	61.0	11.0
	30-34 歳	100	34.0	57.0	9.0

図 2　若年層の TV の一日当たりの平均視聴時間

YouTube 視聴時間（%）

			3 時間以上	30 分～3 時間未満	見ない/30 分未満
全体		1000	20.3	56.0	23.7
男性	13-15 歳	100	24.0	64.0	12.0
	16-18 歳	100	24.0	64.0	12.0
	19-22 歳	100	33.0	54.0	12.0
	23-29 歳	100	20.0	58.0	22.0
	30-34 歳	100	11.0	58.0	31.0
女性	13-15 歳	100	16.0	65.0	19.0
	16-18 歳	100	25.0	60.0	15.0
	19-22 歳	100	25.0	57.0	23.0
	23-29 歳	100	15.0	45.0	40.0
	30-34 歳	100	10.0	39.0	51.0

図 3　若年層の YouTube の一日当たりの平均視聴時間

3 youtube

[3]YouTube（ユーチューブ）とは、カリフォルニア州サンブルーノに本社を置くアメリカのオンライン動画共有プラットフォームである。2005 年 2 月に PayPal の元従業員であるチャド・ハーリー、スティーブ・チェン、ジョード・カリムの 3 人によって設立されたこのサービスは、2006 年 11 月に 16.5 億米ドルで Google に買収され、現在は同社の子会社の 1 つとして運営されている。YouTube は、アレクサ・インターネットランキングによると、Google 検索に次いで 2 番目にアクセス数の多いウェブサイトである。

YouTube では、ユーザーが動画をアップロード、閲覧、評価、共有、プレイリストへの追加、レポート、コメント、他のユーザーのチャンネル登録などを行うことができる。利用可能なコンテンツには、ビデオクリップ、テレビ番組のクリップ、ミュージック・ビデオ、短編映画やドキュメンタリー映画、音声録音、映画予告編、ライブストリーム、ビデオブログ、短編オリジナルビデオ、教育用ビデオなどがある。ほとんどのコンテンツは個人によって生成され、アップロードされるが、CBS、BBC、Vevo、Hulu などのメディア企業は、YouTube とのパートナーシッププログラムの一環として、YouTube を介してコンテンツの一部を提供している。登録していないユーザーも動画を視聴することはできるが、動画をアップロードすることはできない。年齢制限付きの動画は、18 歳以上であることを確認した登録ユーザーのみが視聴できる。youtube 創設当初、2015 年から 2017 年、それ以降と y outube のロゴが時代ごとで違っている。

2019 年 5 月、YouTube には毎分 500 時間以上のコンテンツがアップロードされており、毎日 10 億時間以上のコンテンツが YouTube で視聴されている。YouTube と選ばれたクリエイターは、サイトのコンテンツや視聴者に応じて広告のターゲットを絞るプログラムである Google AdSense から広告収入を得ている。動画の大部分は無料で視聴できるが、サブスクリプションベースのプレミアムチャンネル、映画のレンタル、YouTube Music、YouTube Premium などの例外もあり、それぞれがプレミアム音楽と広告なしの音楽ストリーミングを提供するサブスクリプションサービスであり、著名人から依頼された独占コンテンツを含むすべてのコンテンツに広告なしでアクセスできる。四半期ごとに報告された広告収入に基づいて、YouTube の年間収入は 150 億米ドルと推定されている。You は「あなた」、Tube は「ブラウン管（テレビ）」という意味である。

3.1 アーカイブ

アーカイブとは、YouTube でライブ配信されたものがそのまま動画としてアップロードされたもの、つまり簡単に言うと見逃し配信のことである。ライブ配信をしたあとに必ず精製されるものではなく、配信主がアーカイブを残す設定をすることで精製される。アーカイブには、チャットリプレイという、ライブ配信していた際に入力されていたチャットをライブ配信の時のように再生する機能がある。しかし視聴者がライブチャットを入力することはできない。

