

Marquette University

e-Publications@Marquette

---

Dissertations (1934 -)

Dissertations, Theses, and Professional  
Projects

---

## Temporal Sentiment Mapping System for Time-Synchronized Data

Jiachen Ma  
*Marquette University*

Follow this and additional works at: [https://epublications.marquette.edu/dissertations\\_mu](https://epublications.marquette.edu/dissertations_mu)



Part of the [Mathematics Commons](#)

---

### Recommended Citation

Ma, Jiachen, "Temporal Sentiment Mapping System for Time-Synchronized Data" (2022). *Dissertations (1934 -)*. 1597.

[https://epublications.marquette.edu/dissertations\\_mu/1597](https://epublications.marquette.edu/dissertations_mu/1597)

TEMPORAL SENTIMENT MAPPING SYSTEM FOR TIME-SYNCHRONIZED  
DATA

by

Jiachen Ma, M.S.

A Dissertation submitted to the Faculty of the Graduate School,  
Marquette University,  
in Partial Fulfillment of the Requirements for  
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

August 2022

ABSTRACT  
TEMPORAL SENTIMENT MAPPING SYSTEM FOR TIME-SYNCHRONIZED  
DATA

Jiachen Ma, M.S.

Marquette University, 2022

Temporal sentiment labels are used in various multimedia studies. They are useful for numerous classification and detection tasks such as video tagging, segmentation, and labeling. However, generating a large-scale sentiment dataset through manual labeling is usually expensive and challenging. Some recent studies explored the possibility of using online Time-Sync Comments (TSCs) as the primary source of their sentiment maps. Although the approach has positive results, existing TSCs datasets are limited in scale and content categories. Guidelines for generating such data within a constrained budget are yet to be developed and discussed.

This dissertation tries to address the above issues by leveraging existing live comments from a popular video distributed platform, YouTube, as a primary time-synchronized data source and exploring efficient strategies for generating TSCs with a constrained budget. An automatic data mining system was first developed and deployed across multiple platforms. Then, long-period experiments were conducted to test the efficiency of the framework. Additionally, two large-scale TSCs datasets were created through the proposed data framework and analyzed for their characteristics. Finally, the outcomes were tested against the original temporal Automatic Speech Recognition (ASR) sentiment labeling to validate their accuracy. The experiment shows the potential of automatically generating temporal sentiment datasets through the proposed mapping system. This project also provides valuable tools for future multimedia research.

## ACKNOWLEDGMENTS

Jiachen Ma, M.S.

Firstly, I would like to express my sincerest gratitude to my Ph.D. advisor, Dr. Sheikh Iqbal Ahamed, for his outstanding mentorship in guiding me through this long research journey. Without his support, it would be impossible for me to achieve this accomplishment.

Secondly, I would like to acknowledge Dr. Naeen Bansal and Dr. Mehdi Maadooliat for serving on my committee and spending timeless effort helping me complete my study, polish my writing and prepare my presentations. Without their tremendous help, it would be impossible for me to go this far.

Then, I would like to acknowledge my friend and mentor, Dr. Piyush Saxena, for his generous help during my study. He helped me open my mind to new ideas and possibilities across many disciplines. Without his assistance, it would be tough for me to proceed.

Next, I would like to acknowledge the Department of Mathematical and Statistical Sciences at Marquette University. Brilliant minds and friendly attitudes inspired me to face challenges in research and life. It is a great pleasure for me to work with my labmates, colleagues, and friends here.

Moreover, I would also like to extend my sincere appreciation to Direct Supply. They provided funding for my dissertation research at Marquette University. Their team also assisted me with the experiment settings and offered valuable insights and feedback.

Finally, I am deeply indebted to my parents, Zhenjun Ma and Chen Han, who have loved and supported me unconditionally throughout my life. I am lucky enough to be their son and have their support on my back in pursuing my dreams. Your caring love shines through the darkness and leads me to the ultimate success.

## TABLE OF CONTENTS

ACKNOWLEDGMENTS .....	i
LIST OF TABLES .....	v
LIST OF FIGURES .....	vii
I. INTRODUCTION .....	1
I.A Thesis Statement.....	6
I.B Difficulties and Challenges.....	7
I.C Contribution to Knowledge .....	7
I.D Study Outcomes.....	9
II. LITERATURE REVIEW .....	10
II.A Time-Sync Comments.....	10
II.B Time-Sync Comments Datasets.....	12
II.C Sentiment Analysis.....	14
II.D Textual-based Sentiment Mapping.....	15
II.E Sentiment Labeling via Live Comments .....	16
II.F Event Detection through Sentiment.....	18
II.G Modality Features in Videos .....	20
II.H Data Acquisition.....	22
III. DATA COLLECTION FRAMEWORK .....	25
III.A Data Collection Challenge .....	25
III.A.1 Data Source Selection.....	25
III.A.2 Live Stream Identification .....	26
III.A.3 Official APIs Quota Limit .....	27
III.B Proposed Framework .....	28

III.B.1 Live Video Detection .....	28
III.B.2 Data Collection.....	29
III.B.3 Summary of the Framework.....	30
III.C Framework Development.....	32
III.C.1 Selections Interface .....	33
III.C.2 Options Interface.....	34
III.C.3 Outputs Interface.....	34
III.C.4 Mobile Application .....	35
III.D Data Collection Experiment.....	35
III.D.1 LST-YF20 Dataset .....	36
III.D.2 LST-YT1000 Dataset .....	38
III.E Framework Evaluation .....	39
III.F Data Profiling .....	42
III.F.1 LST-YT1000 Dataset .....	42
III.F.2 LST-YF20 Dataset .....	44
III.F.3 Comparison with Existing Datasets.....	49
IV. SENTIMENT MODELING.....	51
IV.A Problem Statement .....	51
IV.B Auxiliary Data .....	52
IV.C Speech Recognition.....	53
IV.D Offline ASR Experiment.....	55
IV.E Online ASR Data Collection .....	57
IV.F Data Processing.....	59
IV.F.1 Data Subsetting .....	60

IV.F.2 Online ASR Data Pre-processing.....	66
IV.F.3 ASR and TSCs Data Alignment.....	68
IV.F.4 Data Cleaning.....	71
IV.F.5 Sentiment Conversion .....	73
IV.F.6 Data Imputation.....	76
IV.F.7 Semantic Similarity .....	78
IV.G Model Structure.....	80
IV.H Model Evaluation and Comparison.....	81
V. SUMMARY .....	90
V.A Discussion.....	90
V.A.1 Data Collection.....	90
V.A.2 Data Acquisition.....	92
V.A.3 Speech Recognition .....	92
V.A.4 Sentiment Annotation.....	93
V.A.5 Sentiment Modeling .....	94
V.A.6 Semantic Distance .....	95
V.B Conclusion .....	95
V.C Future Work .....	96
VI. STUDY TIMETABLE.....	97
BIBLIOGRAPHY .....	98

## LIST OF TABLES

1	A list of large-scale TSCs datasets . . . . .	12
2	A list of commonly used unsupervised CPD algorithms . . . . .	19
3	A simple example of the output attributes from Pytchat. . . . .	23
4	Streaming ingestion protocols and prefix. . . . .	23
5	Streaming ingestion protocols for third-party clients. . . . .	24
6	Available platforms and protocols supported by Streamlink CLI. . . . .	24
7	This table compares different quota units of the same method between various categories (“YouTube Data API Overview”, 2020). . . . .	27
8	Linear models fit with 20 different fruit keywords. Int* represents Confidence Interval of $\hat{\beta}_1$ . . . . .	41
9	My dataset compared with other large TSCs datasets in the field. *m is a abbreviation for million. *k is a abbreviation for thousand. *Bili represents Bilibili, a popular video platform from China. *AF represents AcFun, a popular video platform from China. *YT represents YouTube. A popular video platform developed by Google. *IntNet represents for Internet. *h means hours. . . . .	50
10	Compatible models officially provided for the Vosk toolkit. Models are tested on TED-LIUM dataset (Korvas et al., 2014). WER (Word Error Rate). . . . .	55
11	Vosk models’ (Korvas et al., 2014) WER (Word Error Rate) on four datasets: Librispeech, Tedlium, Google Commands and Fisher eval2000. . . . .	56
12	An example output of a list of words from Vosk API. . . . .	56
13	Auto-generated subtitle by Tim Smart’s script through Tampermonkey. . . . .	59
14	This table shows the auto-generated captions’ summary statistics for both LST-YF20 and LST-YT1000. . . . .	61
15	This table shows the number of videos left after each filtering operation for all LST-YT1000 live streams. . . . .	66
16	A summary shows the statistic of words in the comments of four different YouTube streams. Thirteen thousand two hundred and one comments are analyzed, and most comments are less than ten words. . . . .	74



17 A list of popular sentiment lexicons exist in Sentiment Analysis literature.	75
18 This table compares MSE loss for different parameter combinations of the shallow LSTM model. WS stands for the window size, and Input stands for the inputs sequence length. Accuracy Improvement is the reduced percentage of MSE loss after applying the similarity feature to the model.	82
19 The timetable of this study . . . . .	97

## LIST OF FIGURES

I-1 Monthly users of main social media platforms, 2002-2018. . . . .	2
I-2 A live streaming screenshot was captured from the YouTube website. Live comments are posted in the right section during streaming. . . . .	4
I-3 A live streaming screenshot was captured from the Twitch website. Live comments are posted in the right section during streaming. . . . .	4
I-4 A danmaku video screenshot was captured from the Bilibili website. Live comments (in Chinese) flow on top of the video during streaming. . . . .	4
II-5 The above figure shows a simple mock-up of the live commenting interface. Audiences on the right side constantly comment on the video streams playing on the left-hand side. . . . .	11
II-6 Plutchik's emotion wheel. . . . .	15
II-7 The sentiment conversion process . . . . .	17
II-8 The workflow of creating a sentiment annotation time series . . . . .	17
II-9 An illustration of the semi-supervised video event classification procedure. A supervised learning classifier uses the "pseudo labels" generated by the previous unsupervised learning step as targets to learn from multimodal features. . . . .	18
II-10 A deep autoencoder framework creates a multimodal representation by learning features from individual modality input (Ngiam et al., 2011). . . . .	20
II-11 The supervised step uses hidden layer representation as input. It tries to classify these representations into different pseudo labels. . . . .	20
II-12 Visual, aural, and textual modalities exist in video streams. They can be used to detect video events. . . . .	21
II-13 General workflow of using sentiment labels in deep multimodal learning. . . . .	22
III-14 This graph shows a high-level overview of the proposed data pipeline's workflow. The video prober on the left uses a set of keywords to search for related videos per day. The parser on the right side occasionally checks for all searched video IDs and collects their meta-information and TSCs. . . . .	28
III-15 A sequence diagram of the proposed data pipeline . . . . .	31

III-16	A screenshot of the graphic layout of the proposed data pipeline . . . . .	33
III-17	This screenshot shows the user interface of the Selections section. A custom keyword entry "games" is input into the search field. . . . .	34
III-18	This screenshot shows the user interface of the Options section. The example shows a running sequence of searching a single keyword, merging previous results, and downloading all TSCs. . . . .	34
III-19	This screenshot shows the user interface of the Outputs section. The example shows the internal progress of searching all related videos for a given searching term "games" on YouTube. . . . .	35
III-20	Running the data pipeline on a mobile device . . . . .	36
III-21	The plot shows the growing number of observations detected from the video prober on 20 fruits keywords. I collected this data during a 23 days experiment, starting on July 3rd, 2021. . . . .	37
III-22	This screenshot was captured during the TSCs downloading step. Commenting actions other than TSCs are also collected. . . . .	39
III-23	This histogram shows the distribution of 20 categories of videos we collected on the first day of the experiment. The left cluster contains three keywords: "raspberry", "nectarine," and "cantaloupe." The right cluster contains rest of the keywords. . . . .	39
III-24	This plot shows the growing number of observations of 20 fruit keywords during 171 days. The experiment started on July 10th, 2021. . . . .	40
III-25	This histogram shows the distribution of TSCs counts in the logarithm of the LST-YT1000. Total 492,331,808 comments are included. . . . .	43
III-26	This histogram shows the distribution of authors count in logarithm of LST-YT1000. Authors from 330,427 videos are included. . . . .	43
III-27	This histogram shows the TSCs intensity distribution in the logarithm of the LST-YT1000 dataset. . . . .	44
III-28	This distribution of each video's approximate duration of LST-YF20. Total length of videos is 18,088 hours. . . . .	45
III-29	This histogram shows the probability distribution of videos' duration in the logarithm of the LST-YF20 versus the LST-YT1000 dataset. . . . .	46
III-30	The histogram shows the distribution of each video's TSCs count in the logarithm of LST-YF20. Total of 6,755,675 comments are included in the plot. . . . .	46

III-31	This histogram shows the distribution of each video's authors count in the logarithm of LST-YF20. 10,448 videos with valid authors are included in the plot. . . . .	47
III-32	This plot shows the number of 15 different categories of streams in the LST-YF20. . . . .	48
III-33	This plot shows the TSCs intensity of different video categories of the LST-YF20. . . . .	48
III-34	This plot shows the number of unique authors within 15 different categories of the LST-YF20. . . . .	49
IV-35	The graph shows the workflow of creating a fine-grained sentiment labeling from auxiliary information. . . . .	52
IV-36	The graph shows the overall process of converting semantic features from audio inputs. The audio signal was first transformed into a text-based transcript by the ASR engine and then examined by NLP algorithms to obtain a precise semantic representation. . . . .	54
IV-37	The graph shows there are three internal models for an ASR system. The AM model recognizes phoneme units. The PM model identifies the correct boundaries of a word. The LM model converts the sequence of phonemes into a semantic representation. . . . .	54
IV-38	A YouTube subtitle download options show up on the web page via Tim Smart's script. . . . .	58
IV-39	Auto-generated English subtitle are available for download through Tim Smart's script. . . . .	58
IV-40	The histogram of the LST-YT1000 transcript's file size. . . . .	60
IV-41	The histogram of LST-YF20 transcript's file size. . . . .	60
IV-42	The histogram of the LST-YF20 transcript's video duration. . . . .	62
IV-43	The histogram of the LST-YT1000 transcript's video duration. . . . .	63
IV-44	The histogram of the LST-YF20 transcript's words count. . . . .	63
IV-45	The histogram of the LST-YT1000 transcript's words count. . . . .	64
IV-46	The histogram of transcript's words per second of the LST-YF20. . . . .	64
IV-47	The histogram of transcript's words per second of the LST-YT1000. . . . .	65

IV-48	Transcripts generated by Automatic Speech Recognition (ASR) models may overlap with each other (pink area) or contain gaps (blue area).	67
IV-49	The overlapping issue was solved by trimming the duration of the first overlapped transcript.	68
IV-50	Transcripts generated by Automatic Speech Recognition (ASR) models are rounded to the closest integers on the time axis.	68
IV-51	Transcripts generated by Automatic Speech Recognition (ASR) models are synced with TSCs data on the same time axis.	70
IV-52	The tokenizer split a complete sentences into individual words.	72
IV-53	The lemmatizer covert the token 'talking' into 'talk'.	72
IV-54	The process of removing stopwords from token set.	73
IV-55	Number of words' distribution for a random 901 comments.	73
IV-56	The plot of average polarized sentiment on a 60 seconds window size.	75
IV-57	The plot of average polarized sentiment on a 180 seconds window size.	76
IV-58	Sentiment sequence comparison under window sizes of 100, 300, and 600 seconds.	77
IV-59	The 1st and 2nd PCA component embedding space the Word2Vec model trained on the 167 videos. Words with similar meanings are placed closely together.	78
IV-60	The cluster distance between TSCs and ASR tokens for video 'mx-CfMsBqgJE.' Data was generated with a 300 seconds window size.	80
IV-61	Structure of Recurrent Neural Network (RNN)	81
IV-62	LSTM model with full connection layers.	81
IV-63	MSE loss of the LSTM model trained on 60 seconds inputs generated from a 100 seconds smoothing window.	84
IV-64	MSE loss of the LSTM model trained on 120 seconds inputs generated from a 100 seconds smoothing window.	85
IV-65	MSE loss of the LSTM model trained on 180 seconds inputs generated from a 100 seconds smoothing window.	85

IV-66 MSE loss of the LSTM model trained on 60 seconds inputs generated from a 300 seconds smoothing window. . . . .	86
IV-67 MSE loss of the LSTM model trained on 120 seconds inputs generated from a 300 seconds smoothing window. . . . .	86
IV-68 MSE loss of the LSTM model trained on 180 seconds inputs generated from a 300 seconds smoothing window. . . . .	87
IV-69 MSE loss of the LSTM model trained on 60 seconds inputs generated from a 600 seconds smoothing window. . . . .	87
IV-70 MSE loss of the LSTM model trained on 120 seconds inputs generated from a 600 seconds smoothing window. . . . .	88
IV-71 MSE loss of the LSTM model trained on 180 seconds inputs generated from a 600 seconds smoothing window. . . . .	88

## **I Introduction**

The sentiment is a combination of emotions and thoughts. It derives from the internal experience of our feelings and emotions. Roberts, 1988 argued emotions are an intentional state which depends on the subject believing some state of affairs of obtaining. Complex emotions or feelings can be polarized into positive and negative sentiments, which contribute to our affection or alienation towards a topic or entity. Understanding such sentiment could be a powerful way of explaining what actions we will take.

Therefore, sentiment analysis or opinion mining has become one of the most powerful tools in today's research. Studies from multiple disciplines have adopted sentiment analysis from the past few decades, and more and more researchers have realized the importance of sentiment. Particular research field such as psychology (Zucco et al., 2017), linguistics (Brooke et al., 2009), public relations (Proksch et al., 2019), communication or marketing (Rambocas and Pacheco, 2018) rely heavily on sentiment analysis to get better understanding of human reactions on certain entity. Beyond that, Sentiment analysis also benefits the business world. Marketing decisions such as a promotion or advertising need to consider consumer's emotional responses to maximize the effect of branding and increase sales. Saura et al., 2019 concludes several commercial success factors from sentiment analysis.

Sentiment data can be collected, extracted, and analyzed from multiple sources. Formats such as text, image, audio, or video can be found online carrying rich sentiment features. Such information structures are called "multimodal" data in today's literature. Gathering such information is easy as most of today's communication happened on the internet. Several tools have been developed and distributed freely for this purpose. Such programs are also called "Web crawlers" or "spider bots," which systematically gather data of interest through the web with no human interaction. With little effort, researchers can access a vast collection of multimodal sentiment data in a short time.

Traditionally, sentiment analyses are performed based on text. In the early 2000s, with the rise of social media platforms such as Twitter, My Space, or Facebook, informative textual material became available to the public for the first time. Researchers quickly find out those online textual contains user sentiment. It could help companies agile adjust their product and marketing strategies based not only on the emotional feedback of users. Since then, textual analysis via Natural Language Processing (NLP) has become popular in the sentiment analysis field. Social media platforms continue to grow in the past decades and have become one of the most successful ways for people to communicate with each other. A recent data survey (Ortiz-Ospina, 2019) suggests that approximately one-third of people in the world are using social media platforms, which contribute to almost two-thirds of all internet users. Figure I-1 shows leading social media platforms such as Facebook and YouTube had more than 2 billion users by the end of 2018 and continue multiplying.

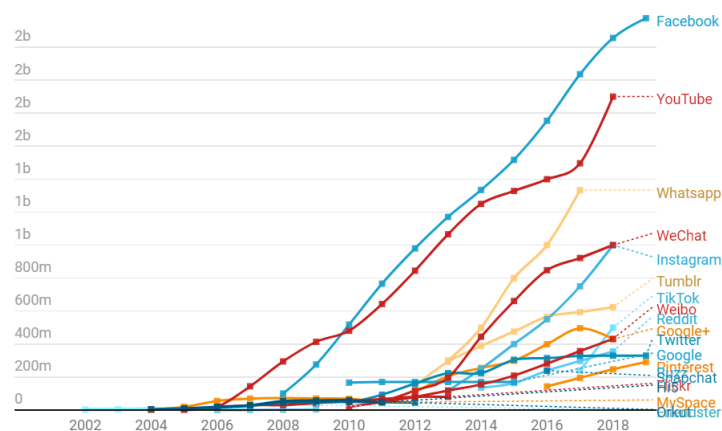


Figure I-1: Monthly users of main social media platforms, 2002-2018.  
Source by: <https://ourworldindata.org/rise-of-social-media#licence>

Although most sentiment data sources are still in textual format, many popular social platforms, such as Youtube, Twitch, or Tiktok, could generate more data in the form of videos daily. Studying the underlying sentiment with this new data type could be exciting but challenging. The main difference between a video and a textual paragraph is how information is stored within data. Textual material tends to have precise



semantic meanings since communication relies on the accuracy of the language.

On the other hand, video-type material is more complicated as audiences are fed with two types of data sources, auditory and visual, simultaneously. Psychology theory suggests that the information acquisition process could be more efficient from both verbal and imagery channels than a single one (Sadoski and Paivio, 1994, Paivio, 2013). A natural extension of textual analysis would be converting video and audio channels into sequences of descriptive words first and then performing the NLP to extract sentiment. But the audio channel does not only contains vocal tracks. Environmental noise or musical tone could also exist.

Furthermore, video channels usually include consecutive images that hardly be translated directly into precise descriptive language with a state-of-the-art (SOTA) computer vision model. A straightforward approach is highly challenging and requires substantial computation resources to perform. Some exciting question occurs. Can we work around this and learn some sentiment indirectly with little computational expense for video-type material? Can we map similar media to sentiment sequences just by textual information?

Luckily, a new type of social media provides a potential solution to the problems. With live streams becoming popular, live comments are also available on some social media platforms such as YouTube (Figure I-2) or Twitch (Figure I-3). Audiences can exchange their thought and comments on screen as the live stream goes. In East Asia, a similar social media platform provides live remarks in a slightly different form. Instead of posting live comments in a chat room, audiences post their messages on top of the video. This type of live comment is called "Danmaku" in Japanese, which refers to a pattern of dozens to hundreds of bullet comments flowing simultaneously over time. Figure I-4 shows an example of "Danmaku" floating on top of a video.

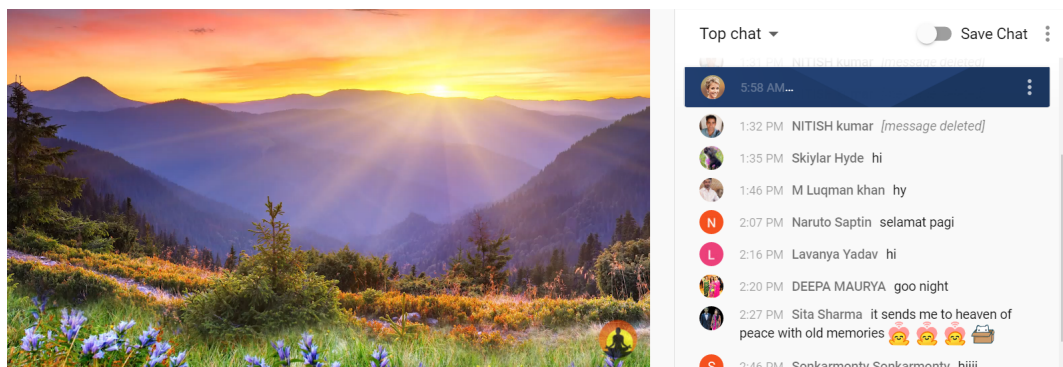


Figure I-2: A live streaming screenshot was captured from the YouTube website. Live comments are posted in the right section during streaming.

Source by: <https://www.youtube.com/watch?v=IioH3S5jDSY>



Figure I-3: A live streaming screenshot was captured from the Twitch website. Live comments are posted in the right section during streaming.

Source by: <https://www.twitch.tv/agraelus>

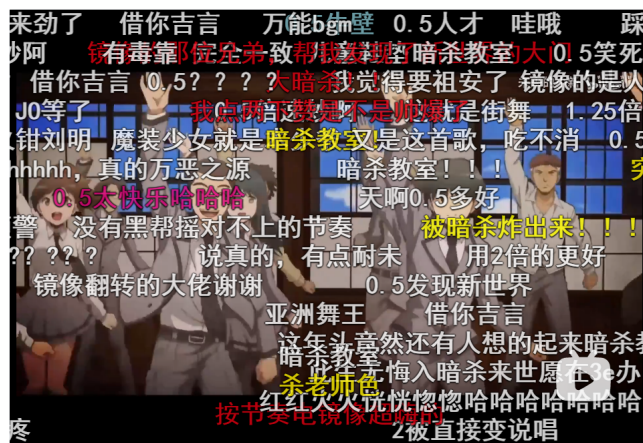


Figure I-4: A danmaku video screenshot was captured from the Bilibili website. Live comments (in Chinese) flow on top of the video during streaming.

Source by: <https://www.bilibili.com/video/BV1MJ411L7vX>

Several studies suggest that textual material attached to a video could be considered human scoring for the content from a sentiment perspective. I tried to crawl 12000 live comments from a single YouTube stream as a preliminary study of my research and verify this type of data source is very sensitive to the video's content. The positive result encouraged me to build a generative framework for online temporal sentiment data. I hypothesized that I could use the live comments attached to live streams to create large-scale time-sync comments (TSCs) datasets efficiently. My later experiments confirmed the idea, and I was able to generate two TSCs datasets through my data pipeline. The study can then be described in three folds:

1. Since live comments are sensitive to the opinion holder's sentiment, I can leverage them to label affective reactions from human viewers in time series.
2. I can build a data mining framework to automatically track online streams and extract their live comments on a large scale for sentiment conversion.
3. The relationship between sentiment annotations and actual sentiment can be studied and leveraged to build a generative sentiment model that predicts the audience's sentiment reaction.

In summary, this work tries to leverage the implicit textural information as sentiment labels of live comments to explore the best strategy for generating temporal sentiment annotation datasets for multimedia research. By expanding the data collection experiment, I was able to create large-scale TSCs datasets on a low budget and test the performance of my data collection framework. The work also provides a set of efficient toolkits for online data crawling, cleaning, and analysis. In addition, a mobile data mining solution was proposed and implemented on top of my framework.

Researchers can now use customer degree devices to create more sentiment datasets. I also verified the accuracy of the sentiment mapping between live comments and the videos through a series of supervised learning experiments. Despite some minor limitations and drawbacks, it can still significantly impact and create value in the

sentiment analysis discipline. In other words, my work systematically studies and explores the automated approach to generating sentiment annotations for temporal data.

### **I.A Thesis Statement**

This study aims to build a temporal sentiment mapping system for online time-synchronized comments (TSCs) from existing live streams. Online stream's live comments can carry important information on the internal states of viewers. Such sentiment feelings could be evaluated and labeled by performing a text-based sentiment analysis on the TSCs gathered from viewers. By analyzing the original video, temporal features such as sentiment values, comments density, and the number of authors can be merged to form unique TSCs datasets for future studies.

The study also aims to provide useful data mining strategies for automatically creating large-scale TSCs datasets with minimal human intervention. A data mining tool will be first prototyped to test the mining strategies on a small sample. Then, the data pipeline will generate large-scale TSCs datasets to evaluate the framework's efficiency. The project will also create a graphic user interface for daily users to better interact with the data pipeline. Finally, the mining program will be held on consumer-degree mobile devices to explore the possibility of generating TSCs datasets with a constrained budget.

Lastly, the study examines the quality of the TSCs datasets by comparing the sentiment difference between original videos to the TSCs labeling. Speech recognition will be first applied to extract semantic information from live streams. Then sentiment labels will be generated using a pre-trained sentiment analyzer. After merging the actual sentiment with the TSCs', the combined sentiment labels will be analyzed through sequence-to-sequence neural networks to evaluate the performance of different temporal features. The study will provide important insight into the sentiment accuracy of the TSCs datasets.

## **I.B Difficulties and Challenges**

Sentiment mapping for videos is usually challenging and expensive due to the limitation of the traditional hand-labeling strategy. Leveraging online temporal synchronized data to generate automatic sentiment tags can significantly improve the labeling accuracy and data scale. Recent studies showed that converting online comments into video taggings can benefit research.

However, most existing TSCs datasets were collected from limited platforms such as the Bilibili and the AcFun, which are mainly used by East Asian audiences. Data bias may be introduced to the datasets since only including people from specific regions could not represent the entire world population. Gathering comments directly from an English native-speaking site is more suitable for creating a TSCs dataset.

Creating a large-scale dataset through a data mining framework can be challenging and expensive. The data collection and cleaning process can be time-consuming. Many data mining frameworks are designed to run on distributed networks, which could introduce additional costs in hardware maintenance. Developing a low-budget-friendly data mining tool is critical for the research community. Some frameworks also lack an interactive interface for casual users. Learning curves or barriers may exist for researchers.

This study explored and built an autonomous data mining pipeline using one of the most popular online streaming resources, YouTube, to address the above problems. The data collection tool has a graphic user interface to lower the barrier of usage and can run on low-budget devices. The project also generated two large-scale datasets for sentiment analysis and video tagging. Some experiments prove the data accuracy and mining efficiency of the framework. I hope my project can help researchers by lowering the cost and reducing the difficulties of preparing such datasets in this field.

## **I.C Contribution to Knowledge**

First, the study proposes an efficient pipeline of annotating affective reactions from live comments and creates a comprehensive dataset for multimodal studies of video

streams. Similar annotations from previous datasets either use video comments from discussion sections or surveys. The proposed method is more accurate than others as it provides fine-grained human annotations in time series. It is also capable of offering multiple resolutions of annotations with different hyper-parameters.

Second, this study tries to use a semi-supervised algorithm to detect video events that trigger salient changes in viewers' sentiment by using their comments. The procedure will help researchers identify which part of the video should be analyzed for the significant influence of its content on viewers.

Third, this study explores the relationship between multimodal features and viewers' affective responses. Researchers can better understand what type of response should be expected given a set of features of videos. The study will potentially help researchers create a universal affect interpreter for arbitrary videos and push the boundary of cognition, education, and healthcare.

The main contributions of my study are summarized below:

1. I built a scalable autonomous data collection framework for online TSCs resources, which significantly increases the efficiency of data collection and reduces manual intervention. Multiple strategies are explored and tested in this study.
2. I built a cross-platform dataset generation tool for online TSCs with a graphical user interface (GUI) that will be available to the research community under an open-access license. Researchers constrained by budgets will benefit from this.
3. I conducted a comprehensive experiment to demonstrate the efficiency of the proposed data pipeline and create two large-scale TSCs datasets covering more diverse topics and video playback length compared to any publically available resource.
4. I analyzed the sentiment accuracy of my datasets against the actual video

sentiment. Results show how video tagging could benefit from my automatic data labeling framework. The experiments provide insight into the efficiency of the proposed temporal sentiment mapping system.

### **1.D Study Outcomes**

This study is mainly focusing on three goals:

1. Creating an effective temporal sentiment mapping framework for online temporal data.
2. Generating large-scale time-sync comments datasets for future sentiment studies and video tagging.
3. Providing useful insights into the performance of the data collection system.

Therefore, the first outcome will be a data mining framework that could produce an online time-sync comments (TSCs) dataset effectively. The framework should work under a constrained budget and reduce the difficulties in detecting, producing, and processing such information for researchers.

The second outcome will be large-scale TSCs datasets generated by the framework. The datasets should cover a variety of video contents and topics to reduce bias. Researchers can use the datasets in different multimedia tasks such as video classification, segmentation, and topic modeling. The datasets should also exceed the existing ones in both quantity and quality.

The third outcome will be a deep analysis of the collected datasets. The accuracy of the sentiment labeling techniques should be examined and checked through state-of-art methods. Different mapping strategies need to be compared and tested. This outcome could provide valuable insights into the performance of the proposed online temporal sentiment mapping system.

## **II Literature Review**

In this section, I will review the most recent literature regarding Time-Sync Comments (TSCs) and their usage in various research. I also summarize applications in sentiment analysis by leveraging TSCs as their primary input and discuss vital features of study media in the form of video.

### **II.A Time-Sync Comments**

Time-Sync Comments, also known as TSCs, are live comments posted by multiple viewers that synchronize with a video's playback time. It was first used in automatic video tagging studies in 2014 (B. Wu et al., 2014). Since then, it has drawn lots of attention in multiple fields, such as video recommendation (Ping, 2018; Z. Wu et al., 2019; Yang, Gao, et al., 2019), highlight shot extraction Xian et al., 2015, clip segmentation (Z. Wang et al., 2020; Fraser et al., 2020), video semantic representation (Famin et al., 2019), user experience improvement (Liao et al., 2018), etc. It has become one of social media research's most critical data sources.

The basic form of the popular live commenting interface consists of two parts. Fig Figure II-5 represents the basic idea of posting a live comment on the video.

Numerous viewers can comment on the right section as the video plays on the left. Each viewer can also review the previous comments and chat with others. Several variants of live commenting interfaces exist based on different platforms. For instance, Bilibili adopts a form of comments called 'danmaku,' originating from the Japanese popular video platform Niconico, allowing users to post their comments like a bullet moving across the screen as videos play. Each comment is synchronized to the video timeline and creates a 'mini forum' around certain video parts. Audiences may continuously discuss through the video playtime and can join the discussion. Some platforms even allow reviewers to post their thoughts after reviewing previous posts.



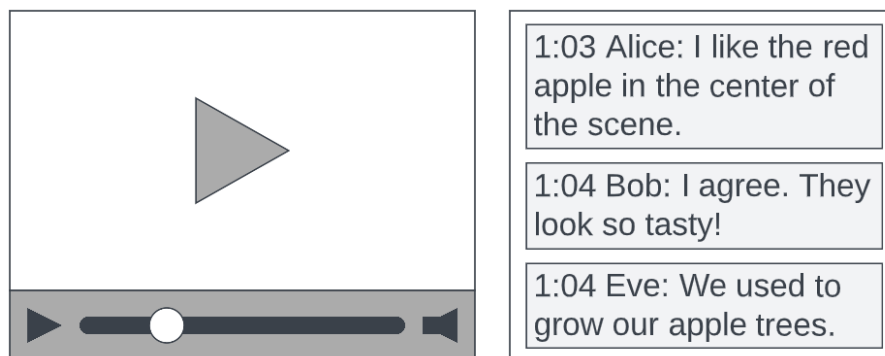


Figure II-5: The above figure shows a simple mock-up of the live commenting interface. Audiences on the right side constantly comment on the video streams playing on the left-hand side.

Compared with traditional manually labeled video tagging, there are several advantages of TSCs tagging:

1. **High availability:** Online data is easily to access.
2. **Robustness:** It contains less bias compared with manual labeled ones.
3. **Good accuracy:** It could generate good representation of videos.

In the first place, live commenting is popular around the globe, and they are easy to access from the internet. Live stream platforms, such as YouTube Live, Tiktok, Twitch, Facebook Live, Niconico, and Bilibili, increased during the past decade. According to one online research (Restream Team, 2021), Twitch averaged two billion watched hours per month, remaining top in the gaming sector in 2021. Meta (Facebook) also claims that compared with pre-made videos, people are willing to spend three times watching Facebook Live videos (Meta, 2016). Hundreds of thousands of live streams were posted online each year, and millions of people commented on them as they watched. Using public-available live comments as a source of video tags can increase the accessibility of data in research.

In the second place, online comments are posted by multiple audiences from different groups (regions, ages, genders, etc.), which are relatively more objective than traditional manual-labeled video tags. Usually, a single frame of TSCs is created by

multiple viewers, potentially averaging out any personal preferences of viewers. Compared with manual-labeled video tags created by a selected group of volunteers in traditional ways, online TSCs are more robust to errors and bias, making them ideal for research in the multimedia field.

Finally, they are highly correlated and synchronized to the video contents. Each comment is attached with a specific playback timestamp. Once the comments are posted online, audiences are allowed to review, edit and remove their previous comments. In essence, most discussions are related to some objects or topics within scenes. They can be summarized using natural language processing (NLP) algorithms to create a short description of the topic. This idea leads to leveraging online TSCs as crowd-sourcing video labels to generate high-resolution video tagging datasets on a large scale without manually labeling the videos.

## II.B Time-Sync Comments Datasets

Very few studies in the literature use live comments for annotating video events. The research on using such information started in 2016, and some studies named it Time-Sync Video Comments (TSCs) instead. A typical TSCs application could be recommending, detecting, segmenting, and finding highlights of videos. However, only a few datasets are available in the literature. Table 1 shows a list of currently available datasets.

Name	Year	Video	Length	Comments	Type	Language	Source
Livebot	2019	2361	114 h	895,929	General	Chinese	Bilibili
TSC D1	2019	287	3015 m	227,780	General	Chinese	Bili, AF
TSC D2	2019	180	4038 m	569,996	Anime	Chinese	Bili, AF
BJUTSD_V2	2020	3682	851 m	120,000	Adult	Chinese	Internet
VideoIC	2020	4951	557 h	5,330,393	General	Chinese	Bilibili

Table 1: A list of large-scale TSCs datasets. AF = AcFun, Bili = Bilibili.

Ma et al., 2019 firstly published a large-scale live comment dataset contains 2,361 videos and 895,929 comments. The dataset covers videos in 19 categories on the

Bilibili website. The authors explored the automatic live commenting pipeline for videos with recurrent neural networks (RNN) and transformer. The result shows Fusion RNN could produce the most fluent results, and transformer could produce the most relevant and correct results in comments generation.

Yang, Wang, et al., 2019 crawled two datasets from the Bilibili and AcFun websites for video tagging purposes. The TSC D1 dataset contains 287 videos with 227,780 comments in music, sports, and movie categories. The TSC D2 dataset has 180 Japanese animation videos with 569,996 comments. The researchers tested different topic modeling methods to identify the best embedding vector algorithms for short comments and concluded that word2vec performs best. For clustering tags by their temporal semantic similarities, the proposed SW-IDF clustering algorithm performs better than TF-IDF, TX, BTM, and GSDPMM methods.

L. Wang et al., 2020 collected an adult classification dataset consisting of 1154 regular live videos, 2528 adult streamer live videos, and 120,000 bullet screen comments. They use a pre-trained CNN for visual features, a Bi-GRU network for temporal context features, and a fine-tuned BERT model for live comments in the dataset. Their experiment shows that using live comments solely for pornography detection is competitive with using spatial and audio feature fusion (64.97% versus 66.6% in accuracy).

W. Wang et al., 2020 published the VideoIC dataset, which contains a higher density of live comments than the Livebot dataset. The dataset contains over 5 million comments with 4951 videos from Bilibili. The study proposed a multimodal multitask learning framework for comments generation by predicting semantic and temporal relationships across different modalities. Their generated comment's quality is significantly higher than the result from Fusional RNN (LSTMs) and Unified Transformer (multiblock transformers).

Duan et al., 2020 explores modeling the interaction between comments and video

features using a multimodal matching transformer. Their work is based on a trimodal attention mechanism algorithm and reported to be more accurate than the bimodal baselines (S2S, Fusional RNN, and Unified Transformer).

As most of the datasets available in the literature are crawled from non-English websites, it is necessary to build a new dataset in English. So, I proposed creating a TSCs dataset using English as its primary language setting and exploring the possibility of recognizing video events with the help of live comments.

### **II.C Sentiment Analysis**

Sentiment analysis is defined as a computational study of understanding people's opinions, appraisals, and emotions toward entities, events, and their attributes (Liu, 2010). It is a quantitative measurement of one's attitude towards a particular entity or topic. Many studies and applications use sentiment analysis in the decision-making process. Sentiment analysis is essential for some fields such as public policy-making or marketing since people's final decisions could vary on the result.

Liu, 2010 summarized the problem of sentiment analysis as a text classification problem where textual opinionated documents or sentences are classified into different categories. Due to the nature of the complexity of human sentiment, a number of categorizations exist in literature (Plutchik, 1980, Ekman, 1992, Parrott, 2001).

Some researchers prefer to use a polarity classification of three types: positive, neutral, and negative for their study (Bakshi et al., 2016). Others adopt more complex models. Ekman, 1992 suggested that each type of sentiment could be categorized into six basic emotions: joy, sadness, anger, fear, disgust, and surprise, which subset from eight feelings proposed by Plutchik, 1980 (Figure II-6). Francisco and Gervás, 2006 summarized three categories of positive sentiments as pleasantness, activation, and dominance. Pearl and Steyvers, 2010 used a custom set of emotions of politeness, rudeness, embarrassment, formality, persuasion, deception, confidence, and disbelief during their research. S. Mohammad and Turney, 2010 and S. M. Mohammad and Turney, 2013 also developed a similar classification consists of anger, anticipation,

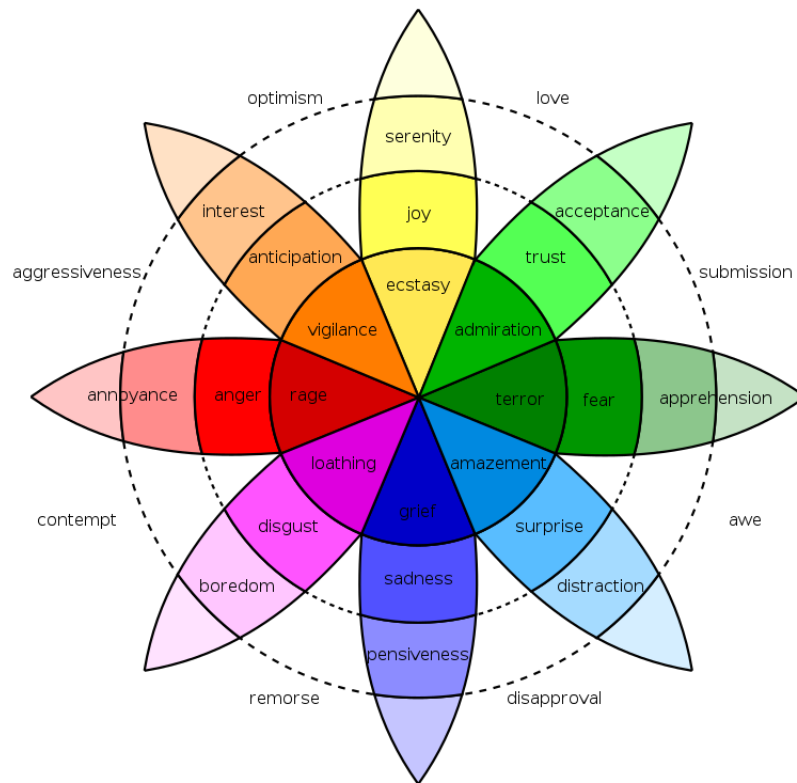


Figure II-6: Plutchik's emotion wheel.

Source by:

[https://en.wikipedia.org/wiki/Emotion\\_classification#/media/File:Plutchik-wheel.svg](https://en.wikipedia.org/wiki/Emotion_classification#/media/File:Plutchik-wheel.svg)

disgust, fear, joy, sadness, surprise, or trust inspired by several empirical research in psychological, physiological domain.

## II.D Textual-based Sentiment Mapping

Two principal components exist for textual material: facts and opinions. Facts are objective expressions relate to specific entities and events, while opinions are subjective expressions of an audience's emotional reactions or beliefs towards the content (Liu, 2010, Feldman, 2013). As we are primarily interested in analyzing audiences' internal sentiments, I want to exclude factual information. Then, I need to study underlying emotions within the refined text to extract the viewer's sentiment. Finally, I also want to understand which entity or event triggers those sentiments. Therefore, the three critical tasks of sentiments analysis can be summarized, mentioned by Liu, 2012, as following:

1. **Subjectivity Classification:** Textual material may or may not express true sentiments from the author(s). The objective text should be filtered first to get accurate opinions.
2. **Sentiment Classification:** After extracting opinionated text, we can use NLP algorithms to calculate the opinion's polarity. In most cases, a binary polarity classification such as positive or negative is sufficient for the research. Otherwise, a multi-class category might be used to extract more information from text.
3. **Entity/Topic Extraction:** Once sentiments are extracted, it is crucial to identify the topic on which the opinion is expressed. Sometimes, it is also necessary to know which entity holds those sentiments as well.

According to Liu, 2012 and Kruse et al., 1999, textual data can be break down into three levels: Document level; Sentence level; and Word level.

1. **Document level:** The whole document is treated as a basic unit for sentiment analysis.
2. **Sentence level:** Only subjective sentences are used for the sentiment analysis. Objective sentences are ignored for the document.
3. **Word Level:** Unusually adjectives are considered as a representation of sentiments. But verbs, adverbs, and nouns could also express subjectivity.

Each level of analysis can be viewed as a different resolution measurement of genuine sentiment. In most cases, a higher level of sentiment is constructed by averaging sentiment from lower levels. For instance, a sentiment of sentences is built on top of individual words' sentiment.

## **II.E Sentiment Labeling via Live Comments**

Live video comments are audiences' textual-based feedback towards certain entities or topics in video streams. They are composed of both subjective opinions and objective facts. The affective information remain in the subjective component can be exploited as annotations to the original video stream. A standard procedure is performing a

coarse-grained sentiment recognition on the text sequence. The Sentiment Analysis (SA) pipeline provides us a powerful way to achieve this. Figure II-7 shows a polarized sentiment value is assigned to a live comment after passing its text sequence through a SA pipeline. It can be viewed as mapping the live comment from a word vector space into a polarized sentiment space.

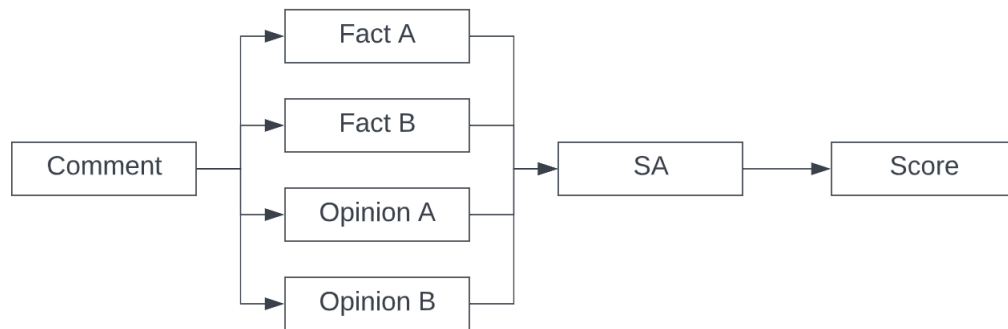


Figure II-7: The sentiment conversion process.

Most SA frameworks run a pre-trained lexicon against input text sequences. The method ignores the semantic meaning of live comments and only considers sentiment information within subjective parts. This procedure results in assigning a single numerical value for the input text. Considering each comment is associated with a unique timestamp, the output of such sentiment score can form a temporal sequence. Figure II-8 shows after acquiring a series of sentiments from comments, and a time series annotation is constructed according to their timestamp.

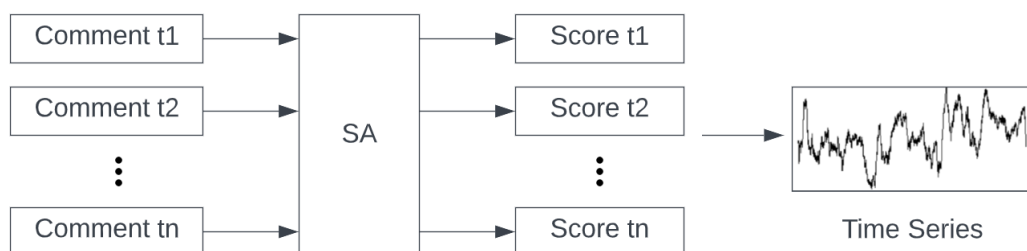


Figure II-8: The workflow of creating a sentiment annotation time series

Please note the occurrence for live comments is not uniformly distributed across time. Converting the sentiment sequence to time series requires choosing a proper window size  $w$  for averaging sentiment scores within the window length. Let  $t_i \in \{t_1, t_2, \dots, t_n\}$  represents timestamp where live comments occurs, and  $f(t)$  represents empirical distribution of the occurrence of comments given time  $t$ . The average of sentiment annotation can be calculated as (Equation 1):

$$A(t_j, w) = \frac{\sum_{t_i > t_j - \frac{w}{2}}^{t_i < t_j + \frac{w}{2}} f(t_i) sa(c(t_i))}{w} \quad (1)$$

where  $c(t_i)$  represents free-form text of a comment,  $sa()$  represents the SA procedure,  $c(t_i)$  represents comments at time  $t_i$ , and  $w$  represents a fixed-size window.

## II.F Event Detection through Sentiment

The overall procedure of video event detection can be viewed as a semi-supervised classification problem (Figure II-9). In reality, we do not have accurate labeling of different "events" in a video stream. However, audiences labeled their sentiment with live comments softly during watching. So we can create a set of "pseudo labels" of sentiment events and perform a supervised study between the multimodal features and the video events.

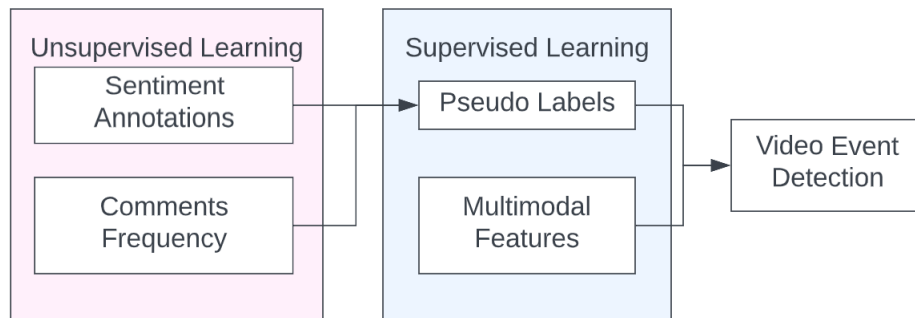


Figure II-9: An illustration of the semi-supervised video event classification procedure. A supervised learning classifier uses the "pseudo labels" generated by the previous unsupervised learning step as targets to learn from multimodal features.



The first step of producing "pseudo labels" can be carried out with a unsupervised change point detection (CPD) algorithm on time series data. There are plenty of such algorithms to choose from. Aminikhanghahi and Cook, 2017 summarized unsupervised CPD algorithms as shown in Table 2.

<b>Type</b>	<b>Methods</b>
Likelihood Ratio Method	CUSUM, Change Finder, KLIEP, uLSIF, RuLSIF, SPLL
Subspace Method	SI, SST
Probabilistic Method	Gaussian Process, Bayesian
Kernel Based Methods	Kernel Based Methods
Graph Based Methods	Graph Based Methods
Clustering	SWAB, MDL, Shapelet, Model Fitting

Table 2: A list of commonly used unsupervised CPD algorithms (Aminikhanghahi and Cook, 2017).

Van den Burg and Williams, 2020 evaluated the performance of several CPD algorithms on a human-annotated dataset and claimed the Bayesian Online Change Point Detection (BOCPD) perform best on both univariate and multivariate time series when hyperparameter tuning is performed. The unsupervised learning phrase in this study can adopt a similar algorithm for time-series segmentation and labeling.

The second step tries to build a connection between "Pseudo labels" and multimodal features. It uses a supervised learning algorithm to classify different video features. Since the feature extraction step involves using an autoencoder to combine multimodalities (Figure II-10), a reasonable choice is to use an Artificial Neural Network (ANN) to reuse the representation layer. Figure II-11 shows the usage of the autoencoder's hidden layer from previous unsupervised learning as an input to an ANN.

After training, the network could identify if a video event is triggered by looking at the representation of visual, aural, and textural features. This model can be generalized on out-of-sample datasets such as new online streams or real live video clips and predict if any behaviors in such video content could cause salient changes in viewer's sentiment.

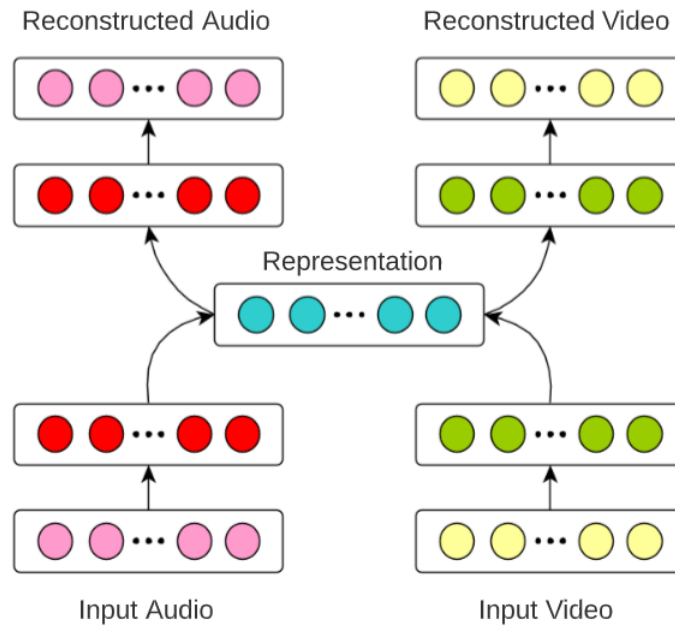


Figure II-10: A deep autoencoder framework creates a multimodal representation by learning features from individual modality input (Ngiam et al., 2011).