3.2 クリップ

クリップとは、YouTube の機能の一つで 2021 年 1 月にリリースされたもの。このクリップ機能を使うと YouTube にアップロードされている動画のお気に入りの部分だけを切り取って 5 秒から 60 秒の短い動画を作成することができる。

3.3 youtuber

YouTuber とは、オリジナルの動画を YouTube 内で継続的に投稿しているチャンネル運営者を意味します。YouTuber の収入源は YouTube の広告収益や視聴者からの寄付、企業からの案件依頼料など YouTube 内外で様々あります。また YouTube の広告収益を受け取れるのは、過去 12 カ月間の総再生時間が 4,000 時間以上でチャンネル登録者数 1,000 人以上の YouTuber です。そのため YouTuber とは一般的に、収益化の条件をクリアできる程にファンが多く発信力のある動画投稿者を指します。

3.4　ストリーマー

[4] ストリーマーとは、オンラインでゲーム実況を配信するストリーマーと呼ばれる人たちが注目されている。ストリーマーとは、ライブ形式で映像配信する人たちを指す言葉で、その配信ジャンルはゲーム実況、音楽演奏やＤＪプレー、雑談など多岐に渡る。特に人気のジャンルなのがゲーム実況で、世界的なゲーム系ストリーマーになれば、年収数億円以上とも言われている。

3.5 Vtuber

[5] バーチャル YouTuber（バーチャルユーチューバー、英: Virtual YouTuber）は、2016 年 12 月に活動を開始したキズナアイが YouTuber 活動を行う際に自身を称した事に始まる語である。元々はキズナアイ自体を指す語であったが、2017 年末以降では主にインターネットやメディアで活動する 2DCG や 3DCG で描画されたキャラクター（アバター）、もしくはそれらを用いて動画投稿・生放送を行う配信者の総称を指す語として使用されている。略語として、VTuber、V チューバー（ブイチューバー）ともいう。2021 年現在では活動の場が YouTube に限定されない場合もあり、その際は VTuber の名称が使われることが多い。配信形態がストリーマーと似ている。

4　スクレイピング

4.1　WEB スクレイピング

[6] スクレイピングとは、Web サイトから情報を取得するソフトウェア技術のことを示します。簡単にいうと、Web サイト上にある文字や画像、URL などのデータを一度に収集するプログラムです。そのために HTML や CSS、JavaScript を解析したりして、特定の情報を抽出します。使用目的は「情報解析」を前提としており、スクレイピングをするサイトに規制がある場合は注意が必要です。

4.2 クローリング

[7]Web の分野では、ソフトウェアが自動的にインターネットを巡回し、様々な Web サイトから Web ページの内容を収集・保存していく処理をクロールという。このような作業を行うソフトウェアを「クローラ」（crawler）、「ロボット」（robot）あるいは「ボット」（bot）、「スパイダー」（spider）などと呼ぶ。クローラはある Web ページをダウンロードして内容を解析し、別のページへのハイパーリンクを発見すると、そのページを取り寄せて内容を解析する。この手順を繰り返し、インターネット上で公開されているサイトやページを次々に発見して文書や画像などのファイルを収集していく。

4.3 スクレイピングとクローリング違い

[6] クローリングもスクレイピングと同様、Web サイトから情報を取得する技術ですが、クローリングは巡回した Web サイトからすべての情報を取得します。一方スクレイピングはクローリングで取得した情報の一部分を抽出することができる技術です。このため、必要な情報のみに特化したい場合にはスクレイピングが有効でしょう。

5　YouTube クリップ自動生成システム

5.1　YouTube クリップ自動生成システム

YouTube の生配信のアーカイブは、ライブ配信をそのまま動画としてアップロードするため、全容を把握するためには長時間視聴する必要がある。アーカイブは編集された動画と違い、配信者がリアルタイムで配信するため無駄な時間がある。そこで動画内容をすばやく把握するために動画の一部を抜擢しようと考えた。そのなかで YouTube の機能であるライブ中のコメントを入力し、チャット欄に掲載することができる機能を利用して、より多くの人がチャットを入力した時間に多くの情報が詰まっているのではないかと考え、YouTube のアーカイブをより見やすく使用と開発したソフトウェアである。

5.2　特徴

従来、YouTube のクリップと言うものは、自分で動画を視聴した後に自分が気に入った 5 秒から 60 秒までの箇所を切り取り保存する機能であるが、この YouTube クリップ自動生成システムを使うことにより使用者は動画を全部見ることなく、ほかの視聴者が最もコメントを残したいと思った場所が視聴することができる。なおかつ一カ所だけでなく複数箇所みたい場合もコメントの量が多い順で切り抜く場所を探すことができる。もとの YouTube のクリップ機能は 60 秒までだがこのシステムは、自分で秒数をしていすることができるので全部は見たくないが多く見たいという時などにも対応できる。

5.3　詳細

本システムを起動後、クリップを作成したい動画の URL を本システムに入力する。すると次にクリップを作成する数を指定できるので好きな数字を入力する。その後、入力された URL を元にその動画サイトの HTML を入手し、その中からライブチャットの記録、つまりチャットリプレイを取り出し保存する。この後、チャットリプレイの一分間ごとのコメントの量を元にクリップを作成していく。このときに、後に出てくる図 10 と図 11 で使用されている comment_diff.txt と comment_data.txt が自動で保存されるようになっている。作成し終えるとパソコンに設定されているデフォルトのブラウザで最初に入力した URL の動画ページが開き作成されたクリップの時間から自動的に再生される。

5.4　動作説明

使うパソコンの OS は windows とする。初めに本システムのプログラムを起動すると URL を聞かれるので、自分の見たい配信アーカイブの URL を貼り付ける。本システムのプログラムを起動した直後の画面を図 4 に示す。

図 4　本システムをコマンドプロンプトで実行した図

本システムを起動した際、配信アーカイブではない、または配信アーカイブであるがチャットリプレイがない動画の URL を入力するとエラーになる。エラー画面を図 5 に示す。

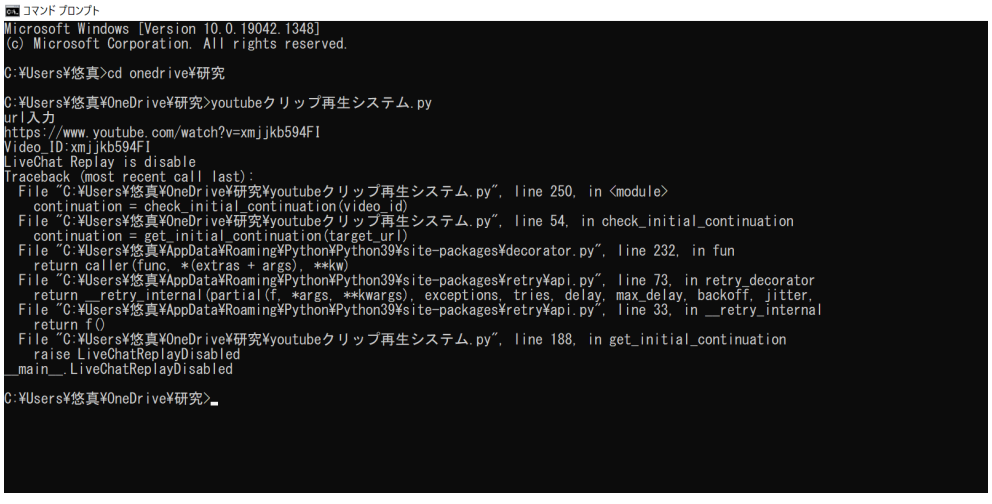

図 5　本システムをコマンドプロンプトで実行後、エラー画面図

URL を入力して Enter を押すと配信アーカイブのチャットを記録する。その後パソコンに設定されているデフォルトのブラウザで youtube を開き、最もコメントの差が大きい部分を再生する。プログラム開始および、必要事項入力後を 6、プログラム終了時を 7、プログラムが終了し画面が遷移し終わった状態を 8 で示す。