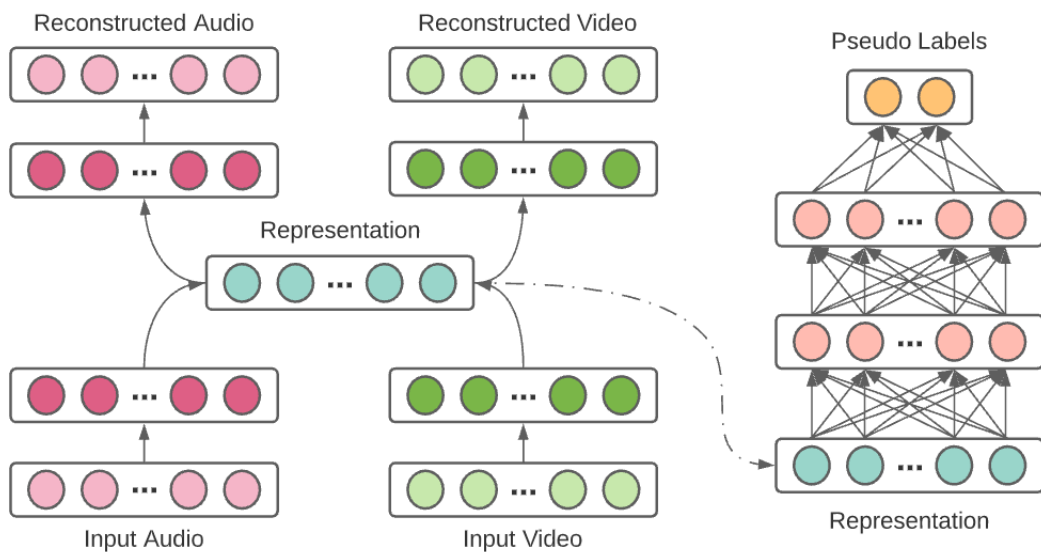


Figure II-11: The supervised step uses hidden layer representation as input. It tries to classify these representations into different pseudo labels.

## II.G Modality Features in Videos

Video streams consist of three significant types of information: visual, aural, and textual, delivered to audiences through video and audio channels. Each type of data is called a modality. Audiences' perceptually salient events can be triggered by changes

in either single or multiple forms of modalities (Evangelopoulos et al., 2013). Visual events include movement, change in objects' shape and appearance, or property of the ambient environment. Aural events can be recognized as changes in texture or tempo of sound. The context of a dialogue or narration can be viewed as a textual event that carries linguistic meanings. Previous studies showed that using multimodal features can create a better representation of a video event compared with using unimodal features (Poria et al., 2017).

Figure II-12 shows a video stream can carry three modalities of information which contribute to the perceptually salient events of audience. Using the multiple modalities will provide higher accuracy and more robustness in video event recognition. Therefore, the proposed representation of video streams will involve using both visual, aural and textual information.

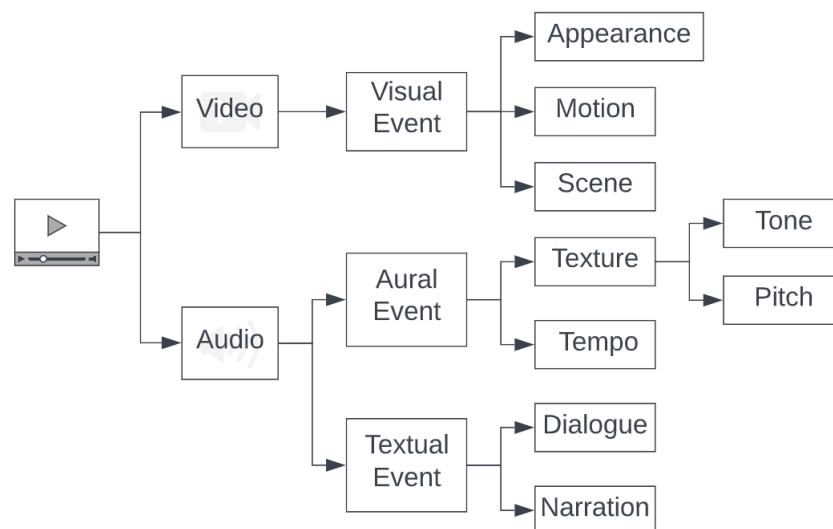


Figure II-12: Visual, aural, and textual modalities exist in video streams. They can be used to detect video events.

According to Ngiam et al., 2011, deep multimodal learning can be divided into three steps - feature learning, supervised training, and testing. There are several ways of creating multimodal features in the feature learning phase. Ngiam et al., 2011 shows concatenating different modalities as input performs poorly on producing

cross-modality representation since the nonlinear correlation exists between modalities. They proposed a deep autoencoder framework that first learns individual modality features and then learns shared modality features based on the output of each hidden layer.

Figure II-13 shows the workflow of combining the live commenting features in deep multimodal learning. Live comments will convert into sentiment sequences and align with video modality features. Machine learning algorithms can later be applied to use sentiment labels to detect video events.

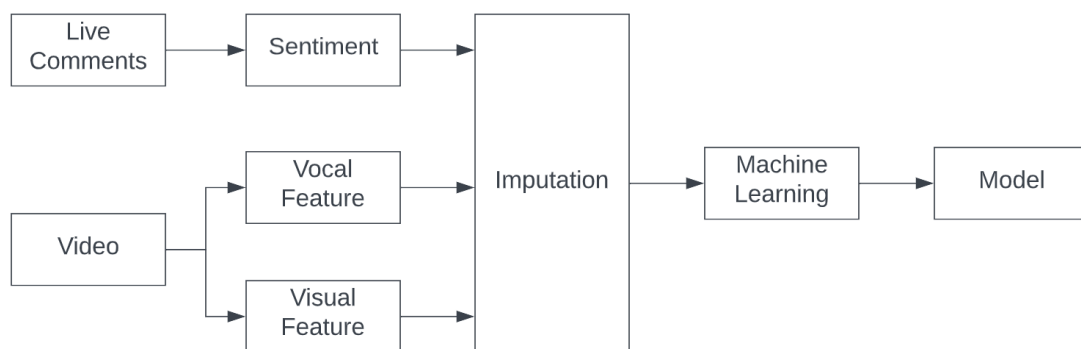


Figure II-13: General workflow of using sentiment labels in deep multimodal learning.

## II.H Data Acquisition

The data acquisition involves gathering unstructured external data online and converting them into a local data structure. Live comments from different websites require different ways to get. Developing a data acquisition pipeline from the ground is time-consuming and is not the primary goal of my research and against the colloquial metaphor of "don't reinvent the wheel." Leveraging existing packages and Application Programming Interface (API) can eventually reduce the time and effort of developing such pipelines.

Pytchat (Taizan-hokuto, 2021) is a python package released in 2021 which could extract comments details from YouTube Chat. Comments attributes such as author name, elapsed time, time stamp, and message are extracted and stored in JavaScript

Object Notation (JSON) format (Table 3).

<b>Name</b>	<b>Timestamp</b>	<b>Elapsed</b>	<b>Message</b>	<b>Type</b>
John	1616334400069	23:14	I like this music!	textMessage
Mary	1616334401217	23:20	Good night everyone	textMessage
Jane	1616334404186	30:41	That is really saaaadd	textMessage

Table 3: A simple example of the output attributes from Pytchat.

Attribute "Name" reveals the authorship of logged comments. Attribute "Timestamp" refers to the Unix Timestamp of a message in time, while "Elapsed" refers to the relative timestamp of a message in the video. Attribute "Message" contains all textual information within a post, and "type" represents whether a comment is textual or not.

Videos streamed online follow different ingestion protocols. Table 4 summarizes several common protocols' name and their streaming prefix. Selecting tools with correct protocols can make data acquisition easier. YouTube Live Streaming supports the ingestion protocols for third-party clients listed in Table 5 ("YouTube Live Streaming Ingestion Protocol Comparison", 2020). Downloaders with compatible ingestion protocols are needed. Streamlink (Streamlink, 2021), a command-line utility that supports multiple streaming protocols (Table 6) could satisfy this purpose.

<b>Name</b>	<b>Prefix</b>
Adobe HTTP Dynamic Streaming	hds://
Akamai HD Adaptive Streaming	akamaihd://
Apple HTTP Live Streaming	hls://
MPEG-DASH	dash://
Real Time Messaging Protocol	rtmp://, rtmpe://, rtmps://, rtmpt://, rtmpte://
Progressive HTTP, HTTPS, etc	httpstream://

Table 4: Streaming ingestion protocols and prefix.

<b>Ingestion Protocol</b>	<b>Encrypted</b>	<b>Video Codecs</b>	<b>Latency</b>
RTMP	No	H.264	Normal, low or ultra-low
RTMPS	Yes	H.264	Normal, low or ultra-low
HLS	Yes	H.264, H.265 (HEVC)	Normal and low
DASH	Yes	H.264, VP9	Normal and low

Table 5: Streaming ingestion protocols for third-party clients.

<b>Platform</b>	Unix-like	Windows			
<b>Protocol</b>	hds:// rtmp://	akamaihd:// rtmpe://	hls:// rtmps://	dash:// rtmpt://	httpstream:// rtmpte://
<b>Quality</b>	audio	160p	360p	720p	1080p

Table 6: Available platforms and protocols supported by Streamlink CLI.

I wrote a piece of python script to test the approach quickly. I used the program to query available comments by giving a video id. The script also searches available video streams with public comments throughout the day and simultaneously starts the recording process. I noticed some chats could not playback once the stream finished, and such comments captured on the fly do not contain elapsed time tagging.

Therefore, pre-selecting a bunch of channels that allow chat replay can accelerate the data collection.

### **III Data Collection Framework**

This study requires a deep analysis of videos associated with annotations of sentiment. To achieve this goal, I decide to use a series of online videos with Time-Sync Video Comments (TSCs) as my primary data source. However, to my knowledge, no public-available datasets are collected from native English-speaking websites. The closest one released on Kaggle is VTuber 500M “Kaggle”, 2021 which collected Virtual YouTubers’ live streams through crowd-sourcing per month from Jan 2021. Therefore, data collection needs to be performed before data analysis. In this section, I will discuss the strategies used in my study.

This section will first propose and design a framework that could easily collect TSCs data directly from English native speak websites with minimum human interventions. The proposed data mining process contains two-phase: live stream detection and video data collection. Then the section will discuss the difficulties and challenges the data pipeline met and introduce the framework’s theoretical idea and implementation details.

#### **III.A Data Collection Challenge**

Hundreds of millions of online videos are available, but not all contain Time-Sync Video Comments (TSCs). And for those videos containing TSCs, there are no easy ways of collecting them on a large scale. In addition, the number of TSCs and video length become vital for the study because low TSCs density can result in a lack of annotations from text-to-sentiment conversion. Due to the limited official support of live chat comments API, the only way to collect live chats is through a web page. I will discuss each of the challenges in detail later. I proposed and built an automatic pipeline for large-scale data crawling to meet these criteria.

##### **III.A.1 Data Source Selection**

The first step of creating a large-scale TSCs dataset is identifying the data source, including the video and associated TSCs. According to previous research (Ma et al., 2019; Liao et al., 2018; W. Wang et al., 2020), crowd-sourced data labeling by popular

online video's live comments proved to be efficient for creating such datasets.

However, previous studies have two major drawbacks in the data source selection.

First, online videos selected by the previous studies have limited content and form.

The TSCs were mainly collected from non-English sites such as Bilibili and AcFun, which were primarily posted by East Asia audiences. Having a regional audience group means some postings, such as slang or abbreviation, may differ from other parts of the world. Moreover, the platforms are heavily influenced by Japanese anime culture, which suggests the associated TSCs may also lack variety from both cultural and national perspectives.

Second, the social platforms chosen by previous studies are not the most popular ones worldwide. According to one recent online research (Thomala, 2021), the monthly active users (MAU) of Bilibili was around 267 million in China, while YouTube was counting 845 million monthly active users (MAU) simply from iOS devices and 2.3 billion users in total in the third quarter of 2021 worldwide. Getting TSCs from YouTube is a better choice since the real MAU on YouTube is 8.6 times more than Bilibili, which provides more variety TSCs and coverage of different population groups.

### **III.A.2 Live Stream Identification**

The second step of gathering a TSC dataset from an online platform is to identify many videos with a good number of TSC attached. Previous research using online media such as Bilibili and AcFun can directly extract a list of video URLs through official APIs. However, official Youtube APIs do not provide such accurate search results. For instance, the returning results for a keyword could only contain a random portion of all available live streams. This problem forces the crawling into a multifold progressive rather than a one-shot process, which increases the difficulties of TSC data mining and cleaning.

Another issue related to the Youtube APIs is they could not provide updated video information. Newly uploaded live streams may still be processed within the Youtube



system, and there is no guarantee when these videos will be available to query through the API. To make things worse, sometimes, the APIs may return videos that are already removed or deleted. To download TSCs correctly, I have to check the availability of each video which decreases the overall efficiency of data mining.

### III.A.3 Official APIs Quota Limit

Although YouTube Data API are free of charge, Google implements a hard daily quota cap for a single account. According to their website (“YouTube Data API Overview”, 2020), a single account could only have a limited number of queries per day. By default, google allocates 10,000 units for each user and applies different quota units for different types of queries. For example, a search request costs 100 units, while a video upload request costs 1,600 units.

Table 7 shows the different units for the same method in types of quota defined by YouTube Data API. The same ‘list’ method triggered a huge unit expense in searching videos and captions, which placed an additional roadblock to obtaining many newly uploaded YouTube streams through the API. For instance, we are limited from searching more than 100 times per day from a single account. Although using multiple accounts could potentially solve this problem, it introduces more problems and complicity from the implementation point of view. An alternate and simply solution is needed.

<b>quota</b>	<b>method</b>	<b>cost</b>
<b>activities</b>	list	1
<b>captions</b>	list	50
<b>channels</b>	list	1
<b>channelSections</b>	list	1
<b>playlistItems</b>	list	1
<b>search</b>	list	100
<b>video</b>	list	1

Table 7: This table compares different quota units of the same method between various categories (“YouTube Data API Overview”, 2020).

### III.B Proposed Framework

The proposed data pipeline can be described by two components: Live Video Detection and Data Collection. Figure III-14 shows a high-level overview of the data mine process. The left part figure in blue shade represents the internal workflow of the video prober, while the right part in yellow tint represents the data collection process. Individual components run separately; for example, the video prober runs per day while the data collector occasionally runs instead. Two parts work together and form a complete data collection framework. A detailed description is given in the following subsections.

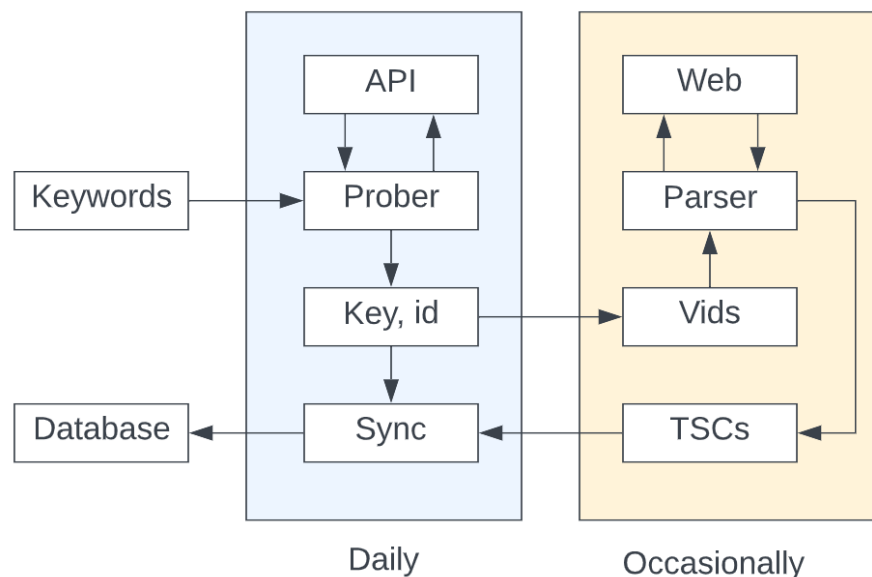


Figure III-14: This graph shows a high-level overview of the proposed data pipeline's workflow. The video prober on the left uses a set of keywords to search for related videos per day. The parser on the right side occasionally checks for all searched video IDs and collects their meta-information and TSCs.

#### III.B.1 Live Video Detection

In order to locate videos with TSCs, I need a prober that could tell whether a video contains valid TSCs; in other words, I need to locate videos having live chats.

Identifying the live streams' IDs on YouTube is the first phase of my data collection. I used a public-available data API to query YouTube Live for videos finished streaming. The main reason for only using completed streams is the timestamp of associated live

comments is unavailable for live streams. There are some disadvantages of this route. First, the query result does not include the most recent videos uploaded to YouTube. Second, each query will return duplicate results for the identical video IDs. Third, the data API could not provide a complete list of historical video IDs. However, this is the only budget way I could figure out to locate TSCs videos on YouTube, so I decided to stick with it in this study.

I scheduled my video prober running daily to get the most recent uploading streams on YouTube. Each day, the prober queries a list of keywords to YouTube through the API and stores each video ID locally. After each query, I registered a background process to clean the results and remove the duplicated IDs. The newly detected video IDs are then merged with my previous results and synced to my local database. This step will guarantee us a list of unique streaming IDs published on YouTube associated with different keywords. The process constantly produces new results per day and refreshes my database.

### **III.B.2 Data Collection**

The second phase of the proposed data pipeline is data collection. Once I got a list of video IDs, I started to verify their live chat availability since some videos disabled the chats' playback or were removed by the authors. Meanwhile, I created a custom script to parse the meta-information out for each video, including their likes, dislikes, languages, ratings, etc. The script runs relatively slow compared with the previous video prober, so I registered it to run occasionally. The right side in the orange shade box of Fig Figure III-14 shows the internal workflow of the data collection phase.

There are two ways of getting meta-information: through API or parsing directly from the webpage. I tried both of them and decided to keep with the API route since querying HTML is time-consuming and inefficient for large-scale data mining. However, due to the limited quota per user account with the official APIs, alternative APIs have to be called, and an extra fee may exist depending on providers.

Once a list of videos with valid TSCs sources is verified, I can download the

associated TSCs through a custom downloader program. The downloader can either be integrated with the main program or run as a standalone application. The downloaded TSCs can later attach with associated meta-information collected by my web crawler to provide a complete description of live stream commenting.

### **III.B.3 Summary of the Framework**

Figure III-15 illustrates the internal sequence of the complete data pipeline. The main thread of the proposed data pipeline first passes a set of keywords to a video prober on a daily basis. Then the prober returns a list of live stream ids to the main thread by querying API and stores it into a database. Next, the main thread sends a list of video IDs to a web crawler to collect their meta information and verify if any TSCs source available from the steams. Finally, all TSCs are downloaded based on the previous result and stored locally.

The whole data pipeline should work automatically with minimum human intervention. Once the framework runs, it will grab the latest TSCs data published online per day and update the local storage version of the data. In the ideal case, I can run this framework on multiple machines with multiple API accounts to scale the volume of the data collected. The efficiency of this framework will depend on the network, the number of machines, and the number of results from API calls. The stability of this framework will depend on the internet connection, the server-side response, and the local platform.

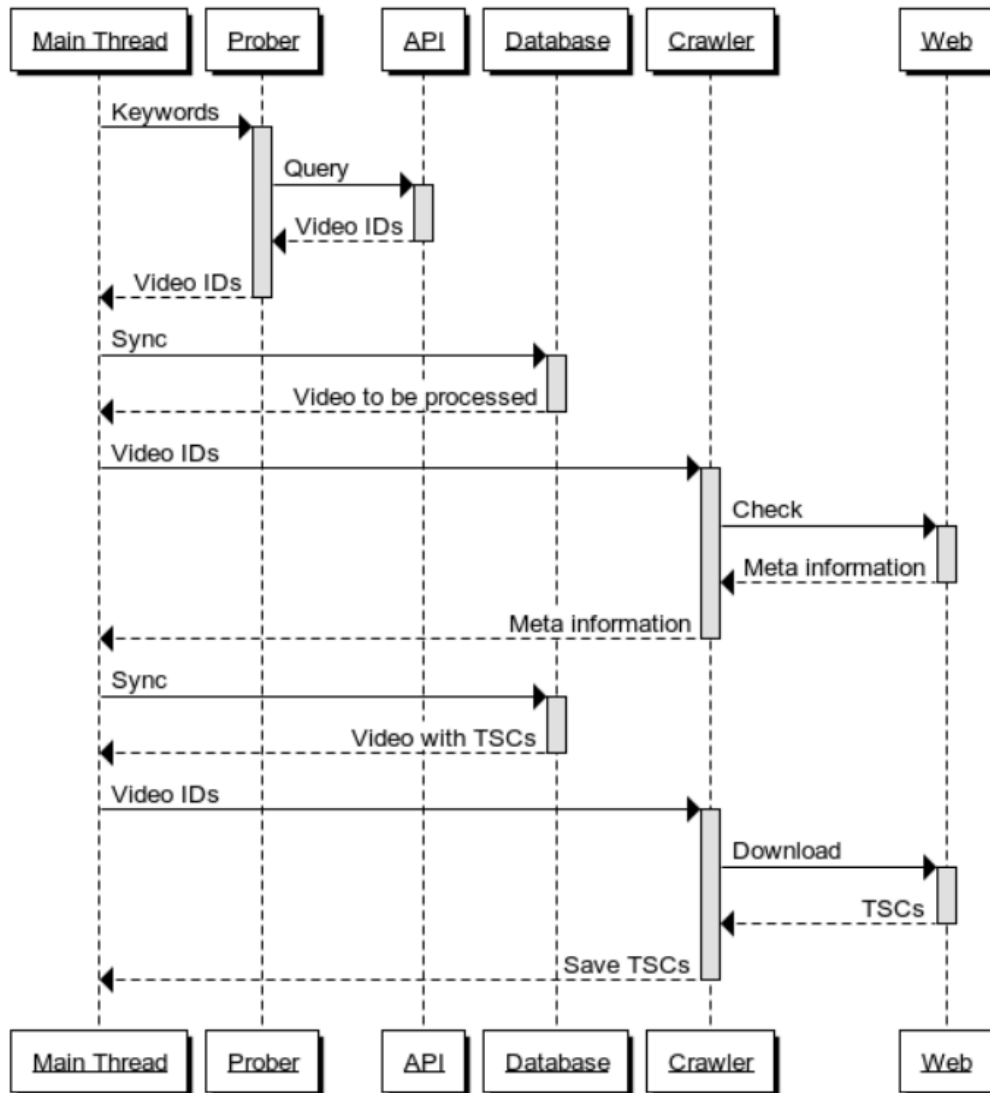


Figure III-15: A sequence diagram of the proposed data pipeline.

### III.C Framework Development

The framework was first implemented on my laptop and later tested on my secondary desktop. Because two machines are on different operating systems, it is natural for me to try to use a scripting language to write the code to run the framework without compiling it on multiple machines. Therefore, I select the most popular script for the present: Python, to reach the goal. The system environment configurations are summarized below.

1. System Type: x64-based PC.
2. Package Management: conda 4.12.0
3. Language: Python 3.8.11

Initially, I wrote separate script pieces to test out individual functions and started collecting data on a small scale. Then I made several improvements to stabilize the code and mining process. I soon realized that the data collection could take days, and a synchronization mechanism needed to be implemented. As the framework gradually grew, the testing posed on large-scale data collection suggests having a main graphic interface could reduce testing time and save debugging efforts. Therefore, I built a graphic user interface (GUI) through Python's standard TK interface to quickly adjust and test different settings. Choosing the TK environment is simple: it is installed by default within the python environment. If I could implement my framework with the TK library, I could have an interface on almost any platform with Python installed.

Then I designed the graphic interface into three sections: Selections, Options, and Outputs. Figure III-16 shows an example of the pipeline's graphic layout. The advantages of having such a visual layout include:

1. Allowing users to interact with the data pipeline easily.
2. Running on portable devices and different operating systems.
3. Debugging and acknowledging the working state of the framework.

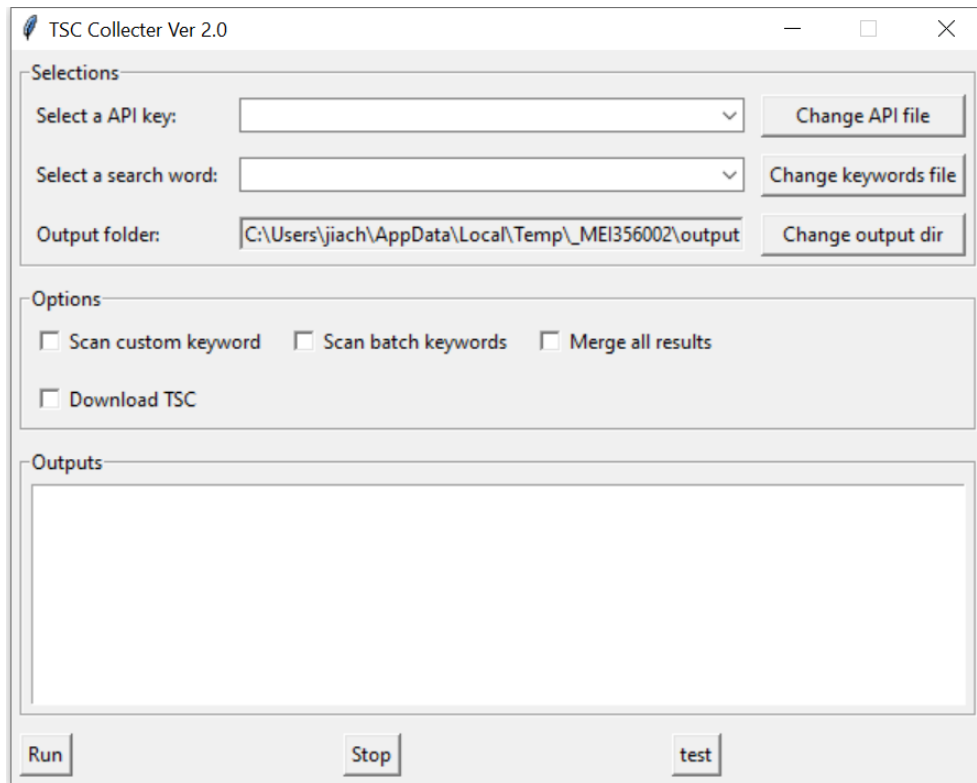


Figure III-16: A screenshot of the graphic layout of the proposed data pipeline.