図 8 ウェブページの上部に URL がある。この URL は最初に入力した URL であり、末尾に追加で、&t=4680s と入力されている。これは動画の再生開始時間を表しており、同じく図 8 の動画内の時間が 1:18:00 となっており秒換算すると 4680 秒になる。

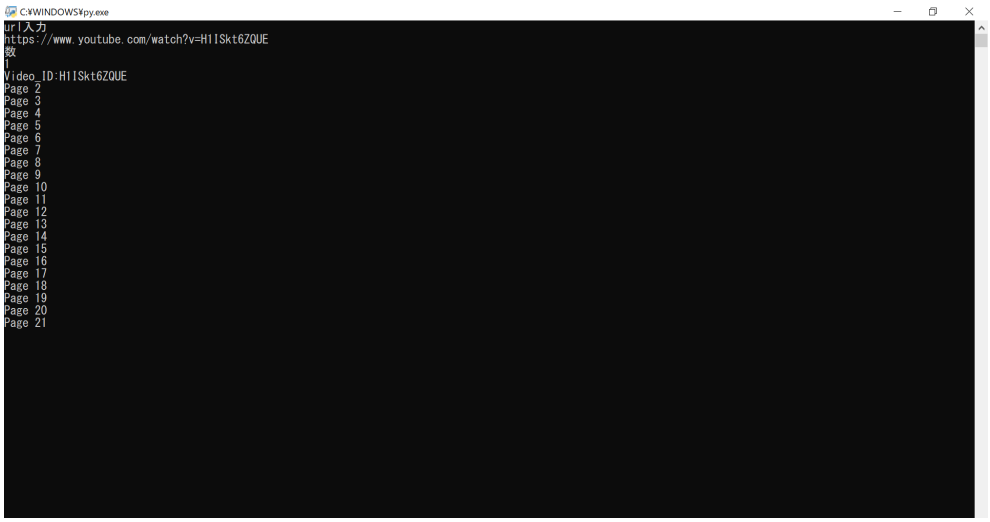

図 6　プログラム起動後、URL、クリップ数を入力した後の図

図 7　プログラム終了時

図８　プログラム終了後、デフォルトのブラウザでクリップを再生している図

6　検証

主に youtube で活動している”三人称”というチャンネルの [8]”三人称 + 標準の【Battlefield 2042】発売記念配信〜芸人&ストリーマー混合対決〜”という動画を使わせていただき検証する。この動画は 1 時間 53 分 50 秒となっており、これを本システムで一分単位でチャットの総量を計り、どこが一番チャットの振れ幅が大きくなっているか検証する。比較対象として、この動画の切り抜きを行っている”お試し三人称〜毎日切り抜き投稿中〜”というチャンネルが切り抜いた動画 [9]”思考回路が狂ってしまい BF イベント配信に遅刻して現れる標準さん【三人称 +1/切り抜き/BF2042】”の切り抜き部分と比較する。切り抜きは本動画の 3 分から 6 分くらいを切り抜いており、動画は全部で 3 分 39 秒となっている。

7　結果と考察

7.1　結果

検証の結果、本システムで抜き出された箇所と切り抜き動画の再生個所は違っていた。検証結果を図 9 で示す。本システムは、1 時間 18 分時点からの部分を抜き抜いていた。切り抜き動画は 3 分時点からの部分を切り抜いていた。

図 9　左：プログラムが終了した画面、右：デフォルトブラウザで youtube が開かれた画面

7.2　チャット合計グラフ

図 10 からは、時間ごとのコメントの数がわかる。本システムで切り抜かれた箇所のコメントは 90 強あり、全体の順位は 3 位。対して切り抜き動画が抜き取った箇所は 80 弱とコメントの数の順位でいえば、4 位だ。

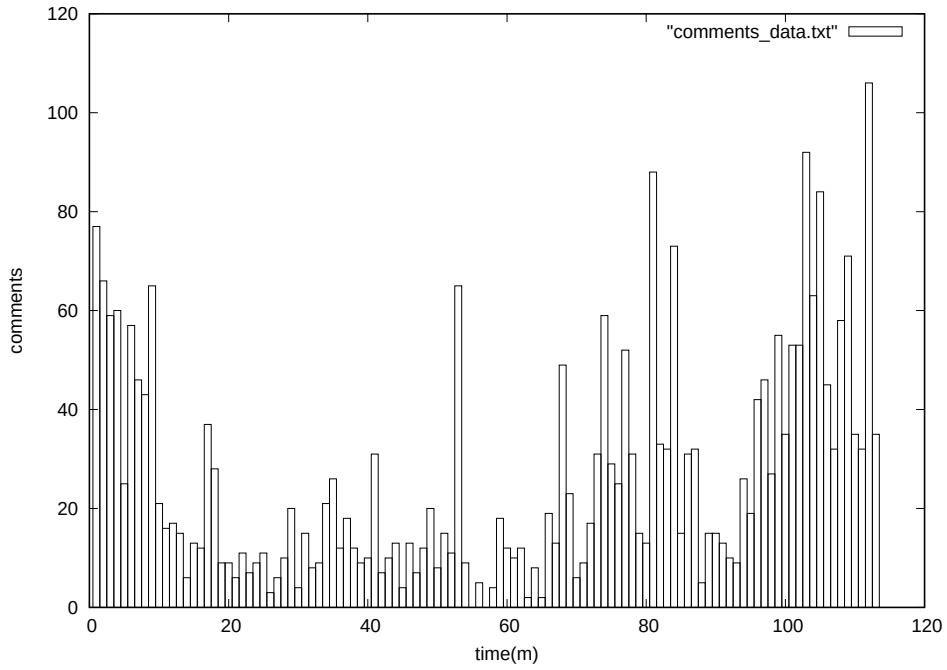

図 10　コメント数 縦軸:一分あたりのコメントの総量　横軸:時間（分）

7.3　チャットリプレイの前後の差のグラフ

11 からは、一分あたりのコメントの差の絶対値がわかる。結果本システムで切り抜かれた箇所が一番差が激しい。切り抜き動画で切り抜いている箇所は前後の差がない。

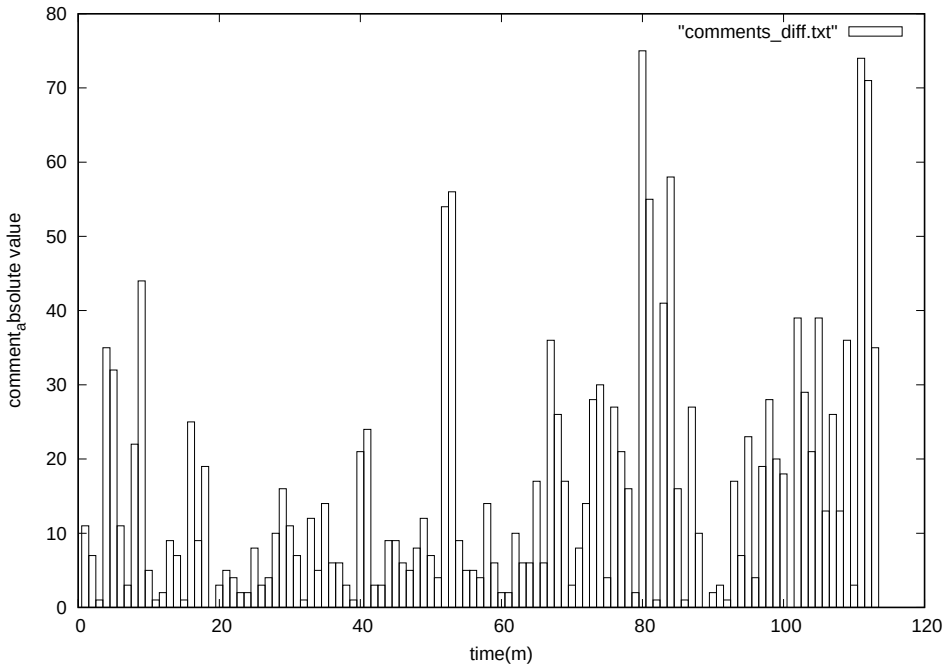

図 11　コメントの差の絶対値、縦軸:一分あたりのコメント差の絶対値 横軸:時間（分）

7.4　考察

図 10、11 からわかるのはコメントの差で動画の面白さが決まるわけではなく、あくまで判断材料の一つになると言うことだ。さらに切り抜き動画は、本動画の 3 分時点を切り抜いていているので配信の挨拶とかぶったのではないかと考える。

8 結論

本研究では、youtube の長時間にわたる配信アーカイブのチャットリプレイを利用し、配信アーカイブのクリップを自動的に見つけ、再生するシステムの開発を行った。その結果、配信アーカイブからクリップを見つけ、PC のデフォルトのブラウザで youtube を開き再生することができるようになった。そのため配信アーカイブの一番盛り上がった箇所を即座に視聴できるようになった。しかし動画一本の中に盛り上がった箇所が一か所とは限らないため見逃している場面もあるようだ。今後の課題として、”チャットリプレイだけに頼らず、動画内の音声等を利用した正確な抜粋の実現”、”チャットテキストからどのような内容で盛り上がったのかを識別する機能”の以上 2 点が上げられる。

謝辞