### III.C.1 Selections Interface

The Selections section includes three user-defined fields: API key, search words, and output directory. Figure III-17 illustrates an example inputs of these fields. The data pipeline uses a third-party Youtube v3 API from RapidAPI.com (RapidAPI, 2020) to communicate with the Youtube database in the background. Thus, users need to provide a valid Rapid API key to use this pipeline—the drop-down list on the right-hand side lists all valid API keys saved in local files.

The search field enables users to input the desired keyword for searching manually. However, it also allows users to import a list of keywords from a local file. The file selection window can be triggered by the button on the right side.

By default, the Output folder field is set under the same directory of the running application. But users can change the output location to any desired folder by using the right-hand-side button as well. The output directory will be the folder that contains all

searching results and downloaded TSCs.

Figure III-17: This screenshot shows the user interface of the Selections section. A custom keyword entry "games" is input into the search field.

### III.C.2 Options Interface

The current version of the pipeline-interface contains four running options. These checkboxes modify the workflow of the software. Figure III-18 shows an example set of options searching all videos associated with a single keyword "games" before downloading related TSCs for each video. Users can run a collection of searching keywords imported from the Selections entries by checking the "Scan batch keywords" options.

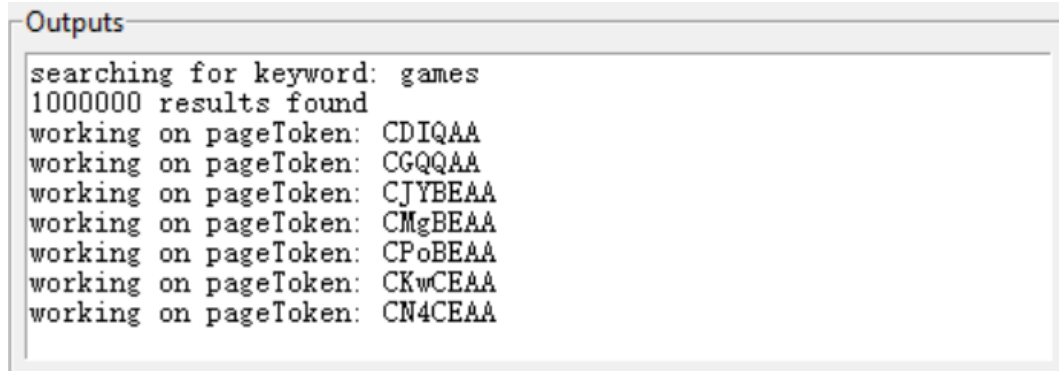
Figure III-18: This screenshot shows the user interface of the Options section. The example shows a running sequence of searching a single keyword, merging previous results, and downloading all TSCs.

### III.C.3 Outputs Interface

The Outputs section provides an easy way to let users monitor the internal running status and analysis the background process. Essentially, it redirects all background console outputs to the front, including custom outputs, warnings, or unexpected errors. Figure III-19 shows an example of searching all related videos on YouTube through



API for a search term "games."



```

Outputs
searching for keyword: games
1000000 results found
working on pageToken: CDIQAA
working on pageToken: CGQQAA
working on pageToken: CJYBEAA
working on pageToken: CMgBEAA
working on pageToken: CPoBEAA
working on pageToken: CKwCEAA
working on pageToken: CN4CEAA

```

Figure III-19: This screenshot shows the user interface of the Outputs section. The example shows the internal progress of searching all related videos for a given searching term "games" on YouTube.

#### III.C.4 Mobile Application

As the data pipeline's interface was coded with default python libraries, it is possible to run it on mobile devices. Figure III-20 shows a screenshot of using an android tablet to start up the application. No extra configuration is needed for devices with the correct python environment installed to run the data mining framework.

I also tried to package the application to be executable through the PyInstaller library ("PyInstaller 5.1", 2020). Once converted, all necessary python environment configurations were included within the application. The data pipeline can run on Windows machines as a standalone application without python installed. However, to create a standalone application on other operating systems, the dependencies bundling step need to perform on the targeting systems. For instance, if I want the executable to work on Mac OS, the converting need to be done on a Mac machine.

#### III.D Data Collection Experiment

My tool is designed explicitly for YouTube streaming videos, so I tested it out in 2021. Each data mining experiment takes roughly one month. Before the experiments, several small-scale sample runs were carried out to check the stability and efficiency of the tool. Then I decided to test it with different types of keywords on a large scale.

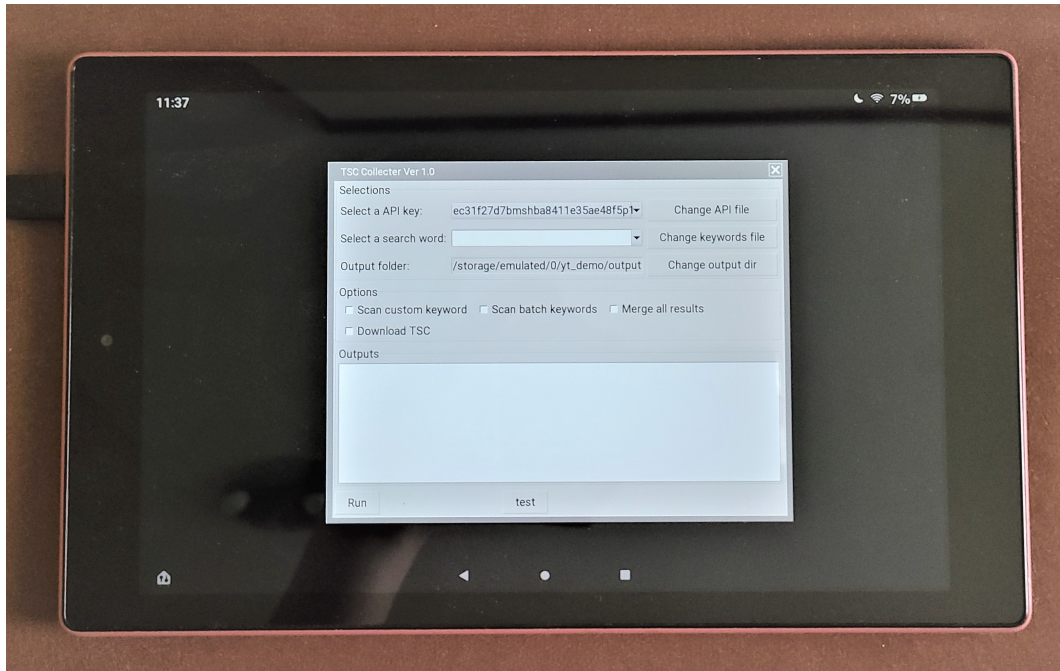


Figure III-20: Running the data pipeline on a mobile device.

Using the proposed pipeline, I collected two datasets with two different sets of keywords: LST-YF20 and LST-YT1000. LST-YF20 contains the top 20 fruits sold in America from the Packer.com (The Packer.com, 2019). I picked this list of keywords simply because my initial testing keyword happens to be "apple." Then I extended my searching terms broadly based on Mondovo's top 1000 monthly average searched words on Google (Mondovo, 2020).

#### III.D.1 LST-YF20 Dataset

The live streams of LST-YF20 were first collected from my YouTube video prober described above. I registered the video prober as a daily cran task from July 10th, 2021, to August 1st, 2021. Two keywords: "apple" and "pear," exclusively ran from July 3rd, 2021 to August 1st, 2021. I did this because we want to check the stability of the daily scheduling system first and add other keywords later once it is verified. Figure III-21 shows the growing number of observations I collected for each fruit during this period. I got 18,852 live streams at this point and let the prober continue running on a separate machine afterward.

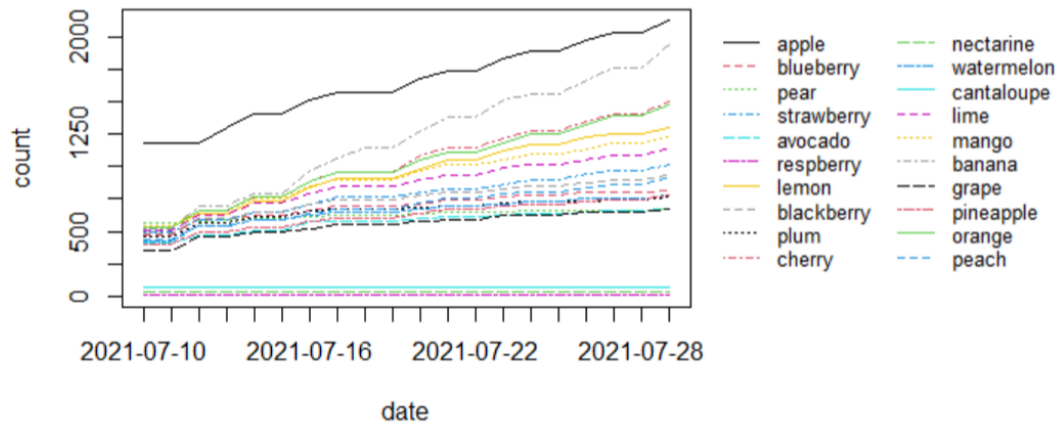


Figure III-21: The plot shows the growing number of observations detected from the video prober on 20 fruits keywords. I collected this data during a 23 days experiment, starting on July 3rd, 2021.

After I got a decent amount of live streams, I started to verify each of them by gathering essential metadata such as title, description, author name, length of the video, number of views, number of likes and dislikes, etc., through a web parser. I verified 17,857 observations through the parser, which roughly takes around 94.7 percent of total videos. Then I downloaded associated TSCs as much as possible and got 10,509 TSCs which roughly takes approximately 58.8 percent of all videos. I noticed there might be some issues with the TSCs downloader package I used from pytchat (Hokuto, 2020), so I replaced it with a custom downloader later in the LST-YT1000 dataset.

Initially, I tried to verify if the videos had valid live chat sources before downloading their TSCs in LST-YF20. Then I figured out comparing with the time spent on getting and checking valid chat sources, the time complexity of downloading all TSCs became more significant. So I decided to simplify my workflow and let the TSCs downloader run first regardless of any valid TSCs source present for each video. It reduces a considerable amount of time for the data pipeline. I adopted this strategy in creating the LST-YT1000 dataset.

In November 2021, YouTube officially announced it would stop showing dislike count on all the videos across its site (Online, 2021). Due to the significant time spent gathering such information and the fact that they will soon be inaccessible, I stopped

getting such meta-information completely in collecting the data of LST-YT1000. The LST-YF20 seems to be the latest TSCs dataset with the exact number of dislike available around currently.

### **III.D.2 LST-YT1000 Dataset**

There are several significant differences between the LST-YT1000 and the LST-YF20 dataset collecting:

1. The LST-YT1000 was ultimately collected through the graphic user interface we developed, while the LST-YF20 was partially collected through a terminal.
2. I occasionally ran the GUI pipeline during the collection phase instead of crawling the live streams daily through a scheduler. I did this to mimic normal user behavior using the pipeline on mobile devices.
3. I built a new TSC downloader from scratch to avoid unexpected errors as much as possible compared to the previous experiment, which significantly increased the success rate of downloading.
4. I captured specific events besides comments during the streams, such as donations, subscriptions, and channel notifications.

The LST-YT1000 was collected between Nov 7th, 2021 to Dec 3rd, 2021, with eight runs on the proposed data pipeline. Over 500 Gigabytes of TSCs were downloaded from 525,312 live streams detected by my live stream prober. I did not keep records for the specific time of my pipeline running, but we noticed that the average downloading time is roughly around 50,000 videos per day. The total downloading time is slightly over a week, by my estimation. I tried to speed up the downloading part with parallel programming, but it turns out that simply increasing the threads could result in a soft block from YouTube, which seems unavoidable in my case. I then decided to use ten parallel threads to download the TSCs. Figure III-22 shows a random screen of downloading progress during the TSCs collection.

```

Outputs
53/133338
12316 actions found for video 8ZxIAvTRrfY
54/133338
16167 actions found for video tBhpPtVP5M0
55/133338
20248 actions found for video tTkE60NQNUM
56/133338
60158 actions found for video ysPDF2Kr5kI
57/133338

```

Figure III-22: This screenshot was captured during the TSCs downloading step. Commenting actions other than TSCs are also collected.

### III.E Framework Evaluation

By the time of the LST-YT1000 being collected, the scheduled live streams prober on 20 fruits kept running at the same time. Figure III-23 shows the distribution of video counts for each keyword we collected on the first day. Three keywords: "raspberry," "nectarine," and "cantaloupe" only return 4, 29, and 70 results. However, the main cluster on the right-hand side contains the rest of the 17 keywords.

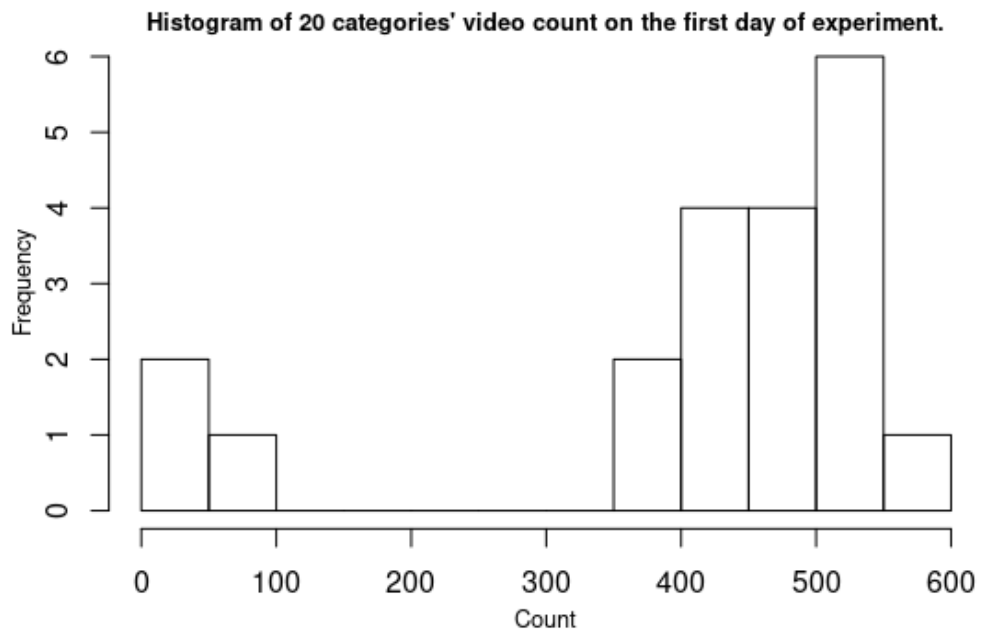


Figure III-23: This histogram shows the distribution of 20 categories of videos we collected on the first day of the experiment. The left cluster contains three keywords: "raspberry", "nectarine," and "cantaloupe." The right cluster contains rest of the keywords.

The difference between keywords could probably result from the different popularity of the fruits. For example, 564 videos associated with the keyword "apple" contribute to the most significant portion of the result, while only 29 videos related to "nectarine" can be detected. We noticed a misspelling in the keyword "respberry," which yields to only four videos returned due to the extremely low popularity of the keyword. However, the rest of the "normal" keywords could provide approximately 300 to 500 searching results. We also noticed some other issues in this long-run experiment. The prober was forced to stop randomly due to some irregular responses from the API. However, these rare corner cases are not a big problem because a second run the next day could recover the missing videos from the last unexpected stop. We kept logging those live streams' ids until this paper was composed. Figure III-24 shows the daily growing trend of the observations within 171 days.

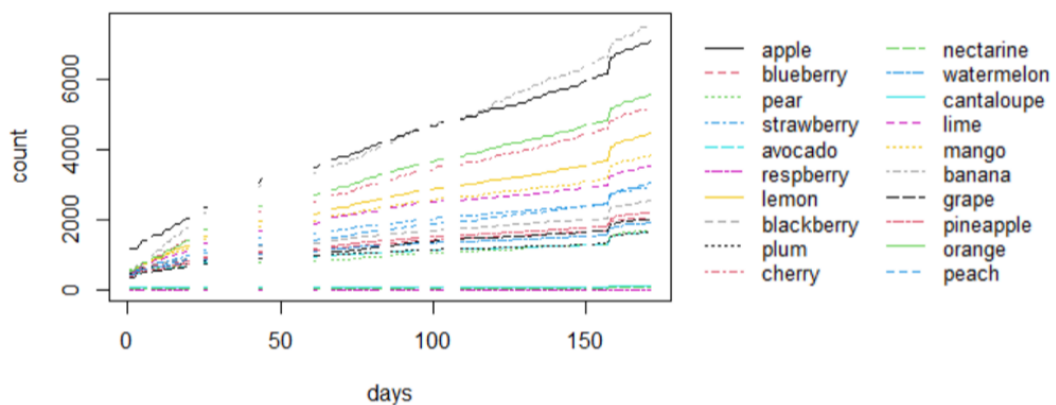


Figure III-24: This plot shows the growing number of observations of 20 fruit keywords during 171 days. The experiment started on July 10th, 2021.

There are some gaps between the data due to the device shut down. But the growing trend is pretty evident from the plot. I do not think this is a big issue since the increasing trend continues to be linear just by eyeballing the plot. The primary factor influencing the growing speed is the popularity of the keywords among users. To verify this, I fit a linear regression model for each keyword. The model can be represented as below (Equation 2):

$$count = \beta_0 + \beta_1 * days + \epsilon \quad (2)$$

Table 8 represents all the coefficients and their confidence interval of the linear models derived from 20 keywords. Interestingly, videos related to "banana" are the fastest-growing ones compared with other keywords.

<b>Model</b>	$\hat{\beta}_0$	$\hat{\beta}_1$	<b>Int* 2.5%</b>	<b>Int* 97.5%</b>
banana	917.19175	37.53795	36.90982	38.16607
apple	1427.07940	31.69988	31.08575	32.31401
orange	907.93584	26.54499	25.94461	27.14537
cherry	912.68646	24.45266	23.93543	24.9699
lemon	858.91651	19.46489	18.87549	20.0543
mango	883.11446	16.06217	15.52717	16.59717
lime	868.37688	14.96493	14.42307	15.50679
peach	591.49914	12.83702	12.46931	13.20473
strawberry	766.52166	11.96838	11.55233	12.38443
blackberry	700.305731	9.648449	9.295125	10.00177
pineapple	597.989318	8.709983	8.400616	9.019351
grape	492.179332	8.419943	8.201687	8.638198
blueberry	653.027572	7.458862	7.177792	7.739932
watermelon	632.469249	6.754389	6.465912	7.042865
pear	518.425738	5.663722	5.368122	5.959322
avocado	566.051250	5.451181	5.172003	5.730358
plum	627.700164	5.088705	4.841367	5.336044
cantaloupe	67.323801	0.180934	0.154478	0.207390
nectarine	29.280587	0.092247	0.076647	0.107847
respberry	4.000e+00	2.026e-17	-1.028e-18	4.155e-17

Table 8: Linear models fit with 20 different fruit keywords. Int\* represents Confidence Interval of  $\hat{\beta}_1$ .

The table shows the growing speed of getting videos associated with different keywords ranging from 0 to 38. However, if I discard the misspelling keyword "respberry" and the two unpopular keywords "cantaloupe" and "nectarine," at least five videos can be collected each day by my video detector. On average, I got approximately 14 videos daily for each of the keywords by using the top 20 fruits. If I increase my keywords base to 1000 words and assume 90 percent of them are popular

among the users, then by estimation, I could collect at least 135,000 live stream videos in 30 days. The video collector collected 525,312 videos in only eight runs within 20 days in the LST-YT1000 collecting experiment. The average number of videos I got for each keyword per day is roughly 26, five times higher than the minimum estimation and 1.86 times higher than the average estimation. The above results prove the high efficiency of my data mining pipeline.

### **III.F Data Profiling**

After the experiment, I got two types of TSCs datasets: LST-YF20 and LST-YT1000. These two datasets were collected from two different sets of keywords. In this section, I will first describe the nature of the two datasets. Later, I will compare the datasets with the existing ones in the literature. This section will generate insight into the datasets' distribution, categories, and characteristics.

#### **III.F.1 LST-YT1000 Dataset**

The LST-YT1000 dataset contains TSCs associated with 348,470 live streams published on YouTube, derived from 525,312 live stream events published between Sep 6th, 2011 and Dec 4th, 2021. Twenty-three percent of the total live streams are unavailable due to the removal of original posts or closure of live comments replay. Overall, our pipeline captured around 492 million live comments from 330,427 live streams on YouTube and detected 18,043 live streams without comments.

Figure III-25 shows the distribution of total TSCs count for each video in logarithm.

The average number of TSCs per video is approximately 1,490 comments.



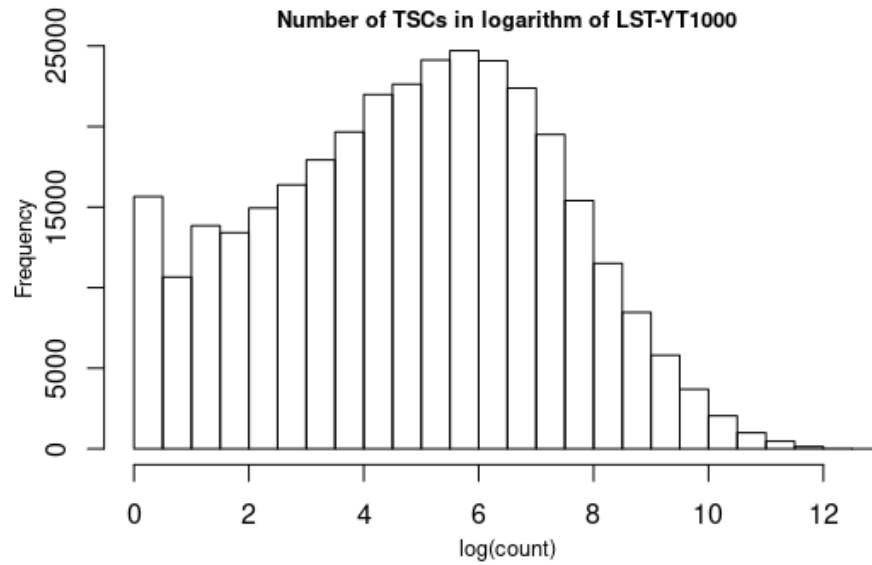


Figure III-25: This histogram shows the distribution of TSCs counts in the logarithm of the LST-YT1000. Total 492,331,808 comments are included.

Figure III-26 represents the distribution of number of authors for each video with valid TSCs. This plot includes all LST-YT1000 videos, with at least one author posting comments during the video replay. Each video contains an average of 290 authors and 1,450 comments (five posts per author). We estimate more than 80 million authors contribute to our dataset by commenting on the live streams, which include audiences and channel holders.

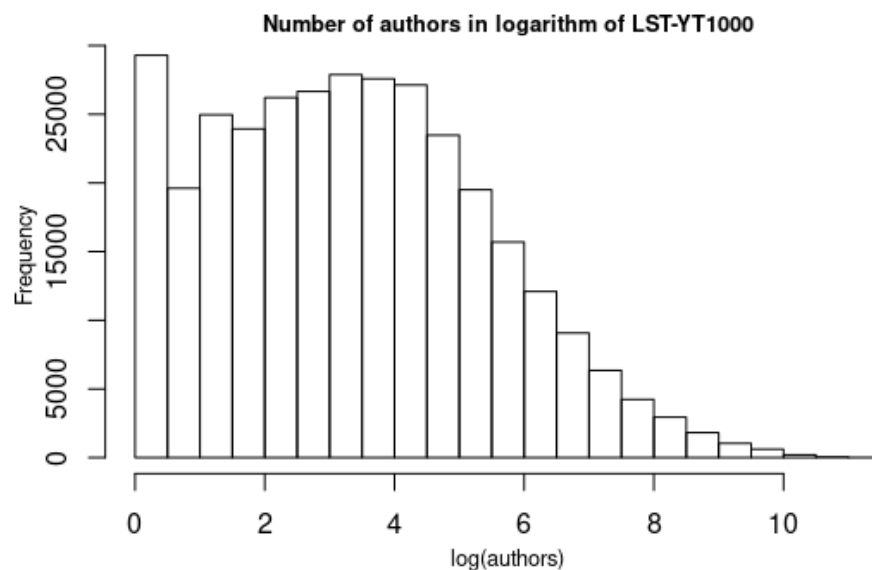


Figure III-26: This histogram shows the distribution of authors count in logarithm of LST-YT1000. Authors from 330,427 videos are included.

The total length of all videos in LST-YT1000 is unknown due to the lack of meta info collection. We estimate the entire video duration is around 105,626,097 hours by adding up the last recorded comment's timestamp for each video. However, after a close examination, we figured some moderator's comments could contain an inaccurate timestamp. We set up a 100,000 seconds threshold and calculated the entire video length as 664,252 hours for 329,229 videos left in the dataset. Figure III-29 contains the distribution of the logarithm of each video's duration in seconds. Most live stream videos posted on YouTube have playback time lasting 20 minutes to 2 hours. The average length of streams is 7,263 seconds. We also calculated the average intensity of TSCs for LST-YT1000 videos and plotted the result in Figure III-27.

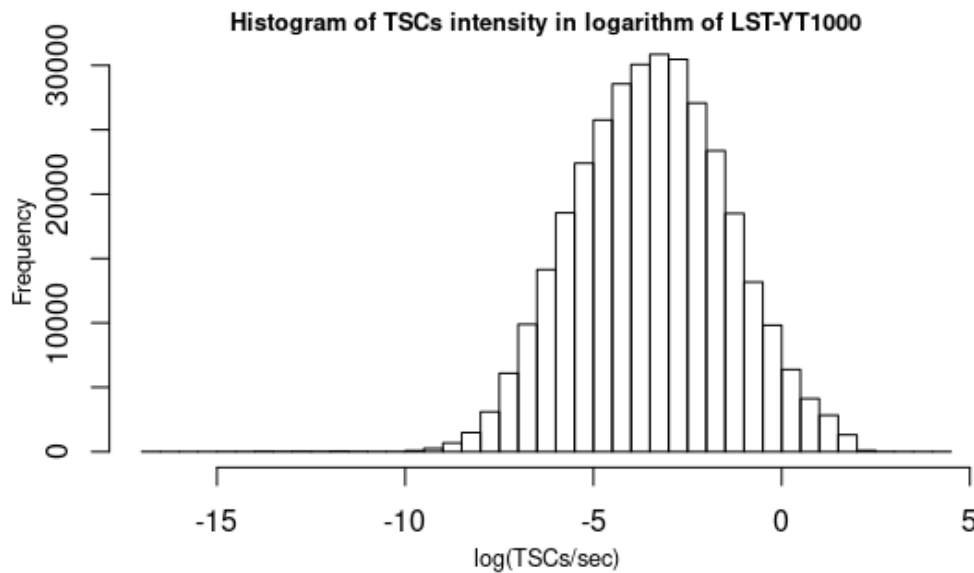


Figure III-27: This histogram shows the TSCs intensity distribution in the logarithm of the LST-YT1000 dataset.