本論文執筆及び研究室等、研究室での活動の際以外でもご指導・ご協力いただきました大垣斉准教授、情報教育システム研究室の学生、当研究室の OB に深く感謝いたします。

参考文献

[1] 【2021 年度版】SNS の年代別、利用数・利用率や目的を徹底比較！ https://grove.tokyo/media/g0113/

[2] 大学生の TV 離れ”が顕著、YouTube は 3 時間以上視聴者が 3 割近くに【クロス・マーケティング調べ】 https://webtan.impress.co.jp/n/2020/11/16/38147

[3] https://ja.wikipedia.org/wiki/YouTube

[4] https://www.anime.ac.jp/contents/column/2021/11/29/streamer/

[5] https://ja.wikipedia.org/wiki/バーチャル YouTuber

[6] https://services.sms-datatech.co.jp/pig-data/2020/12/22/whatisscraping/

[7] https://e-words.jp/w/クロール.html

[8] 三人称 + 標準の【Battlefield 2042】発売記念配信〜芸人&ストリーマー混合対決〜 https://www.youtube.com/watch?v=H1ISkt6ZQUE

[9] 思考回路が狂ってしまい BF イベント配信に遅刻して現れる標準さん【三人称 +1/切り抜き/BF2042】 https://www.youtube.com/watch?v=xmjjkb594FI