### III.F.2 LST-YF20 Dataset

The LST-YF20 dataset was collected from 17,857 videos returned by the top 20 popular fruits search terms published on the YouTube platform between Mar 10th, 2018, to Aug 2nd, 2021. Meta information such as duration of videos, number of likes, number of dislikes, average rating, categories, number of subscribers, and languages are also collected. Overall, 12,437 clips contained live chat comments among the

above videos, and 10,448 clips' TSCs were downloadable through our pipeline.

The LST-YF20 provides 6,755,675 TSCs for 10,448 live videos, aggregating 18,088 hours of online streaming. Figure III-28 shows the actual duration of all TSCs videos, and Figure III-29 converts the original duration into logarithm. The mode of stream duration of the LST-YF20 is very close to the LST-YT1000, which appears to be around 90 minutes. Compared with the LST-YT1000 average video length, the LST-YF20 videos has a smaller duration of 5,896 seconds.

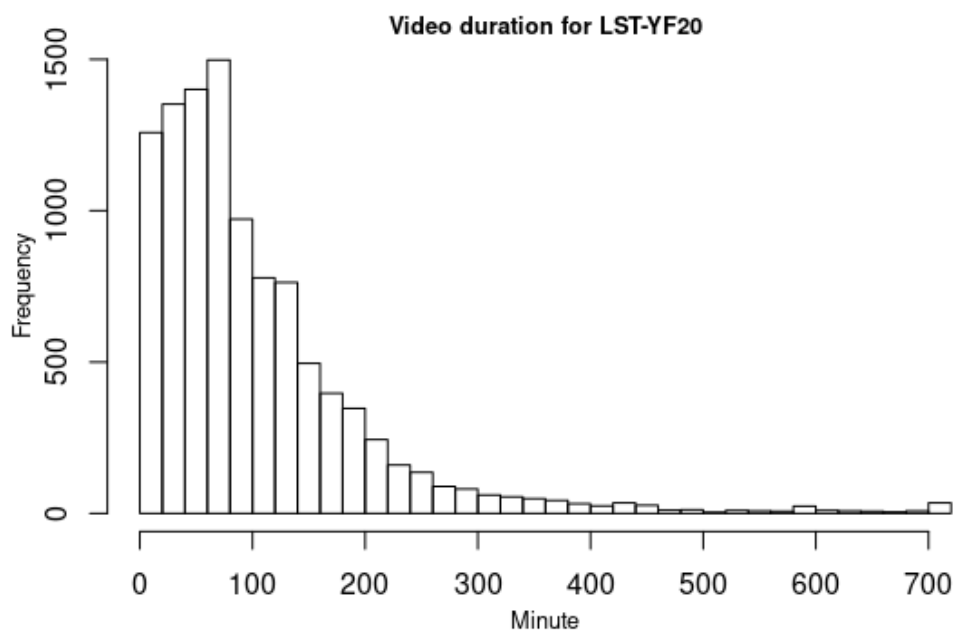


Figure III-28: This distribution of each video's approximate duration of LST-YF20. Total length of videos is 18,088 hours.

The distribution of authors and TSCs counts are shown in Figure III-30 and Figure III-31. On average, each video contains around 646 TSCs written by 117 authors. So, each author contributes to 5.5 TSCs, which is slightly higher than the LST-YT1000 dataset. However, the average video duration of the LST-YF20 is around 103 minutes, which is significantly less than the LST-YT1000.

As I created the LST-YF20 dataset, I also collected meta information of each video. I noticed there are 15 categories defined for the live streams. Figure III-32 shows how

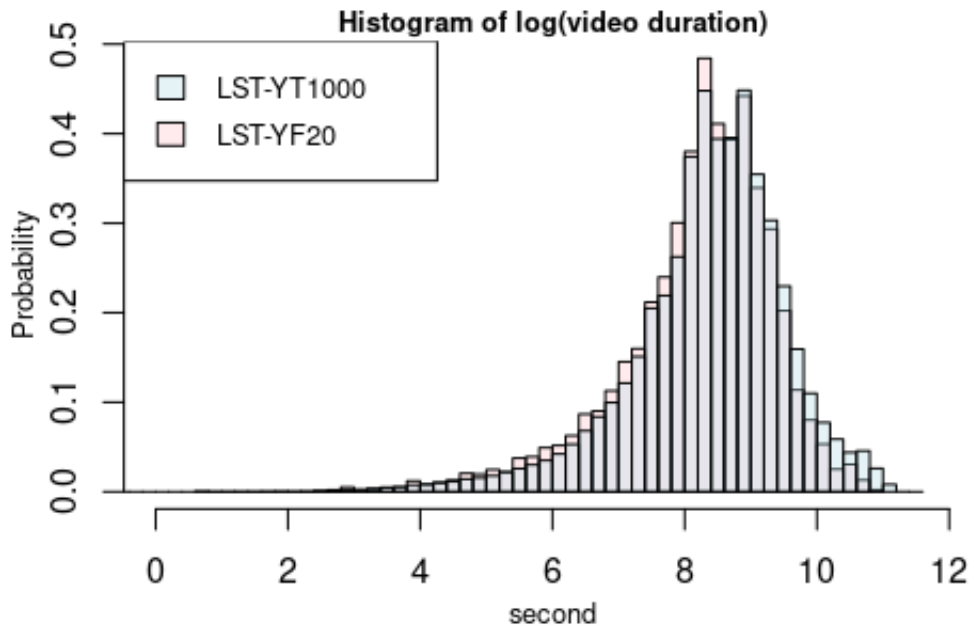


Figure III-29: This histogram shows the probability distribution of videos' duration in the logarithm of the LST-YF20 versus the LST-YT1000 dataset.

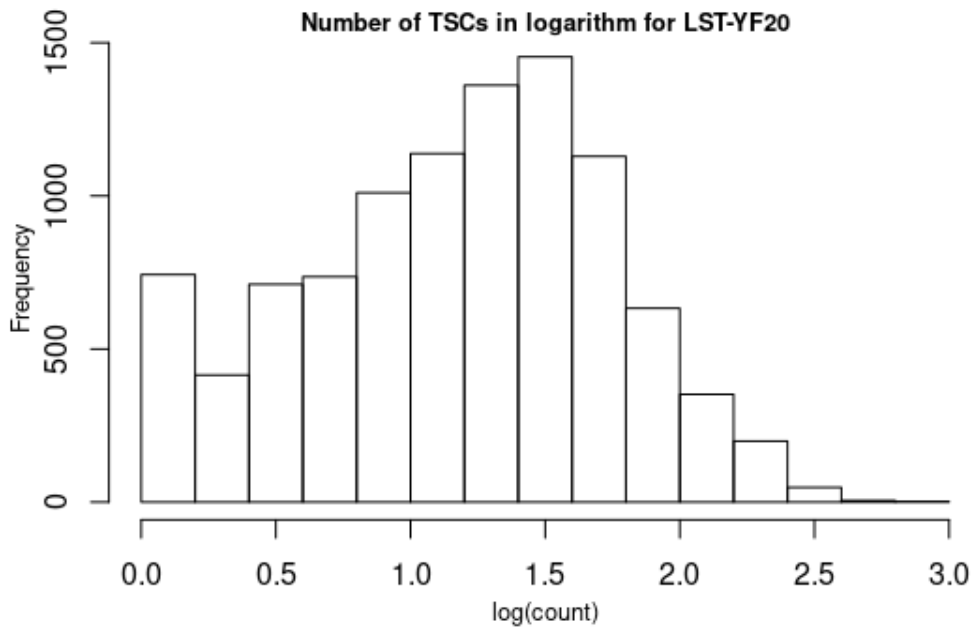


Figure III-30: The histogram shows the distribution of each video's TSCs count in the logarithm of LST-YF20. Total of 6,755,675 comments are included in the plot.

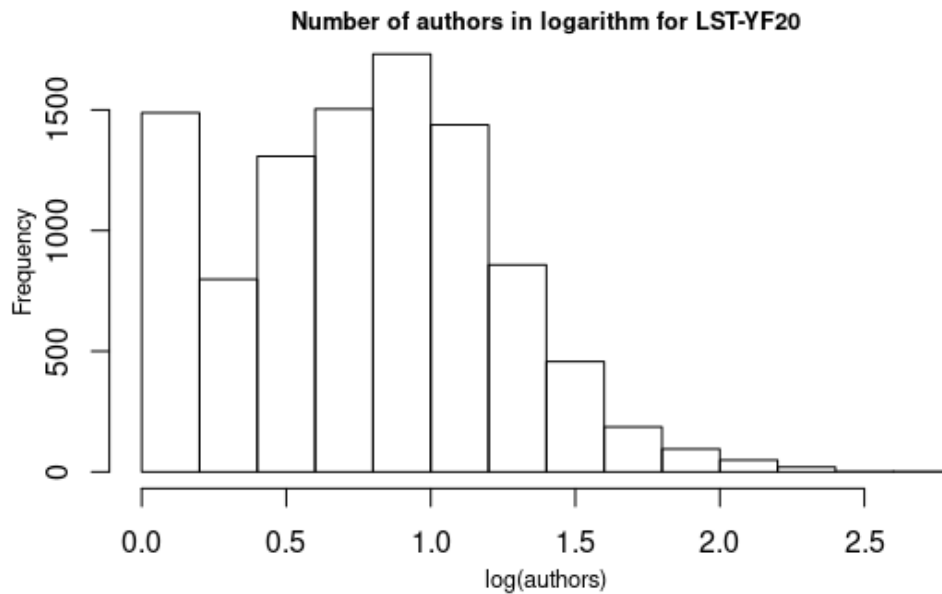


Figure III-31: This histogram shows the distribution of each video's authors count in the logarithm of LST-YF20. 10,448 videos with valid authors are included in the plot.

videos spread across categories in descending order. This plot provides a good insight into what type of videos I've collected from YouTube. The top two categories are "Entertainment" and "Gaming," which contributed to 29.8 percent and 28.1 percent of my dataset.

I calculate the average density of TSCs for each category in LST-YF20 and present the result in Figure III-33. The plot was generated by averaging the number of TSCs appear in each video divided by the unique authors. The top category is "Film & Animation," which contains 9.8 TSCs per second on average video. However, the "News & Politics" video category has the least TSCs density of 3.58 per second.

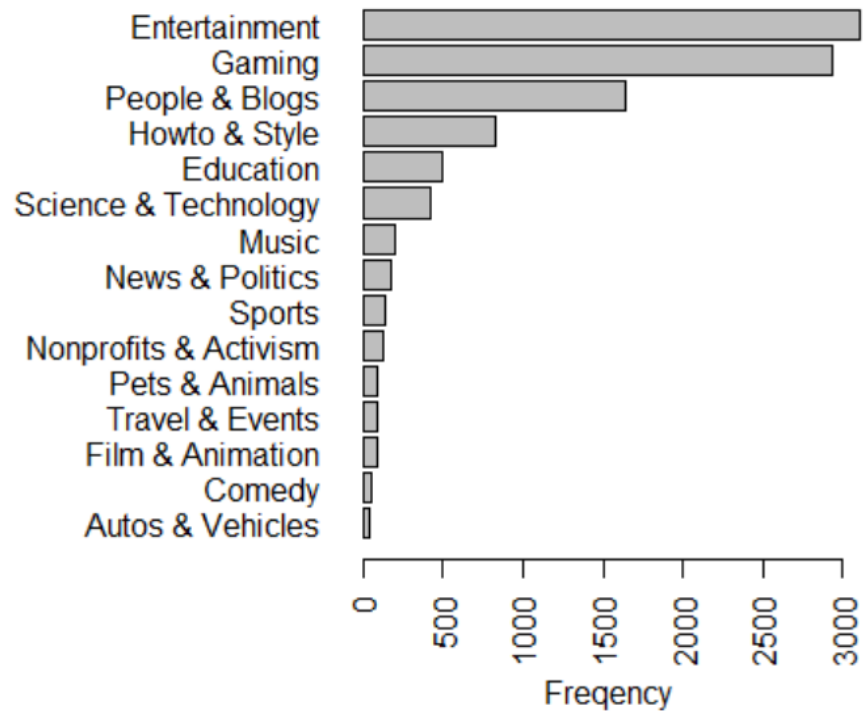


Figure III-32: This plot shows the number of 15 different categories of streams in the LST-YF20.

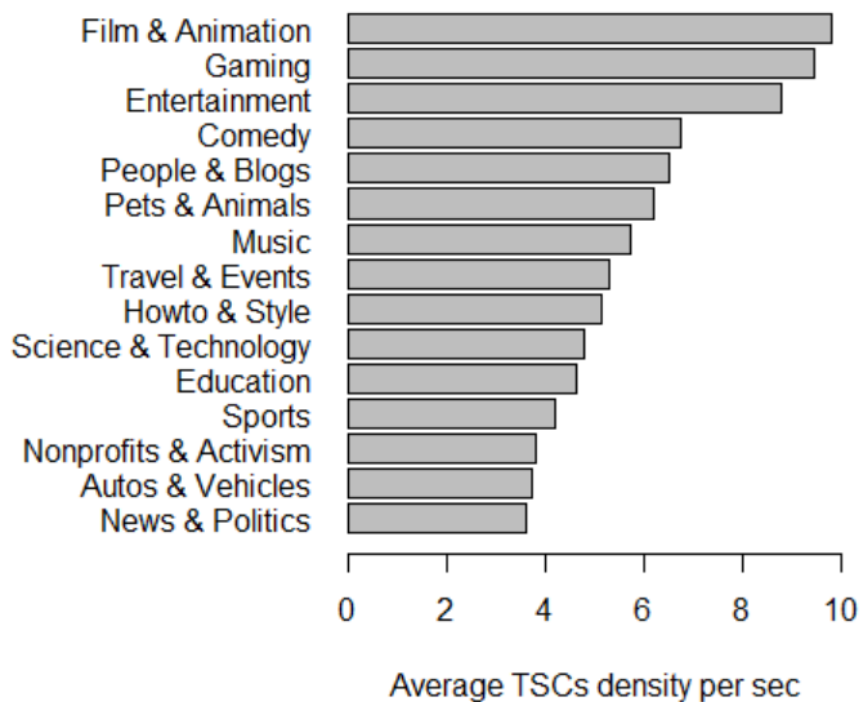


Figure III-33: This plot shows the TSCs intensity of different video categories of the LST-YF20.

Then I present the distribution of unique authors for each category in Figure III-34. To my surprise, among the top three categories, "News & Politics" videos achieve 162 unique authors while "Science & Technology" achieves 227 individual authors per video. Meanwhile, the "Film & Animation" category only has 41 unique authors in contrast.

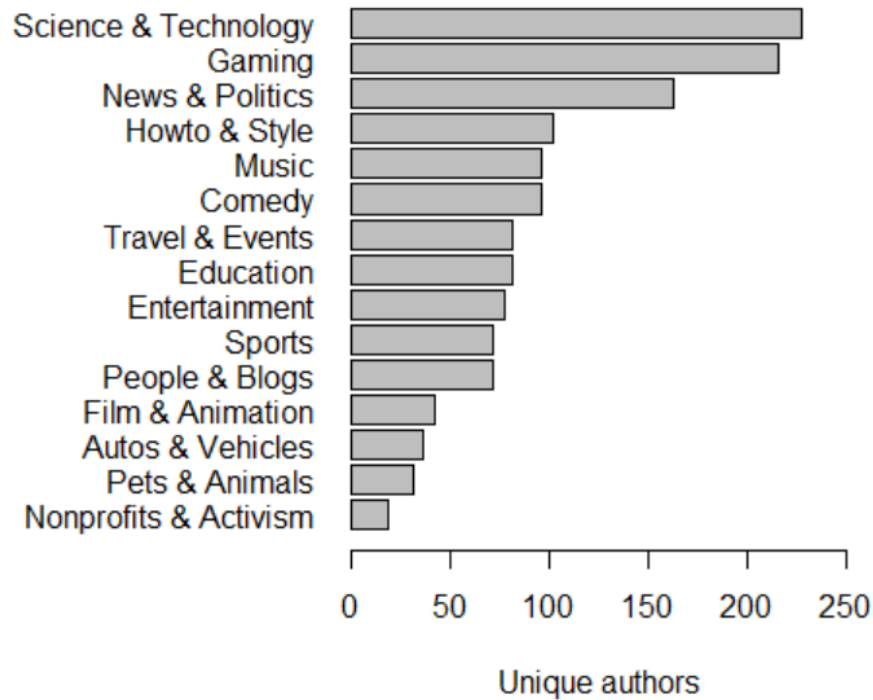


Figure III-34: This plot shows the number of unique authors within 15 different categories of the LST-YF20.

### III.F.3 Comparison with Existing Datasets

Compared with previous TSCs datasets in this research field, my datasets are significantly larger from comments and video duration perspectives. It also contains more variations of videos which include both virtual and real YouTubers. Table 9 compares the most recent available large TSCs dataset in the research field to my TSCs dataset.

Name	Video	Length	TSCs	Type	Source
BL-906 (Liao et al., 2018)	17,870	47.8kh	33m	Anime	Bili
Livebot (Ma et al., 2019)	2361	114h	896k	General	Bili
TSC D1 (Yang, Wang, et al., 2019)	287	50.25h	228k	General	Bili,AF
TSC D2 (Yang, Wang, et al., 2019)	180	67.3h	570k	Anime	Bili,AF
BJUTSD-V2 (L. Wang et al., 2020)	3682	14.2h	120k	Adult	IntNet
VideoIC (W. Wang et al., 2020)	4951	9.3h	5.3m	General	Bili
VTuber 500M (Uechi, 2021)	-	-	500m	Virtual	YT
<b>LST-YF20</b>	<b>10,509</b>	<b>18kh</b>	<b>6.8m</b>	<b>Fruit</b>	<b>YT</b>
<b>LST-YT1000</b>	<b>330,427</b>	<b>66.4kh</b>	<b>105m</b>	<b>General</b>	<b>YT</b>

Table 9: My dataset compared with other large TSCs datasets in the field.

\*m is a abbreviation for million. \*k is a abbreviation for thousand. \*Bili represents Bilibili, a popular video platform from China. \*AF represents AcFun, a popular video platform from China. \*YT represents YouTube. A popular video platform developed by Google. \*IntNet represents for Internet. \*h means hours.

My current approach provides two unique datasets compared with existing ones. The main advantage of my datasets can be summarized as:

1. My datasets contain general video content directly from an English-speaking multimedia platform.
2. I provide the top 1000 popular search terms from Google associated with each video.
3. I collected accurate dislike counts and other meta information within LST-YF20. These make the datasets unique of their kind.
4. My datasets are significantly larger than other datasets in the research field with similar content.



## **IV Sentiment Modeling**

I collected two large-scale TSCs datasets through my custom-designed data pipeline between 2021 and 2022. Both datasets contain high-quality TSCs information associated with hundreds of thousands of online streaming on the YouTube platform. To demonstrate the possibility of using such TSCs data to detect and predict the viewers' response. I will describe the sentiment modeling experiments I conducted in this section.

There are a few obstacles I need to mention before the modeling. First, as all TSCs data are collected directly online, they are unstructured and unsuitable for the modeling. Therefore, a deep data cleaning should be done before using the TSCs dataset. Second, I'm missing important information from the origin video sources. The videos and their auxiliary data are necessary for the experiment. Third, a proper model needs to be selected to fit the purpose of the investigation. Researchers use various machine learning algorithms in this field, and setting an appropriate framework is essential.

These obstacles add a significant amount of time to solve beyond the data collection and therefore limit the scale of my modeling experiment. However, I tried my best and was able to verify my hypothesis through the experiment. This section will state the problem I'm trying to address and introduce the data process and experimenting details I've developed.

### **IV.A Problem Statement**

Online streams consist of two types of information: First, the information distributed by the uploader; Second, the response from viewers. After viewing the video contents, the TSCs data logs each frame's comments posted by multiple audiences. The datasets I created provide a deep insight into the natural response of humans. By examining the TSCs data, I could explore the links between the original video information with the human response and potentially predict the viewer's behaviors or vice versa. The problem can then be viewed as a regression fitting problem: I am fitting a natural response sentiment score of humans with the sentiment values derived from the video.

This experiment is built on three simple assumptions:

1. The TSCs can represent the human nature sentiment response.
2. The actual sentiments can be extracted from the live streamings.
3. There are correlations between the original video sentiment with the viewer's sentiment response.

Because one of the goals of this study is to understand how accurate and efficient the proposed sentiment labeling framework is, knowing the actual sentiment values of the original videos is crucial for making comparisons. Previous studies only focused on using an overall sentiment evaluation, such as reviews or like/dislike, instead of a fine-grained sentiment annotation. I need to develop a data pipeline to gather necessary auxiliary data before converting them into fine-grained sentiment values. The workflow comprises three vital steps: auxiliary data identification, auxiliary data collection, and sentiment conversion (Figure IV-35).



Figure IV-35: The graph shows the workflow of creating a fine-grained sentiment labeling from auxiliary information.

## IV.B Auxiliary Data

To verify the accuracy of TSCs' converted sentiment data, I need to compare them with the original sentiment values carried by the videos. Since no actual sentiment values were measured or logged by the online multimedia platforms, some auxiliary data are needed to estimate the original sentiments. There are three ways of evaluating the actual sentiment values: using visual channels, using audio channels, and mixing visual signals with audio tracks.

The accuracy of the three methods depends on the contents of each video. For

instance, musical recordings such as concerts or albums could carry sentiment stimulus through the audio signal, while movies or animations could arouse more sentiment to audiences through graphics changes. In theory, mixing with both visual and audio signals could provide the highest accuracy for sentiment values since they carry the complete information of the original videos. However, taking graphic information into account could be computationally expensive and time-consuming. I decided only to use the audio data in this research.

The semantic information carried by the audio channel plays an essential role in audiences' sentiment response. However, direct analysis of the acoustic track could only provide basic, non-semantic features such as pitch, rhythm, power, or word pronunciation. These low-level acoustic features can indeed influence the audiences' perceptual impressions to some degree occasionally. They sometimes may become significant and influential in live music shows or concerts. Still, most online videos consist of dense dialogues and speeches. For example, a piece of light background music in a weather forecast TV show is helpful to relax the viewer's mind but could not distract their attention if tomorrow's snowstorm is coming.

Therefore analyzing the semantic meaning of the acoustic tracks is essential. Modern Automatic Speech Recognition (ASR) techniques can recognize speeches directly from the audio inputs and convert them into a human-readable text form transcript. Applying Natural Language Processing (NLP) on the text-based transcript afterward makes it possible to examine the hidden semantic features from audio channels. Figure IV-36 shows the general process of how to encode an input audio signal into a semantic representation.

#### **IV.C Speech Recognition**

An ASR model usually consists of three major components: an Acoustic Model (AM), a Pronunciation Model (PM), and a Language Model (LM) (Pundak and Sainath, 2016). The acoustic model works directly with raw audio waveforms at the subword level, mapping small audio pieces into specific phonetic units. The phonetic units,

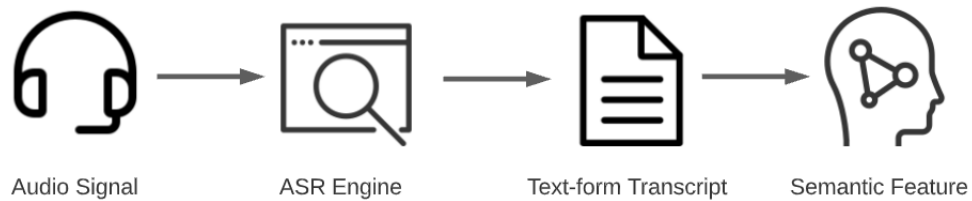


Figure IV-36: The graph shows the overall process of converting semantic features from audio inputs. The audio signal was first transformed into a text-based transcript by the ASR engine and then examined by NLP algorithms to obtain a precise semantic representation.

which are indivisible units of speech sound such as the  $\backslash k \backslash$  of 'cool' or the  $\backslash b \backslash$  of 'bat' in English. The pronunciation model then predicts the boundary of a word by identifying the correct start and stop phonemes based on prior acoustic knowledge of words. Finally, the language model then provides translation between the acoustic and linguistic representation based on the construction of the phonetic sequence of the word. Figure IV-37 shows the complete process of converting an audio input into a sequence of words.

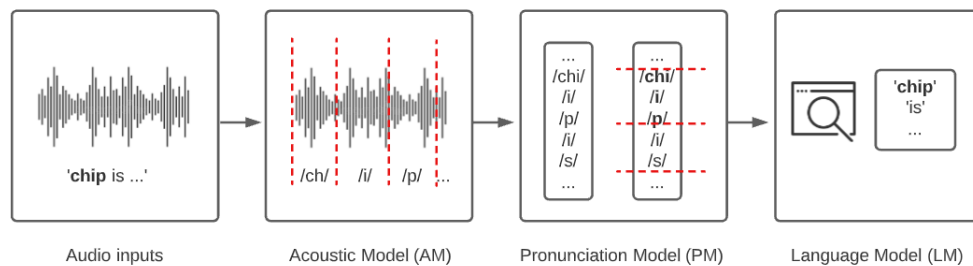


Figure IV-37: The graph shows there are three internal models for an ASR system. The AM model recognizes phoneme units. The PM model identifies the correct boundaries of a word. The LM model converts the sequence of phonemes into a semantic representation.

Training an ASR model can be difficult, and it is still an active research field for many years. Many factors may vary the performance of a given ASR model, which is speech recognition accuracy. Briefly speaking, the quality of the audio signal (for example, background noise level), the variation of pronunciation (for example, different accent

and dialect), and the capacity of the ASR model (for example, limited vocabulary) are the three major factors that significantly contribute to the final recognition result (Svendsen, 2004). To overcome this, ASR models are usually trained on different datasets (Pundak and Sainath, 2016).

#### IV.D Offline ASR Experiment

There are many pre-trained offline ASR engines already available to the public. The most popular open-source ASR engines are provided by Kaldi (“Kaldi”, 2011), Wav2letter (“facebook research wav2letter”, 2021), NeMo (“NVIDIA/NeMo”, 2021) and DeepSpeech (“mozilla/DeepSpeech”, 2021).