付録 A　ソースコード

Listing 1　foo

```
from bs4 import BeautifulSoup
import ast
import requests
from retry import retry
import json
import sys
import csv
import webbrowser
import time


class ContinuationURLNotFound(Exception):
    pass

class LiveChatReplayDisabled(Exception):
    pass

class RestrictedFromYoutube(Exception):
    pass


def get_ytInitialData(target_url, session):
    headers = {'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/
    html = session.get(target_url, headers=headers)
    soup = BeautifulSoup(html.text, 'html.parser')
    for script in soup.find_all('script'):
        script_text = str(script)
        if 'ytInitialData' in script_text:
            for line in script_text.splitlines():
                if 'ytInitialData' in line:
                    if 'var ytInitialData =' in line:
                        st = line.strip().find('var ytInitialData =') + 19
                        return(json.loads(line.strip()[st:-10]))
                    if 'window["ytInitialData"] =' in line:
                        start_pos = line.strip().find('window["ytInitialData"]') + len('wi
                        end_pos = len(';</script>') * -1
```

```
                    return(json.loads(line.strip()[start_pos:end_pos]))

    if 'Sorry for the interruption. We have been receiving a large volume of requests from
        print("restricted from Youtube (Rate limit)")
        raise RestrictedFromYoutube

    print("Cannot get ytInitialData")
    return(None)

def get_initial_continuation(target_url):
    session = requests.session()
    try:
        ytInitialData = get_ytInitialData(target_url, session)
    except RestrictedFromYoutube:
        return(None)

def check_initial_continuation(video_id):
    target_url = "https://www.youtube.com/watch?v=" + video_id
    continuation = get_initial_continuation(target_url)
    return(continuation)

def get_chat_replay_from_continuation(video_id, continuation, pagecount_limit, is_locally_r
    count = 1
    pagecount = 1
    continuation_prefix = "https://www.youtube.com/live_chat_replay?continuation="
    session = requests.Session()
    result = []
    data = []
    total = []


    while(pagecount < pagecount_limit):
        if not continuation:
            print("continuation is None. maybe hit the last chat segment")
            break

        try:
            ytInitialData = get_ytInitialData(continuation_prefix + continuation, session)
            if not ytInitialData:
                print("video_id: " + video_id + ", continuation: " + continuation)
```

```
            continuation = None
            break

        if not 'actions' in ytInitialData['continuationContents']['liveChatContinuatio
            continuation = None
            break
        for action in ytInitialData['continuationContents']['liveChatContinuation']['a
            if not 'addChatItemAction' in action['replayChatItemAction']['actions'][0]
                continue
            chatlog = {}
            item = action['replayChatItemAction']['actions'][0]['addChatItemAction']['
            if 'liveChatTextMessageRenderer' in item:
                chatlog = convert_chatreplay(item['liveChatTextMessageRenderer'])
            elif 'liveChatPaidMessageRenderer' in item:
                chatlog = convert_chatreplay(item['liveChatPaidMessageRenderer'])
            if 'liveChatTextMessageRenderer' in item or 'liveChatPaidMessageRenderer'
                result.append(chatlog)
                data.append(int(chatlog/second))
                count += 1


        continuation = get_continuation(ytInitialData)
        pagecount += 1
        if is_locally_run:
            print('\rPage %d\n' % pagecount, end='')

    except requests.ConnectionError:
        print("Connection Error")
        continue
    except requests.HTTPError:
        print("HTTPError")
        break
    except requests.Timeout:
        print("Timeout")
        continue
    except requests.exceptions.RequestException as e:
        print(e)
        break
    except KeyError as e:
        print("KeyError")
```

```
                print(e)
                print(item['liveChatTextMessageRenderer'])
                break
            except SyntaxError as e:
                print("SyntaxError")
                print(e)
                break
            except KeyboardInterrupt:
                break
            except RestrictedFromYoutube:
                print("Restricted from Youtube, Rate limit")
                break
            except Exception as e:
                print("Unexpected error:" + str(sys.exc_info()[0]))
                print(e)
                break

        print(video_id + " found " + ("%03d" % pagecount) + " pages")

        return(data, continuation,count)