Vosk is an open-source offline multi-language toolkit, which runs ASR engines on acoustic data (“VOSK Offline Speech Recognition API”, 2020). It is lightweight and could support 17 languages and dialects. Alphacephei provides a list of compatible ASR engines for English recognition (Table 10). It is also possible to train a custom ASR model through the Kaldi ASR toolkit (Povey et al., 2011).

ASR Engine	Size	WER	Speed	Model
vosk-model-en-us-aspire-0.2	1.4G	13.64	12.89	Fisher, LM model
vosk-model-small-en-us-0.15	40M	9.85	10.38	Wideband GMM model
vosk-model-en-us-daanzu-20200905	1.0G	7.08	8.25	Wideband GMM model
vosk-model-en-us-daanzu-20200905-lgraph	129M	8.2	9.28	Wideband GMM model
vosk-model-en-us-librispeech-0.2	845M	TBD	TBD	Kaldi, Librispeech
vosk-model-small-en-us-zamia-0.5	49M	11.55	12.64	Zamia f_250, Kaldi

Table 10: Compatible models officially provided for the Vosk toolkit. Models are tested on TED-LIUM dataset (Korvas et al., 2014). WER (Word Error Rate).

The quality of recognition depends on audio quality and depends on the ASR engine’s accuracy and training data set. Table 11 shows a comparison of Vosk models test on four different datasets. From the table, daanzu’s Kaldi model holds overall good

performance on all datasets.

<b>ASR Engine</b>	<b>Librispeech</b>	<b>Tedlium</b>	<b>Commands</b>	<b>Fisher eval2000</b>
en-us-aspire	13.49	12.53	55.62	17.39
en-us-daanzu	8.36	8.68	9.30	31.37
en-us-small	15.34	12.09	45.52	N/A
en-us-librispeech	4.37	N/A	N/A	N/A
deepspeech	6.12	18.03	N/A	N/A

Table 11: Vosk models' (Korvas et al., 2014) WER (Word Error Rate) on four datasets: Librispeech, Tedlium, Google Commands and Fisher eval2000.

The output of Vosk API is stored in JSON format. Table 12 represents an example of words recognized by the ASR engine. The starting/ending time and confidence for each term are also listed.

<b>Word</b>	<b>Start</b>	<b>End</b>	<b>Confidence</b>
it's	589.47	589.60	0.369431
seems	589.60	589.86	0.327919
see	590.46	590.73	0.321319
well	590.91	591.18	1.000000

Table 12: An example output of a list of words from Vosk API.

However, this approach faced two significant obstacles. First, downloading hundreds of thousands of videos adds pressure to the local storage and networking. Considering the size of my data, crawling all audio channels out of the YouTube platform could disturb the network traffic and might not meet their End User Licence Agreement (EULA). Second, processing all audio data offline can be time-consuming and expensive. Performing speech recognition is computationally intensive and requires powerful machines to speed up. This method could not meet the condition of my low-budget experiment environment.

#### **IV.E Online ASR Data Collection**

Online ASR methods are the way to solve the difficulty offline ASR methods have. There are several advantages of using the online ASR method. First, the speech recognition is done on the server side, so local hardware requirements are relatively low compared to offline ASR methods. Second, online ASR engines are more accurate compared with the offline version. Most ASR service providers are eager to rapidly upgrade and maintain their engines to be competitive in the market, which means customers always get the latest version of ASR products. Third, some ASR models are implemented close to the data source. Less data flow moves during the speech recognition task, saving time and reducing expense.

After searching, I figured YouTube provides an automatic captions function for some videos. As Google is one of the major players in the speech translating field, I think maybe using the captions automatically generated by Google is a good option for representing the result of online ASR methods. Leveraging auto-generated subtitles can relax local computation resources, reduce complexity, and speed up data processing. YouTube does provide an option for downloading subtitles through Data API (“Captions YouTube Data API”, 2021). However, due to the budget limitation, I need to find an alternative approach.

The first step is to check the data structure of automatic captions. Tampermonkey, a popular userscript manager for the browser, makes it possible to manually download a small batch of auto-generated subtitles (“Tampermonkey for Chrome”, 2021). Tampermonkey is a browser extension allowing users to import third-party scripts to modify web page behavior through JavaScript. It works as an ad-hoc program alternating the activities the browser performs on a specific web page.

Tim Smart developed a custom script that allows Tampermonkey users to manually download the YouTube subtitles in 2009 (“Download YouTube Captions for Greasemonkey”, 2014, “Tim-smart Userscripts”, 2021). Figure IV-38 shows a green drop-down menu appears below a YouTube video which allows users to download the

auto-generated subtitle created by YouTube (Figure IV-39). The information contained within the auto-generated subtitle is summarized in Table 13, where vocal speech is converted into text and annotated with a timestamp. Both the starting and ending of a transcript line are timestamped.



Figure IV-38: A YouTube subtitle download options show up on the web page via Tim Smart's script.

Source by: <https://www.youtube.com/watch?v=EjjWi5PizXw>

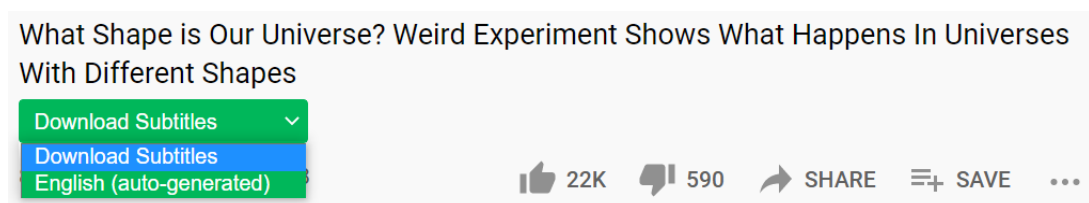


Figure IV-39: Auto-generated English subtitle are available for download through Tim Smart's script.

Source by: <https://www.youtube.com/watch?v=EjjWi5PizXw>

After verifying that the automatic captions meet the project requirements, I wrote a Python script that wraps the YouTubeTranscriptApi library to extract the online auto-generated captions automatically. The script loops over the existing videos' id I collected from the previous crawled TSC dataset and downloads associated transcripts.



<b>ID</b>	<b>Start</b>	<b>End</b>	<b>Text</b>
1	00:00:00,000	00:00:03,840	hey there this is Josh welcome back to
2	00:00:01,709	00:00:05,490	let's game it out this is shopkeep 2 or
3	00:00:03,840	00:00:07,470	as I like to call it shop Ikki P 2
4	00:00:05,490	00:00:09,030	really hoping this game lets us torment

Table 13: Auto-generated subtitle by Tim Smart's script through Tampermonkey.

I also record the corresponding timestamps, duration, and messages for the auto-generated captions and store them locally on my hard disk.

I got 104,326 auto-generated captions out of 348,469 videos from the LST-YT1000 dataset and 4,646 auto-generated captions out of 10,509 videos from the LST-YF20 dataset. Videos without ASR captions results or in non-English format are discarded. Overall, the coverage of captions in LST-YT1000 is 29.9 percent, and caption coverage for LST-YF20 is 44.2 percent. Although the average coverage is less than 50 percent, the total number of speech-recognized live streams is enough for this project.

Then I analyzed the file size of the converted ASR text. The distribution of all transcript's file size of both LST-YT1000 and LST-YF20 are shown in Figure IV-40 and Figure IV-41. The distribution is left-skewed for both datasets, showing that most transcripts are less than 100 kilobytes.

#### **IV.F Data Processing**

Before data analysis, I need to perform a series of data operations to structuralize the dataset. The raw TSCs data crawled from the web is in free form and contains a lot of noise. For example, some entries are missing or contain special characters, which could break the following data analysis. The dataset has to be cleaned before being modeled. In this section, I will perform a series of data processing techniques, including reducing the complete dataset for a smaller one, fixing the gaps between labels, resolving the overlapping issue, cleaning the data for unwanted characters, and interpolating the data into a proper time sequence.

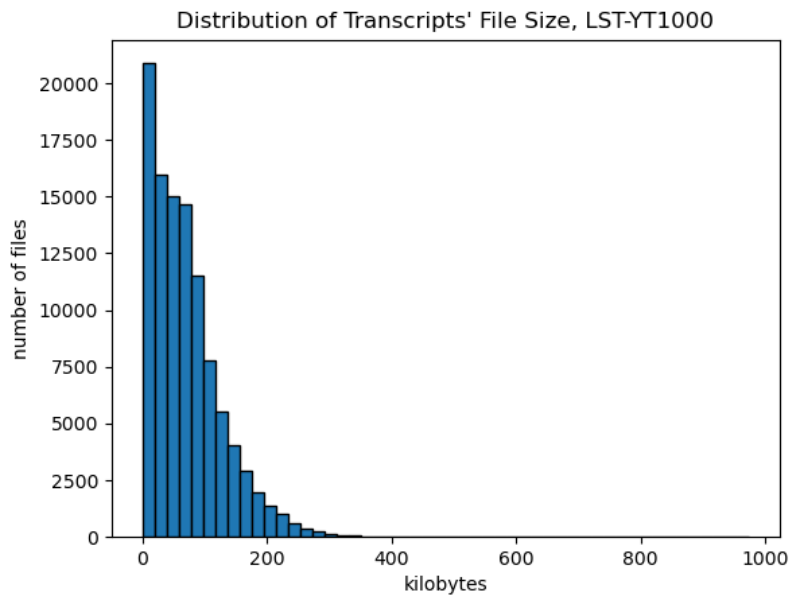


Figure IV-40: The histogram of the LST-YT1000 transcript's file size.

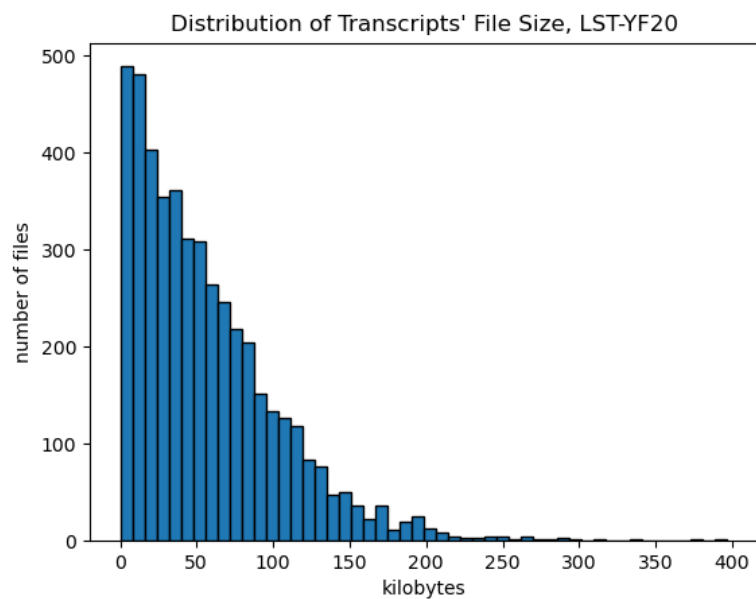


Figure IV-41: The histogram of LST-YF20 transcript's file size.

#### IV.F.1 Data Subsetting

As a result, I got two large datasets from my previous data collection experiment. The LST-YF20 dataset contains 17,857 video ids, and 10,448 videos have live chat information. The LST-YT1000 dataset consists of 330,427 videos with TSCs

downloaded. These videos cover a variety of contents and therefore have a different number of TSCs and duration. By dividing the number of comments by the length of each stream, I can calculate the intensity of comments (number of comments per second) for all videos (Equation 3). Meanwhile, by splitting each TSCs observation into a sequence of words, I can also calculate the intensity of words for each video (Equation 4).

$$TSC\_Intensity = \frac{Number\_of\_Comments}{Video\_Duration} \quad (3)$$

$$Word\_Intensity = \frac{Number\_of\_Words}{Video\_Duration} \quad (4)$$

However, splitting all TSCs is not practical since the LST-YT1000 dataset contains over one hundred million comments. The estimation of word intensity can be derived from the approximate number of terms within a TSC (Equation 5).

$$Est\_Word\_Intensity = \frac{Number\_of\_Comments}{Video\_Duration} \times Est\_Words\_per\_TSC \quad (5)$$

Table 14 includes the summary statistics of all ASR transcripts for both LST-YF20 and LST-YT1000 datasets.

	<b>LST-YF20</b>	<b>LST-YT1000</b>
<b>ASR Count</b>	4,646	104,325
<b>ASR Coverage (%)</b>	44.2	29.9
<b>Avg Video Duration (sec)</b>	4,535.46	5,219.06
<b>Avg Words Count (words)</b>	7,816.37	9,606.72
<b>Words Intensity (words/sec)</b>	1.80	2.06

Table 14: This table shows the auto-generated captions' summary statistics for both LST-YF20 and LST-YT1000.

I also calculated each video's average transcript length, density, and video duration with the equation mentioned above. The detailed distribution of video length, number of words, and words' intensity within each video are plotted in Figure IV-42, Figure IV-43, Figure IV-44, Figure IV-45, Figure IV-46 and Figure IV-47.

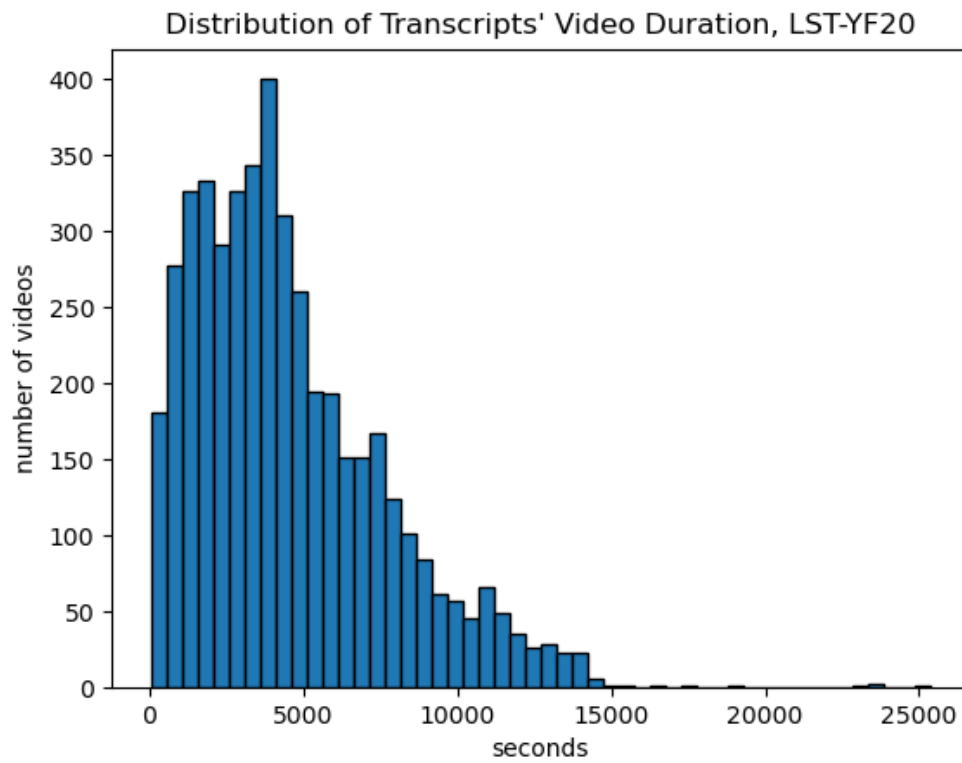


Figure IV-42: The histogram of the LST-YF20 transcript's video duration.

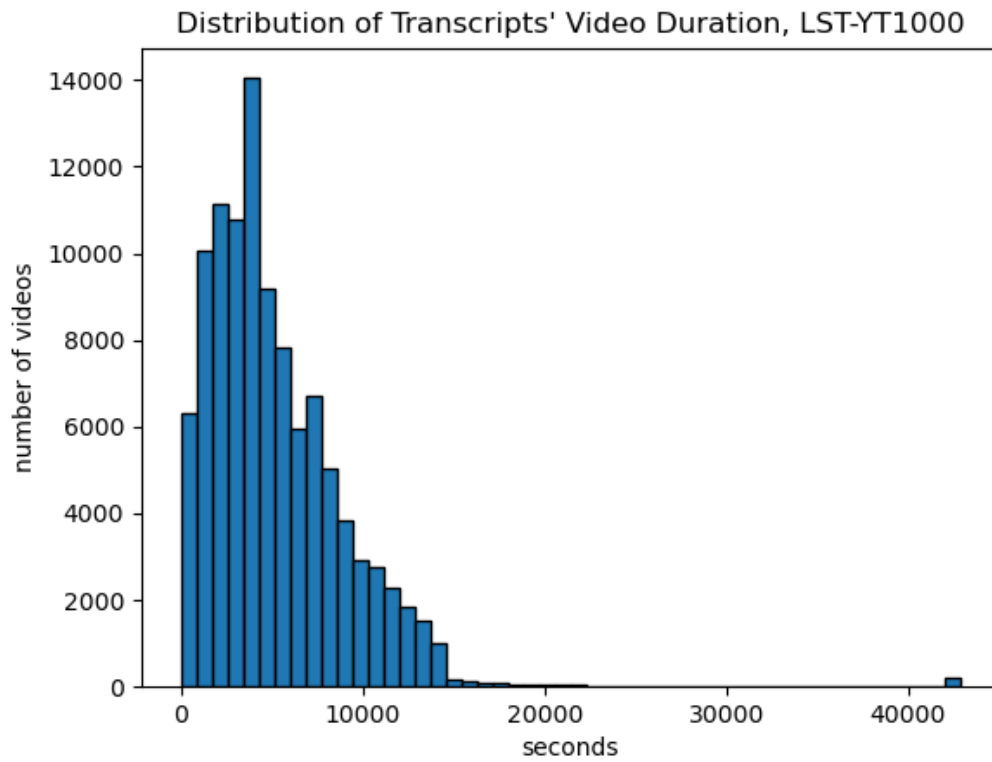


Figure IV-43: The histogram of the LST-YT1000 transcript's video duration.

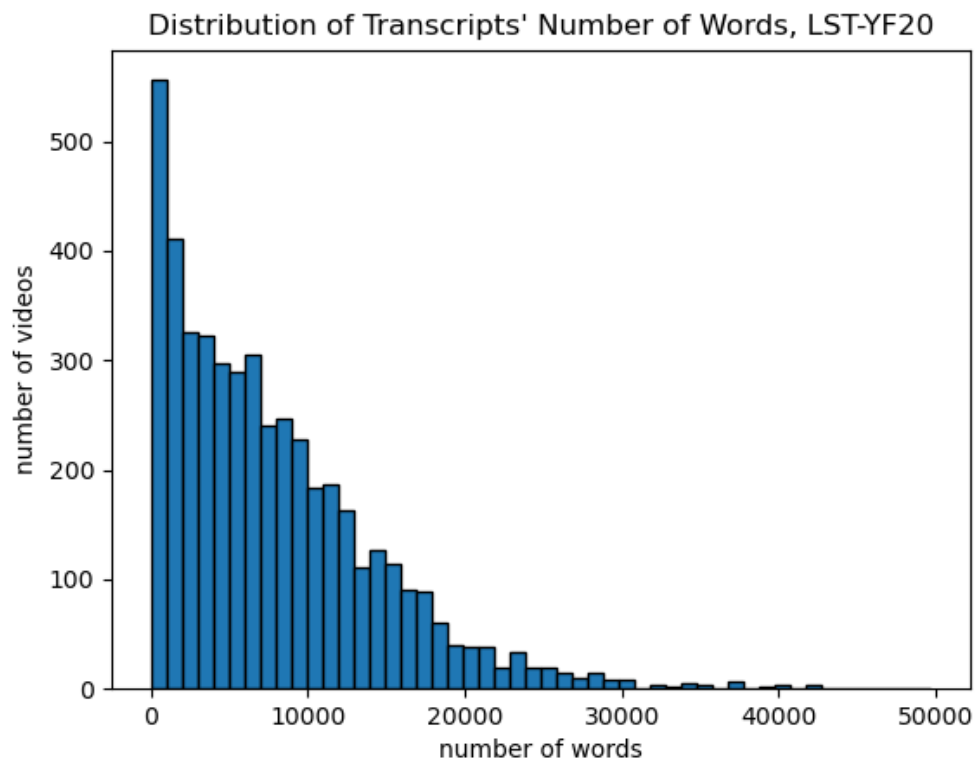


Figure IV-44: The histogram of the LST-YF20 transcript's words count.

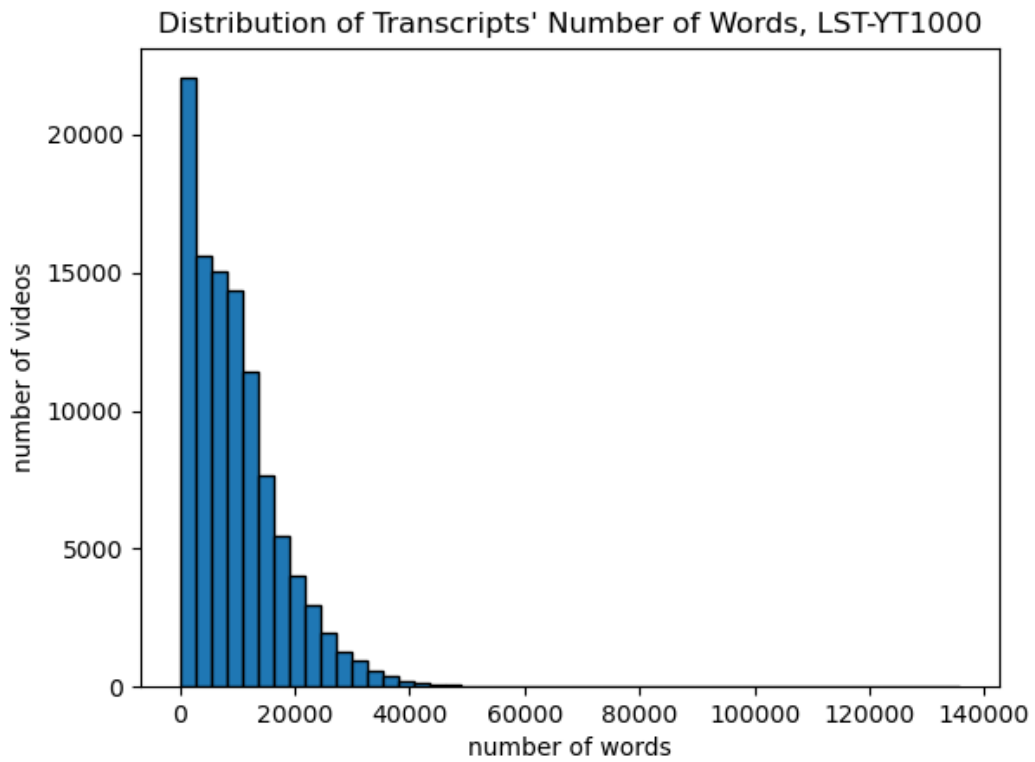


Figure IV-45: The histogram of the LST-YT1000 transcript's words count.

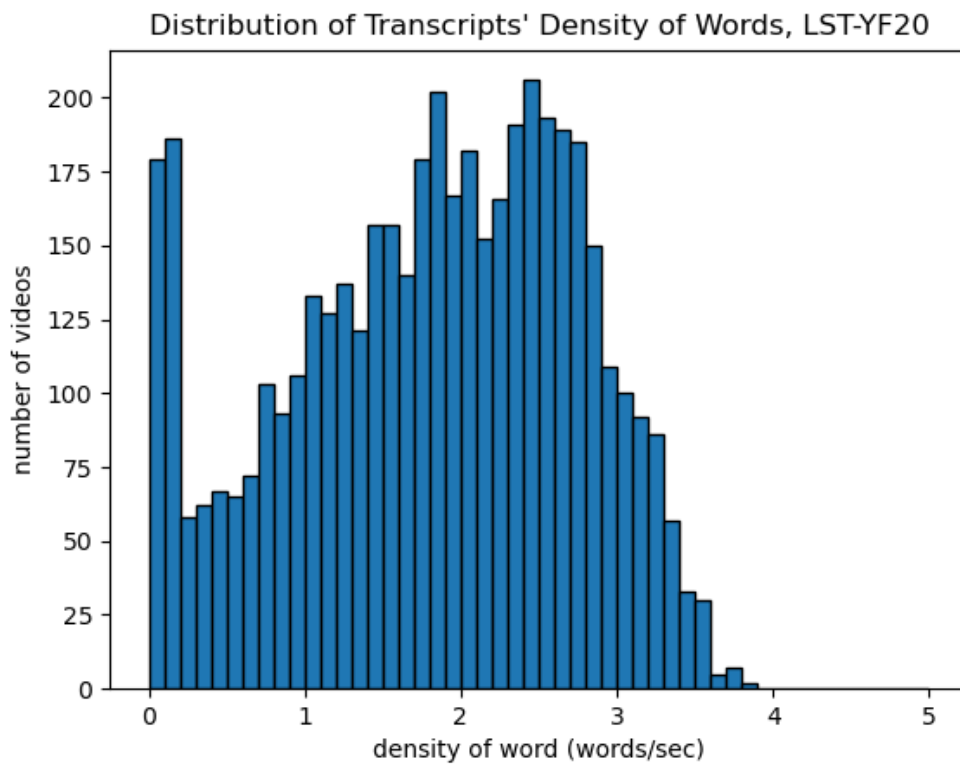


Figure IV-46: The histogram of transcript's words per second of the LST-YF20.