    def convert_chatreplay(renderer):
        chatlog = []

        chatlog = renderer['timestampText']['simpleText']

    # すべての時間を秒にする。
        if len(chatlog) == 4:
            chatlog = int(chatlog[2:]) + int(chatlog[:1]) * 60
        elif len(chatlog) == 5:
            chatlog = int(chatlog[3:]) + int(chatlog[:2]) * 60
        elif len(chatlog) == 7:
            chatlog = int(chatlog[5:]) + int(chatlog[2:4]) * 60 + int(chatlog[:1]) * 3600
        elif len(chatlog) == 8:
            chatlog = int(chatlog[5:]) + int(chatlog[2:4]) * 60 + int(chatlog[:2]) * 3600



        return(chatlog)

    def get_continuation(ytInitialData):
```

```
    continuation = ytInitialData['continuationContents']['liveChatContinuation']['continua
    return(continuation)

def check_livechat_replay_disable(ytInitialData):
    conversationBar = ytInitialData['contents'].get('twoColumnWatchNextResults',{}).get('c
    if conversationBar:
        conversationBarRenderer = conversationBar.get('conversationBarRenderer', {})
        if conversationBarRenderer:
            text = conversationBarRenderer.get('availabilityMessage',{}).get('messageRende
            print(text)
            if text == 'この動画ではチャットのリプレイを利用できません。':
                return(True)
    else:
        return(True)

    return(False)

@retry(ContinuationURLNotFound, tries=2, delay=1)
def get_initial_continuation(target_url):
    session = requests.session()
    try:
        ytInitialData = get_ytInitialData(target_url, session)
    except RestrictedFromYoutube:
        return(None)



    if not ytInitialData:
        print("Cannot get ytInitialData")
        raise ContinuationURLNotFound

    if check_livechat_replay_disable(ytInitialData):
        print("LiveChat Replay is disable")
        raise LiveChatReplayDisabled

    continue_dict = {}
    try:
        continuations = ytInitialData['contents']['twoColumnWatchNextResults']['conversatio
        for continuation in continuations:
            continue_dict[continuation['title']] = continuation['continuation']['reloadCon
    except KeyError:
```

```
        print("Cannot find continuation")

    continue_url = None
    if not continue_url:
        if continue_dict.get('上位のチャットのリプレイ'):
            continue_url = continue_dict.get('上位のチャットのリプレイ')
        if continue_dict.get('Top chat replay'):
            continue_url = continue_dict.get('Top chat replay')

    if not continue_url:
        if continue_dict.get('チャットのリプレイ'):
            continue_url = continue_dict.get('チャットのリプレイ')
        if continue_dict.get('Live chat replay'):
            continue_url = continue_dict.get('Live chat replay')

    if not continue_url:
        continue_url = ytInitialData["contents"]["twoColumnWatchNextResults"].get("convers

    if not continue_url:
        raise ContinuationURLNotFound

    return(continue_url)



if __name__ == '__main__':
# を変えることで何秒ごとでデータをとるのかを決めれる。second
    global second
    second = 60

    print("入力url")
    target_url = input()
    print("数")
    kazu = input()

    if '&t=' in target_url:
        target_url = target_url[:target_url.find('&t=')]
    if '?t=' in target_url:
        target_url = target_url[:target_url.find('?t=')]
```

```
    video_id = target_url.replace("https://www.youtube.com/watch?v=","")
    print("Video_ID:"+ video_id)

    dict_str = ''
    next_url = ''
    session = requests.Session()
    cnt = 0
    x = 1
    comment_data = []
    comment_diff = []



    html = session.get(target_url)
    soup = BeautifulSoup(html.text, 'html.parser')

    title = soup.find_all('title')

    continuation = check_initial_continuation(video_id)
    data, continuation, count = get_chat_replay_from_continuation(video_id, continuation,

# コメントをで決めた秒数ごとでカウントする。get_chat_replay_from_continuation
    for i in data:
        if i == cnt:
            x += 1
        else:
            cnt = i
            comment_data.append(x)
            x = 0
# コメントごとの差を調べる。
    g = 1
    for z in comment_data:
        try:
            if z - comment_data[g] <= 0:
                comment_diff.append((z - comment_data[g])*-1)
            else:
                comment_diff.append(z - comment_data[g])
            g += 1
        except:
            comment_diff.append(z)
```

```
# ファイル出力
    x = 0
    with open('comments_diff.txt','w+') as f:
        for j in comment_diff:
            x += 1
            f.write("%d %d\n" % (x, j))
    x = 0
    with open('comments_data.txt','w+') as f:
        for j in comment_data:
            x += 1
            f.write("%d %d\n" % (x, j))

    print("output comments_diff, comments_data")

#ブラウザを開く

    for j in range(int(kazu)):
        url = target_url + "&t=" + str(second * comment_diff.index(sorted(comment_diff)[-1
        webbrowser.open(url)

    time.sleep(30)
```