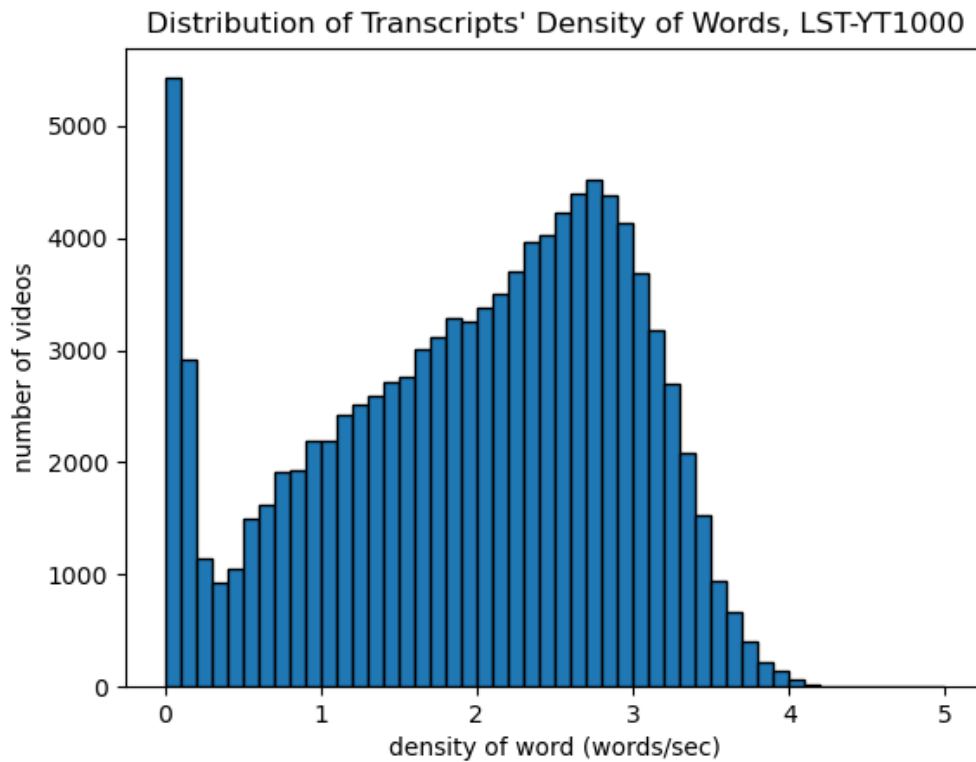


Figure IV-47: The histogram of transcript's words per second of the LST-YT1000.

The words' density distribution plots show the peak of words' density for both datasets is around 2.8 words per second. The result suggests the majority of videos have acceptable ASR results. I noticed some videos are clustered on the very left of the histogram, meaning they either fail on the speech recognition phrase or contain little audial semantic information. Moreover, due to the large data size, I decided to filter out low-quality data and sample a small portion of it from the LST-YT1000 dataset for further analysis.

The first sampling step is to combine TSCs data and ASR data. The combination step was done by aligning the identical video ids in a data frame. It allows me to apply more restrictive rules to filter my dataset. I applied the following rules to all videos in the LST-YT1000 dataset:

1. The TSCs intensity should be valid.
2. The TSCs intensity must be greater than one comment per second.

3. The ASR intensity must be between 2.8 to 4 words per second.
4. The duration of the live streams has to be 1 to 2 hours.
5. The videos must contain over 5000 TSCs but less than 15000 TSCs.

The above filtering steps are applied in Python to restrict the data size and control the videos' quality. After each step, portions of data were cropped and distilled. Table 15 shows the number of videos left after each operation. Finally, after filtering, I got 167 videos for future data analytics.

<b>Operation</b>	<b>Number of videos</b>
<b>Total</b>	104,325
<b>Remove NAs</b>	99,647
<b>Filter TSCs intensity</b>	3,457
<b>Filter ASR intensity</b>	829
<b>Filter duration</b>	249
<b>Filter TSCs counts</b>	167

Table 15: This table shows the number of videos left after each filtering operation for all LST-YT1000 live streams.

#### **IV.F.2 Online ASR Data Pre-processing**

The raw ASR-generated text is recognized loosely around specific timestamps. Three attributes define each piece of the transcript: text, start timestamp, and duration. There are several issues with this type of data structure. Some data cleaning should be done before using the data in sentiment analysis.

First, limited by the accuracy of the ASR engine, transcript pieces may overlap with each other. The starting timestamp is relatively accurate for each part of the sentence, but after calculating the duration of the ending timestamp, some transcript pieces are conflicted in the time domain. For example, in Figure IV-48, fragments of sentences were recognized individually by the ASR engine, and the second fragment of "fantastic show for your guys ..." was identified before the first fragment of "ladies and gentlemen ..." ends.

Second, many mini gaps occur between sentences. The recognizable speeches do not



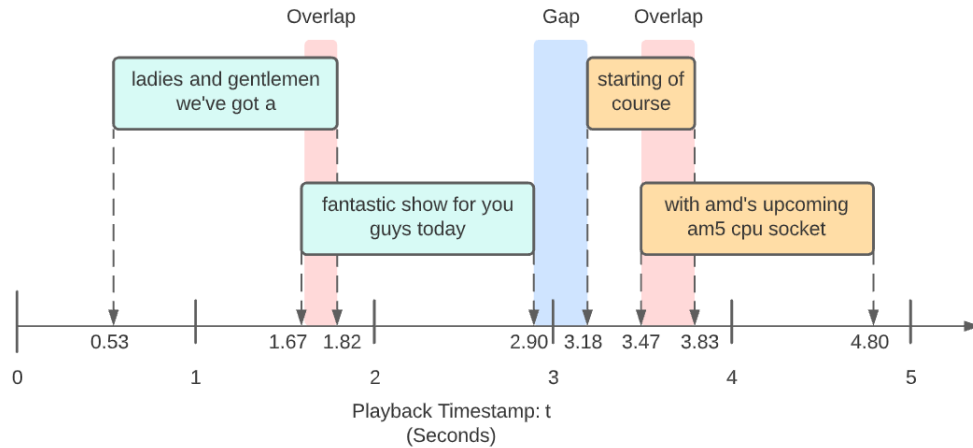


Figure IV-48: Transcripts generated by Automatic Speech Recognition (ASR) models may overlap with each other (pink area) or contain gaps (blue area).

continuously happen over each timestamp. Other acoustic activities such as playing music or background noise may exist between the talks or even overtake the speaker's voice. For example, the blue segmentation range from 2.9 sec and 3.18 sec is a gap between two sentences.

Third, the majority of the timestamp converted by the online ASR engine are not integers. A proper time series of sentiment requires each timestamp to be a piece of text with semantic meanings. A segmentation or grouping algorithm must be applied to the continuous text into individual semantic labels.

I applied a timestamp alignment algorithm to the raw ASR textual data to address the temporal out-of-sync issue. The algorithm can be split into three parts. First, the overlapping ASR texts are trimmed and connected with the following ones.

Figure IV-49 shows the process of creating a completed sentence by concatenating two overlapped text pieces. Second, the starting timestamp of each sentence was rounded into the closet integer seconds. This step provided all ASR text data with a standard time alignment per second. Figure IV-50 shows after the rounding step. The text pieces successfully point to starting timestamps of integer. Then, multiple ASR text with identical starting timestamps was merged into a single piece of text.

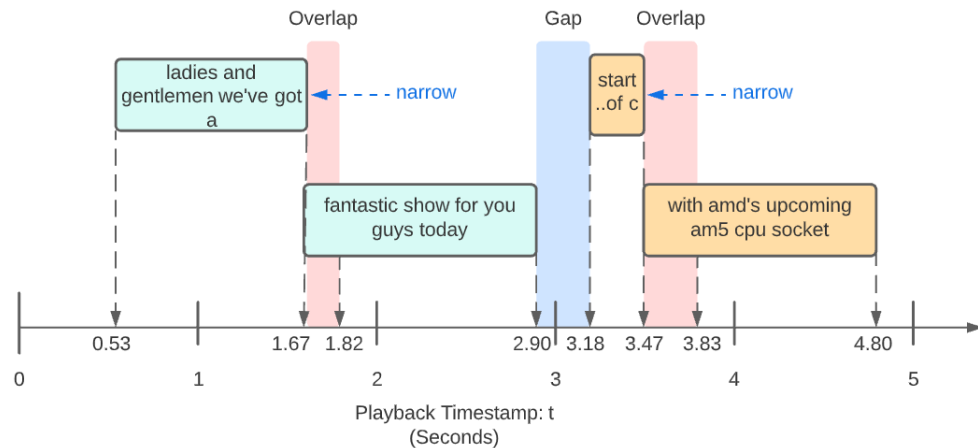


Figure IV-49: The overlapping issue was solved by trimming the duration of the first overlapped transcript.

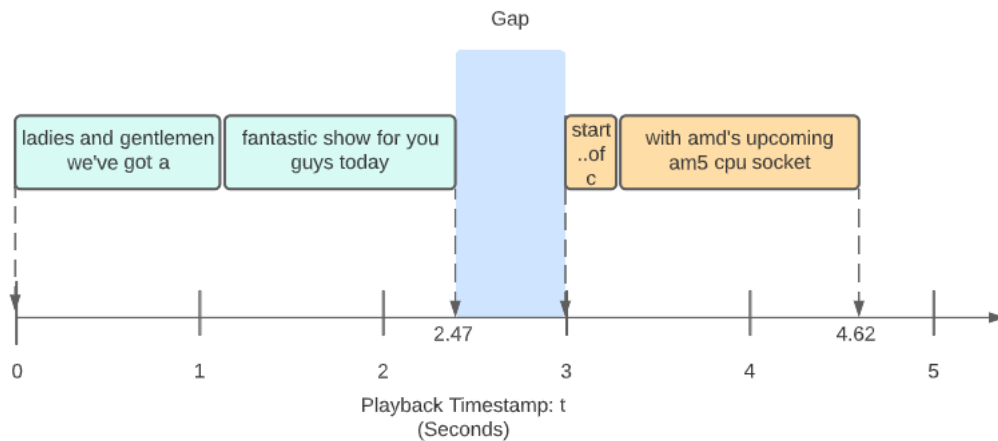


Figure IV-50: Transcripts generated by Automatic Speech Recognition (ASR) models are rounded to the closest integers on the time axis.

The three main parts of the algorithm are represented below (Algorithm 1, 2, and 3). The first algorithm solves the overlapping problem of the raw ASR text. The second piece of code aligns the raw timestamps into a proper time axis. The third one merges ASR text with identical timestamps. Some intermediate steps for connecting the three pieces of code are ignored due to the length of the paragraph.

### IV.F.3 ASR and TSCs Data Alignment

The ASR and TSCs data should be synchronized to the same time axis before sentiment analysis. I wrote a script to loop over each video's data files to combine the

---

**Algorithm 1** Solve for the ASR text overlaps
 

---

**Require:**  $i \geq 1$ 
 $N \leftarrow$  total number of text

 $t() \leftarrow$  return a timestamp

 $d() \leftarrow$  return a duration

**while**  $i \leq N$  **do**
 $d \leftarrow d(i)$ 
 $t_0 \leftarrow t(i)$ 
 $t_{start} \leftarrow t_0$ 
 $t_{end} \leftarrow t_0 + d$ 
 $T \leftarrow t_{end}$ 
**if**  $i \geq 2$  **then**
**if**  $t_{start} \leq T$  **then**
 $d(i - 1) \leftarrow t_{start} - t(i - 1)$ 
**end if**
**end if**
**end while**


---



---

**Algorithm 2** Round the ASR text timestamps
 

---

**Require:**  $i \geq 1$ 
 $N \leftarrow$  total number of text

 $t() \leftarrow$  return a timestamp

 $d() \leftarrow$  return a duration

**while**  $i \leq N$  **do**
 $d \leftarrow d(i)$ 
 $t_0 \leftarrow t(i)$ 
 $t_{start} \leftarrow \text{round}(t_0)$ 
 $t_{end} \leftarrow t_{start} + d$ 
 $t(i) \leftarrow t_{start}$ 
**end while**


---

---

**Algorithm 3** Concatenate multiple texts with identical timestamps
 

---

**Require:**  $i \geq 1$ , and text sorted by timestamps

```

 $N \leftarrow$  total number of text
 $t() \leftarrow$  return a timestamp
 $ASR() \leftarrow$  return a ASR text
while  $i \leq N$  do
   $asr \leftarrow ASR(i)$ 
   $t_i \leftarrow t(i)$ 
   $t_{start} \leftarrow t_i$ 
   $j \leftarrow i + 1$ 
  while  $j \leq N$  do
     $t_j \leftarrow t(j)$ 
    if  $t_j == t_{start}$  then
       $ASR(i) \leftarrow ASR(i) + ASR(j)$ 
       $ASR(j) \leftarrow Null$ 
    else
       $i = j$ 
      break
    end if
  end while
end while

```

---

two pieces of information. Figure IV-51 shows the temporal alignment of the ASR and TSCs data. Generally speaking, the script assigned each ASR and TSCs text with a specific timestamp to better standardize the data structure. Some timestamps may attach with long ASR or TSCs text, while others do not. The data structure will be later used in generating the sentiment values from textural analysis methods.

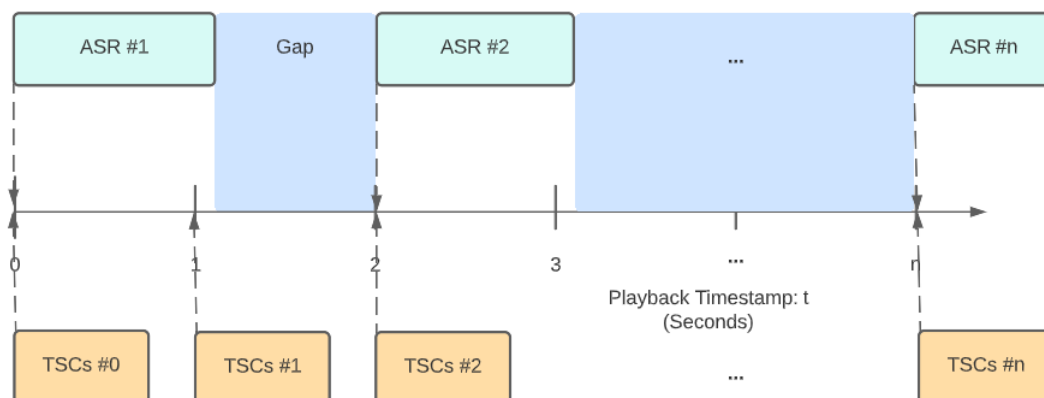


Figure IV-51: Transcripts generated by Automatic Speech Recognition (ASR) models are synced with TSCs data on the same time axis.

#### IV.F.4 Data Cleaning

Both ASR and TSCs data are text-based information gathered by the proposed data frame. Since multiple people contribute to this data, some free-form text exists. Before diving into sentiment analysis, the data needs to be 'cleaned' and structured. There are several steps for the data cleaning, and they are summarized as below:

1. Contractions expansion.
2. Symbols removal.
3. Tokenization.
4. Token stemming or lemmatization.
5. Stopwords removal.

The first step is fixing the contractions within the text. According to Saha ("Handle contractions in text preprocessing - NLP", 2022), "Contractions are words or combinations of words that are shortened by dropping letters and replacing them by an apostrophe." Abbreviations or short-forms are commonly used on online text, and they need to be expanded to retrieve their root forms to simplify the Natural Language Processing (NLP) task. I used a Python library called 'pycontractions' to expand the ASR and TSCs contractions (Beaver, 2019). The model relies on a grammar checker to calculate the Word Mover's Distance (WMD) between the converted sentences with their raw form to expand to the correct contractions. For instance, the sentence "I'd like to do something." can be translated into "I would like to do something."

The second step is removing every symbol from the text. Since most NLP algorithms only focus on the words or sentences, symbols are not necessary to be included. I used a regular expression in Python to exclude all characters from letters. This process will turn a piece of text: 'Hello, Bob. How are you doing today?' into something like: 'Hello Bob How are you doing today.' The step will change the raw text into a pure-letter form.

The third step is to tokenize the sentences. This step splits the sentences into small

pieces, also called 'tokens,' to find the words and punctuation within them. I used the Natural Language Toolkit (NLTK)'s default tokenizer (Bird et al., 2009) in this step. Figure IV-52 shows the tokenizer splits the sentence: "I just really enjoy talking to you." into multiple words.

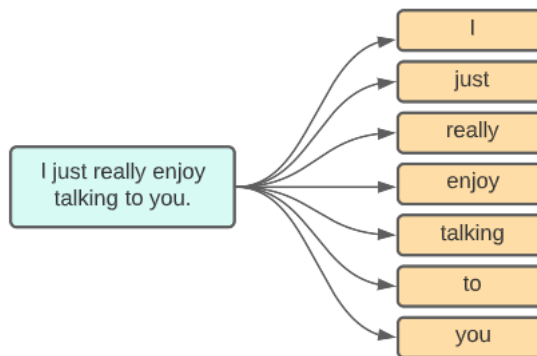


Figure IV-52: The tokenizer split a complete sentences into individual words.

The fourth step of the process is stemming or lemmatization. This part reduces tokens into their root or base form, allowing the sentiment model used later to recognize each word correctly. This step uses the WordNetLemmatizer from the NLTK library. The lemmatizer first converts each token into one of four syntactic categories: noun, verb, adjective, and adverb, and later translates the verb based on its syntactic. Figure IV-53 shows an example of tagging and stemming one token, 'talking,' in the middle of a sentence.

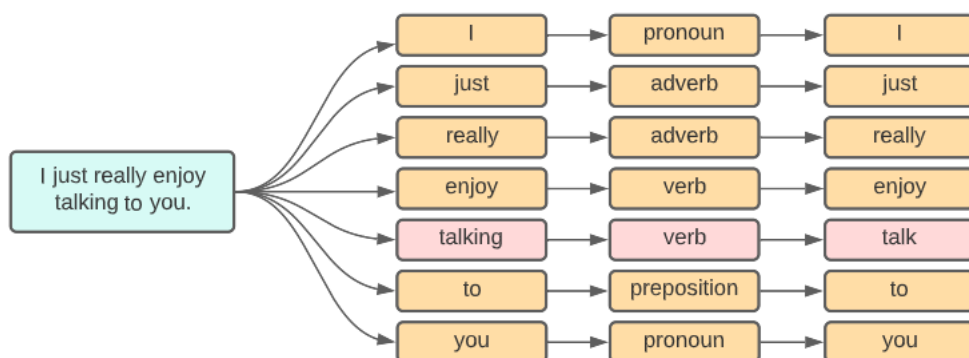


Figure IV-53: The lemmatizer covert the token 'talking' into 'talk'.

The final step is to remove stopwords from the token set. I used a pre-defined corpus from the NLTK library to filter out insignificant words. Figure IV-54 shows the process of removing stopwords from a token set.

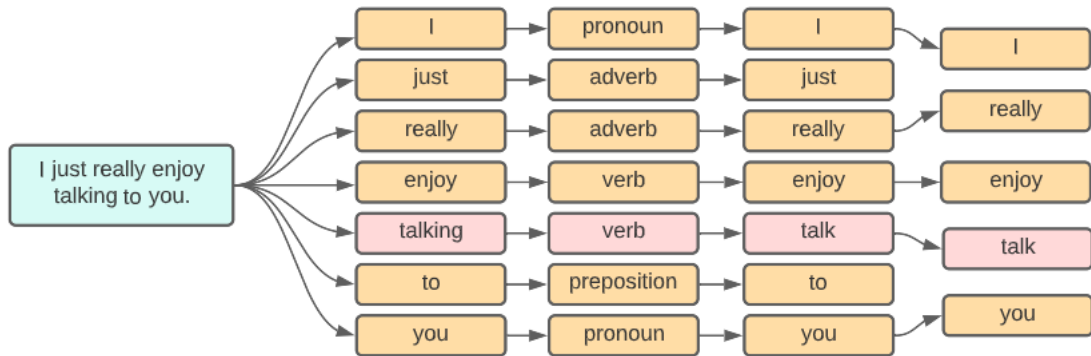


Figure IV-54: The process of removing stopwords from token set.

#### IV.F.5 Sentiment Conversion

Most comments from a live stream are short (a few words or sentences), a reasonable assumption is each comment only refers to one entity and contains only one opinion. Figure IV-55 shows the distribution of the number of words within a random 901 comments. Table 16 shows the average number of words within four different YouTube streams is both less than ten, which suggests most online comments only consist of one sentence or less.

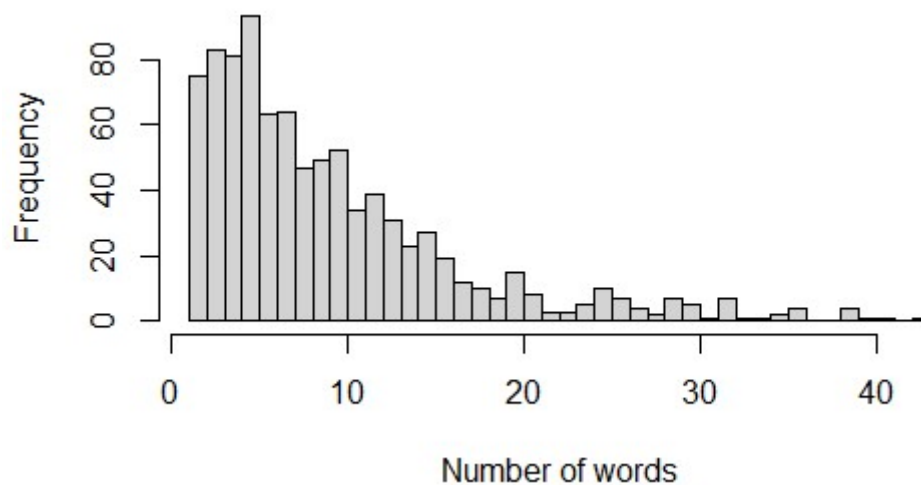


Figure IV-55: Number of words' distribution for a random 901 comments.

<b>Min</b>	<b>1st Qu</b>	<b>Median</b>	<b>Mean</b>	<b>3rd Qu</b>	<b>Max</b>	<b>Mode</b>	<b>Total</b>
1	4	7	9.55	12	43	5	901
0	2	4	4.834	6	36	3	3595
1	3	5	6.622	9	29	3	460
1	2	4	5.412	7	45	1	8245

Table 16: A summary shows the statistic of words in the comments of four different YouTube streams. Thirteen thousand two hundred and one comments are analyzed, and most comments are less than ten words.

The first step is to identify the subjectivity of each comment. Both supervised and unsupervised approaches can be used in this classification. Wiebe et al., 1999 proposed using a Naive Bayes approach on tags assigned by human judges and achieving a good result. However, getting human annotation for large-scale sentences is difficult. We often want to use external knowledge to determine whether sentence contains subjective components.

Some linguistic features, such as N-grams or part of speech (POS), can be used to identify the subjectivity of a sentence (Chenlo and Losada, 2014). Previous studies showed the uni-gram presence of subsets of adjectives and verbs could provide important subjective information. Therefore, I can leverage a list of predefined sentiment words and combine them with target sentences to derive overall sentiment scores. Previous studies used the lexical approach, but recent studies have changed to more complex methods involving neural networks.

Table 17 shows different schemes of sentiment models exist in literature. Most of these lexicons use a polarized measurement of positive and negative sentiments.

To check the sensitivity of live-comments-based sentiment, I captured a video clip from YouTube Live with 45 minutes and 18 seconds and performed extract sentiment from the TSCs. One hundred fifty-two unique users contribute to 460 comments on 420 unique timestamps, covering around 14 percent of the total video length.

Figure IV-56 and Figure IV-57 show the sentiment plot of average polarized sentiment values with a window size of 60 seconds and 180 seconds. I notice the existence of



Name	Year	Measurement	Method
OpinionFinder	2005	Positivity/Negativity	Lexical
SentiWordNet	2006	Positivity/Negativity/Objectivity	Lexical
NRC	2010	Positivity/Negativity/Objectivity Specific emotions	Lexical
SentiStrength	2010	Positivity/Negativity/Netural	Lexical
AFINN	2011	Positivity/Negativity	Lexical
VADER	2014	Positivity/Negativity/Netural	Lexical
TensiStrength	2017	Relaxation/Stress	Lexical
Flair	2020	Positivity/Negativity	Embedding-based

Table 17: A list of popular sentiment lexicons exist in Sentiment Analysis literature.

multiple video events by identifying "peaks" and "valleys" from plots. The result shows that even 14 percent coverage of affect annotation can be good enough to represent some video events. While the most popular videos have over 40 percent timestamp coverage, it is suitable to use the converted values as actual sentiment labels for the original video.

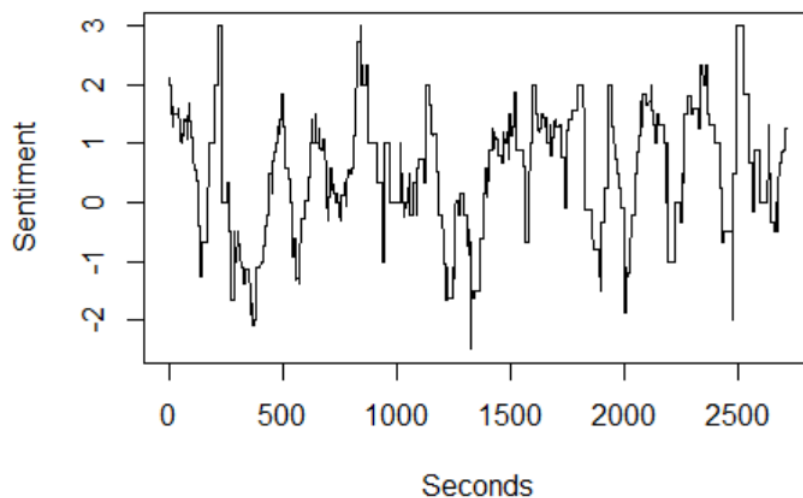


Figure IV-56: The plot of average polarized sentiment on a 60 seconds window size.

After comparing and experimenting with different sentiment models, I decide to use the sentiment analyzer, VADER (Hutto and Gilbert, 2014), as my sentiment annotator. The VADER is a rule-based soft sentiment classifier for social media-style text. It operates on a sentence level and can predict three positive, negative, and neutral

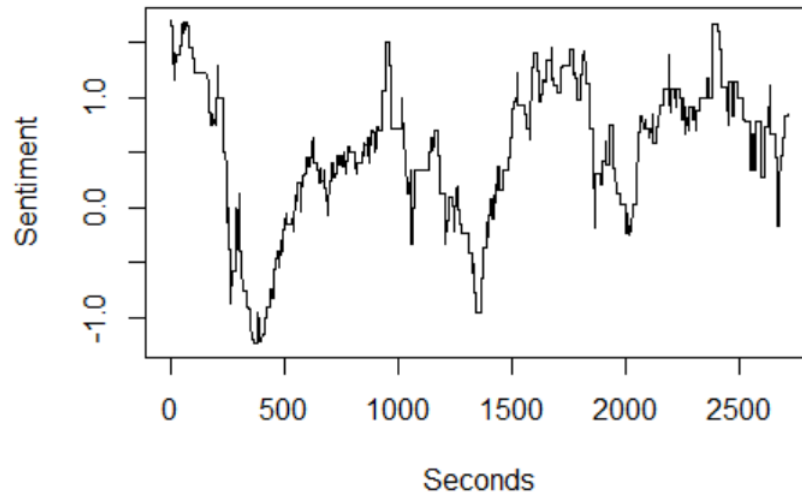


Figure IV-57: The plot of average polarized sentiment on a 180 seconds window size.

sentiment probabilities.

#### IV.F.6 Data Imputation

First, I converted each set of tokens into a single sentiment value on each timestamp. I figured some gaps existed in the sentiment sequence. These gaps are due to the missing values in the raw TSCs and ASR data. The gap could be filled with values inferred from local results. I used a linear interpolation algorithm to complete the sequences.

Then I applied an average rolling method to accumulate the sentimental values to their average (Algorithm 4). A fixed-length moving window was selected to generate sentiment sequences on different resolutions. I converted my data based on three types of window sizes: 100 seconds, 300 seconds, and 600 seconds. The reason for doing this is that the raw data is noisy. The sentiment sequence will be smoothed and filtered by applying a moving window. The moving average imputing algorithm is shown below. After the above steps, I have a one-to-one matching for each timestamp for the ASR and TSCs sentiment labels.

After smoothing and linear interpolation, the sentiment sequence with different window sizes can be viewed in Figure IV-58. Three sentiment sequences are plotted in red, blue, and green. The red line is created by a 100 seconds window, which has more spikes and is noisy. In contrast, the green line is created by a 600 seconds window,

---

**Algorithm 4** Data imputing algorithm
 

---

**Require:**  $i \geq 0$ 
 $T \leftarrow$  largest timestamp

 $\delta t \leftarrow$  window size

 $t() \leftarrow$  return a timestamp

 $s() \leftarrow$  return a sentiment value

**while**  $\delta t/2 \leq i \leq T - \delta t/2$  **do**
 $t_{min} \leftarrow t(i) - \delta t/2$ 
 $t_{max} \leftarrow t(i) + \delta t/2$ 
 $s_{tmp} \leftarrow 0$ 
**while**  $t(i) - \delta t/2 \leq j \leq t(i) + \delta t/2$  **do**
 $s_{tmp} \leftarrow s_{tmp} + s(j)$ 
**end while**
 $s_{avg} \leftarrow s_{tmp}/\delta t$ 
 $s(i) \leftarrow s_{avg}$ 
**end while**


---

which is more smooth. Different window sizes emphasize sentiment features on different resolutions.

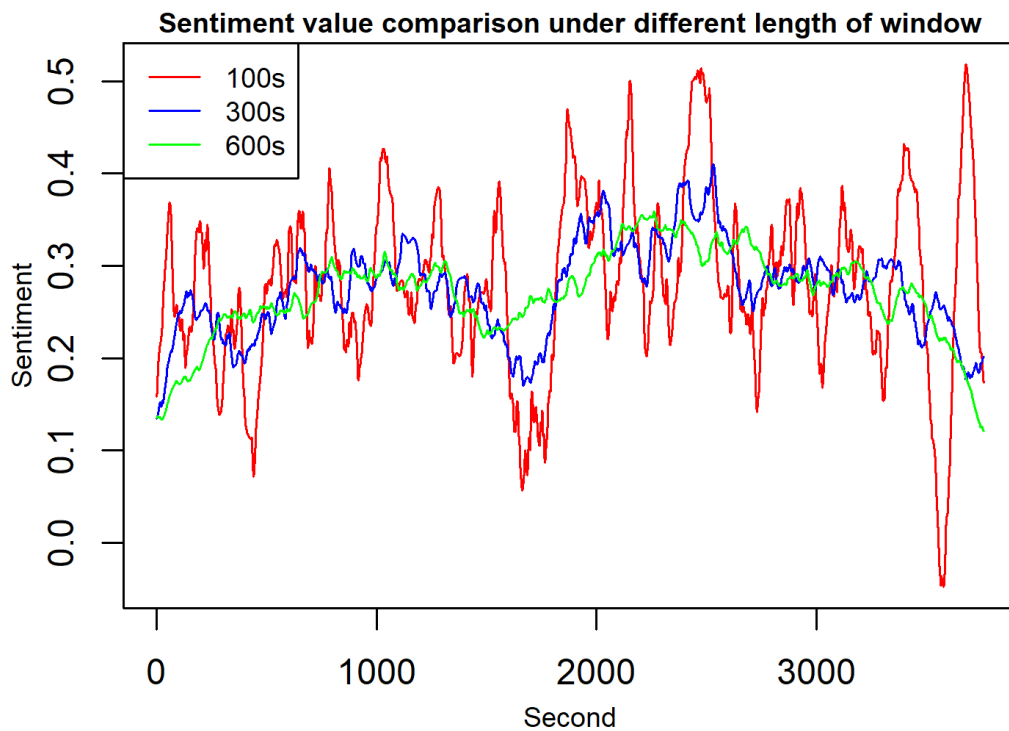


Figure IV-58: Sentiment sequence comparison under window sizes of 100, 300, and 600 seconds.

#### IV.F.7 Semantic Similarity

I hypothesize that the semantic similarity between the TSCs and ASR data could influence the sentiment mapping. To prove that, I need a way to evaluate the semantic distance between two text groups. I solved this problem in three folds. First, I converted my text into a group of vectors in semantic space with an embedding algorithm. Then I evaluated the distance between vector clusters' mass. Finally, I repeated the previous step for each timestamp with a similarity value based on its TSCs and ASR data.

I used embedding algorithms (Word2Vec) to convert tokens into vectors. The Word2Vec algorithms (Mikolov et al., 2013) are a set of neural network architectures which could group tokens from contexts into numerical representations based on their meaning and past appearances. Tokens with similar meanings, such as synonyms, will be placed close to each other in the embedding space. I trained a Word2Vec model on the 167 videos using Python's gensim library. The model contains 20770 tokens vocabulary and projects them into a 100-dimensional embedding space. A plot of its first and second PCA components is shown in Figure IV-59.

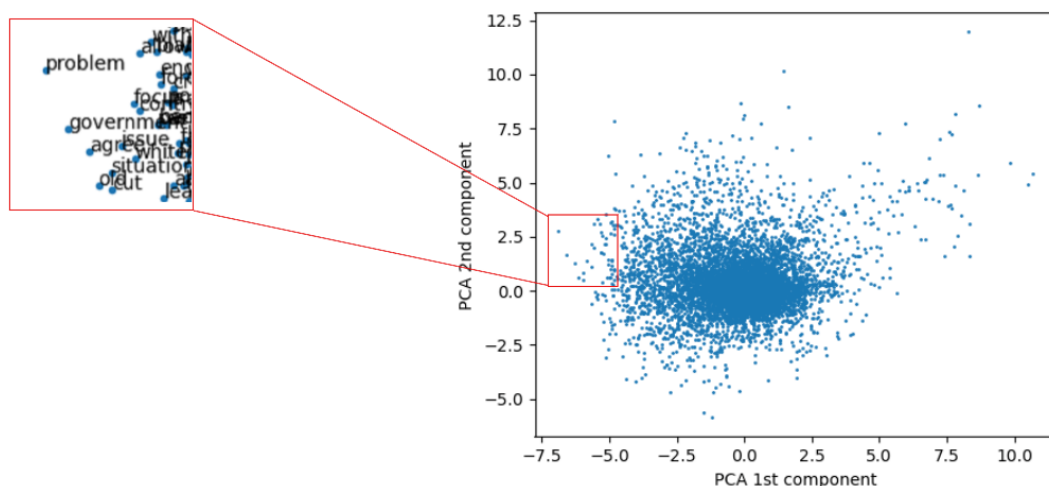


Figure IV-59: The 1st and 2nd PCA component embedding space the Word2Vec model trained on the 167 videos. Words with similar meanings are placed closely together.

Through a fixed-size time window  $w$ , I can appoint each timestamp at time  $t$  with a

group of text  $T_{t-w/2}, T_{t-w/2+1}, \dots, T_{t+w/2}$  and convert them into embedding vectors  $\vec{v}_{t-w/2}, \vec{v}_{t-w/2+1}, \dots, \vec{v}_{t+w/2}$  with the Word2Vec model I trained. Two groups of vectors were extracted from the TSCs and ASR data. Because similar semantic representations are naturally clustered in this embedding space, I can use cluster distance  $D$  to measure the semantic similarity between the TSCs cluster  $V = \vec{v}_{t-w/2}, \dots, \vec{v}_{t+w/2}$  and ASR cluster  $U = \vec{u}_{t-w/2}, \dots, \vec{u}_{t+w/2}$  at a specific timestamp.

Ward's method (Ward Jr, 1963) measures the distance between clusters by evaluating the additional variance cost of merging them. It is a special form of the Lance-Williams dissimilarities (Equation 6) which specify the dissimilarity  $d$  between a point  $k$  to a cluster  $i \cup j$ , where  $\alpha_i = \frac{|i|+|k|}{|i|+|j|+|k|}$ ,  $\beta = -\frac{|k|}{|i|+|j|+|k|}$ , and  $\gamma = 0$ .

$$d(i \cup j, k) = \alpha_i d(i, k) + \alpha_j d(j, k) + \beta d(i, j) + \gamma |d(i, k) - d(j, k)| \quad (6)$$

The Ward's distance can be calculated by subtracting the sum of squares of individual groups from the merged one. Equation 7 shows the distance represents a weighted least squares between the center of two clusters:

$$\begin{aligned} D_{U,V} &= \sum_{i \in U \cup V} \|\vec{w}_i - \vec{m}_{U \cup V}\|^2 - \sum_{i \in U} \|\vec{u}_i - \vec{m}_U\|^2 - \sum_{i \in V} \|\vec{v}_i - \vec{m}_V\|^2 \\ &= \frac{n_U n_V}{n_U + n_V} \|\vec{m}_U - \vec{m}_V\|^2 \end{aligned} \quad (7)$$

where  $\vec{m}$  is the center of a cluster, and  $n$  is the number of samples within it.

After applying Ward's method to every timestamp, I formed a distance sequence that measures the semantic similarity between the TSCs and ASR text. Figure IV-60 shows an example of the semantic similarity plot on one video. Part of the video is out-of-sync between the ASR and TSCs sequence (peaks in red box).

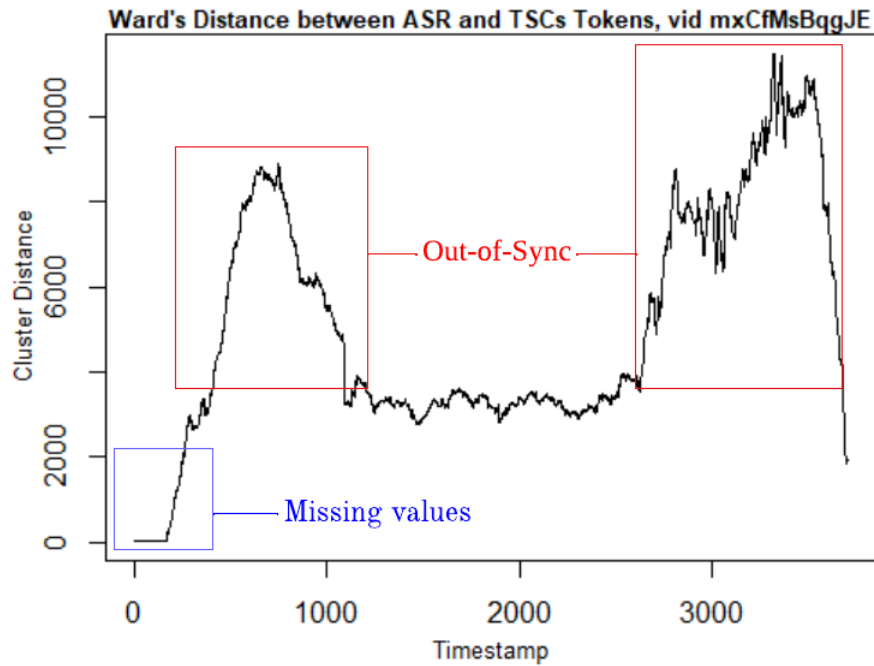


Figure IV-60: The cluster distance between TSCs and ASR tokens for video 'mxCfMsBqgJE.' Data was generated with a 300 seconds window size.

#### IV.G Model Structure

I used a particular neural network architecture, called long short-term memory (LSTM), to learn the relationship between the TSCs and ASR sentiment labels. The LSTM is a special type of Recurrent Neural Network (RNN) that memorizes past information through hidden layers (Hochreiter and Schmidhuber, 1997). Figure IV-61 shows the unrolled structure of the RNN network. This type of network is particularly suitable for solving the sequence-to-sequence problem.

The LSTM network implements memory cells in the RNN hidden layers to cope long and short memory. It uses three types of gates: input, output, and forget gate, to adjust the influence from past information (Hochreiter and Schmidhuber, 1997). I concatenate a shallow LSTM with full connection layers to map the TSCs sentiment sequence to the ASR sentiment sequence. Figure IV-62 shows how a LSTM layer connected with a full connection layer. The model is shallow since I only used a single layer of hidden neurons.

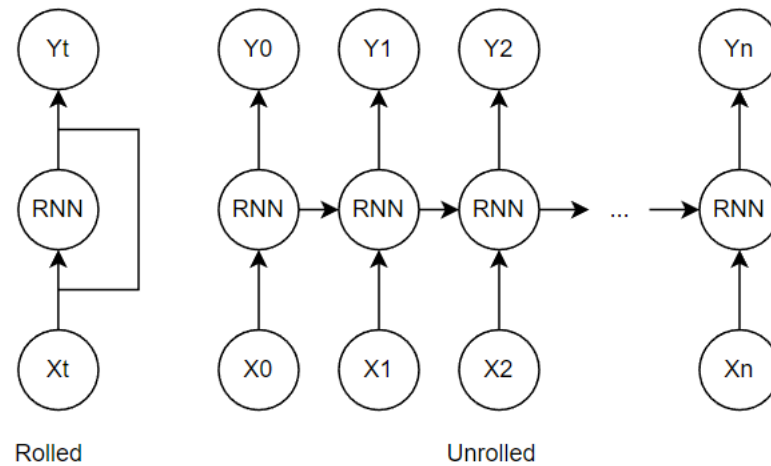


Figure IV-61: Structure of Recurrent Neural Network (RNN)

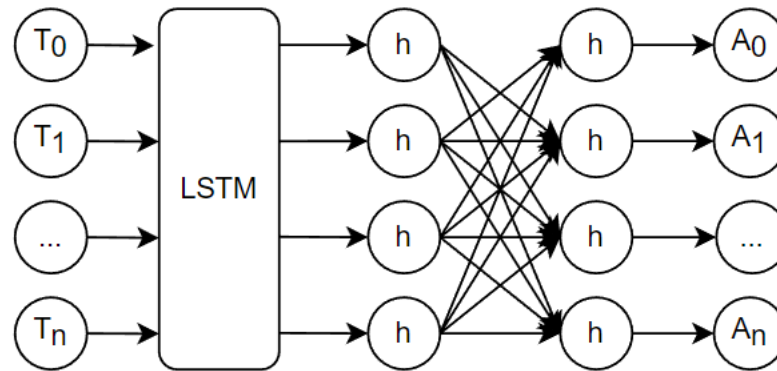


Figure IV-62: LSTM model with full connection layers.

#### IV.H Model Evaluation and Comparison

I set the Adaptive Moment Estimation (Adam) as my optimizer and 0.01 as my initial learning rate. I choose the Mean Square Error (MSE) (Equation 8) as my loss function in the neural network. I trained my model on 167 videos with different window sizes and sequence input length combinations. Eighty percent of the data was used as the training set, and the rest was used as the validation set. For each iteration, I picked 60, 120, and 180 seconds sentiment values from the 100, 300 and 600 window-smoothed TSCs/ASR data to feed into the LSTM model. The data size for each video is between 3600 to 7200. Every model is trained with ten epochs.

$$MSE = \frac{1}{N} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (8)$$

Then I add the semantic similarity feature to the LSTM inputs. Table 18 shows the MSE comparison between the two settings under different parameter combinations. I also calculated the improvement of the accuracy in percentage after adopting the similarity sequence.

<b>MSE Loss</b>	<b>WS = 100s</b>	<b>WS = 300s</b>	<b>WS = 600s</b>
<b>Input = 60s</b>	0.005576832	0.002621518	0.001477843
<b>Input = 60s (with similarity)</b>	0.005425467	0.002560942	0.001446142
<b>Accuracy Improvement</b>	2.91%	4.30%	3.64%
<b>Input = 120s</b>	0.002809225	0.001355333	0.0007483041
<b>Input = 120s (with similarity)</b>	0.002694337	0.001315808	0.0007271637
<b>Accuracy Improvement</b>	4.06%	4.41%	5.63%
<b>Input = 180s</b>	0.001865821	0.000921985	0.0005185198
<b>Input = 180s (with similarity)</b>	0.001741244	0.000868069	0.0004861399
<b>Accuracy Improvement</b>	6.69%	5.52%	4.52%

Table 18: This table compares MSE loss for different parameter combinations of the shallow LSTM model. WS stands for the window size, and Input stands for the inputs sequence length. Accuracy Improvement is the reduced percentage of MSE loss after applying the similarity feature to the model.

Figure IV-63 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 60 seconds and generated through a 100 seconds window. The mean losses are 0.005576832 and 0.005425467 (not include and include similarity feature). Around 78.4 percent of results have less MSE after adding the similarity feature.

Figure IV-64 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 120 seconds and generated through a 100 seconds window. The mean losses are 0.002809225 and 0.002694337 (not include and include similarity feature). Around 80.8 percent of results have less MSE after adding the similarity feature.



Figure IV-65 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 180 seconds and generated through a 100 seconds window. The mean losses are 0.001865821 and 0.001741244 (not include and include similarity feature). Around 85.6 percent of results have less MSE after adding the similarity feature.

Figure IV-66 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 60 seconds and generated through a 300 seconds window. The mean losses are 0.002621518 and 0.002560942 (not include and include similarity feature). Around 74.3 percent of results have less MSE after adding the similarity feature.

Figure IV-67 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 120 seconds and generated through a 300 seconds window. The mean losses are 0.001355333 and 0.001315808 (not include and include similarity feature). Around 70.7 percent of results have less MSE after adding the similarity feature.

Figure IV-68 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 180 seconds and generated through a 300 seconds window. The mean losses are 0.000921985 and 0.0008680694 (not include and include similarity feature). Around 79 percent of results have less MSE after adding the similarity feature.

Figure IV-69 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 60 seconds and generated through a 600 seconds window. The mean losses are 0.001477843 and 0.001446142 (not include and include similarity feature). Around 64.7 percent of results have less MSE after adding the similarity feature.

Figure IV-70 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 120 seconds and generated through a

600 seconds window. The mean losses are 0.0007483041 and 0.0007271637 (not include and include similarity feature). Around 76 percent of results have less MSE after adding the similarity feature.

Figure IV-71 shows the histogram of MSE distribution from the LSTM model trained on 167 videos' dataset. The input sequence is 180 seconds and generated through a 600 seconds window. The mean losses are 0.0005185198 and 0.0004861399 (not include and include similarity feature). Around 66.5 percent of results have less MSE after adding the similarity feature.

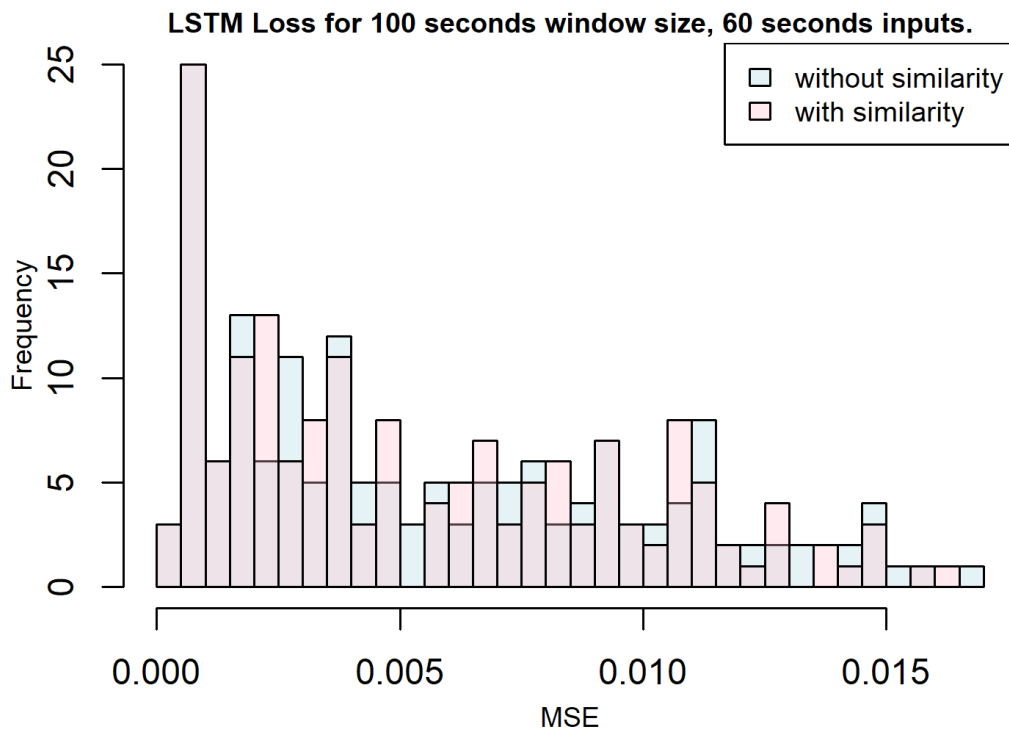


Figure IV-63: MSE loss of the LSTM model trained on 60 seconds inputs generated from a 100 seconds smoothing window.

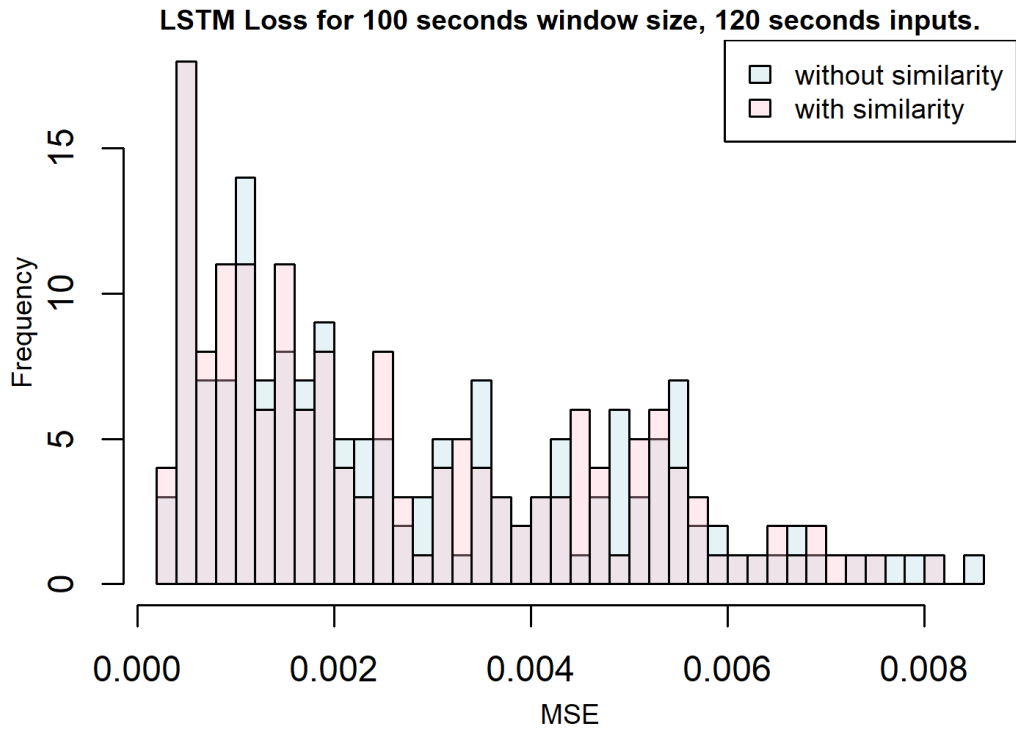


Figure IV-64: MSE loss of the LSTM model trained on 120 seconds inputs generated from a 100 seconds smoothing window.

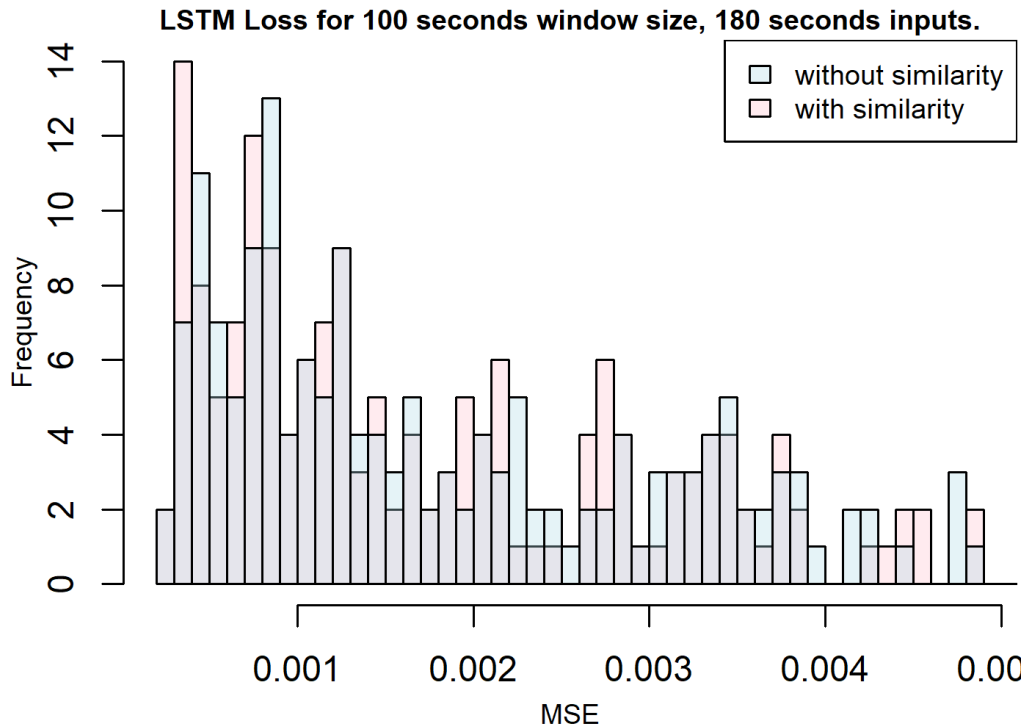


Figure IV-65: MSE loss of the LSTM model trained on 180 seconds inputs generated from a 100 seconds smoothing window.

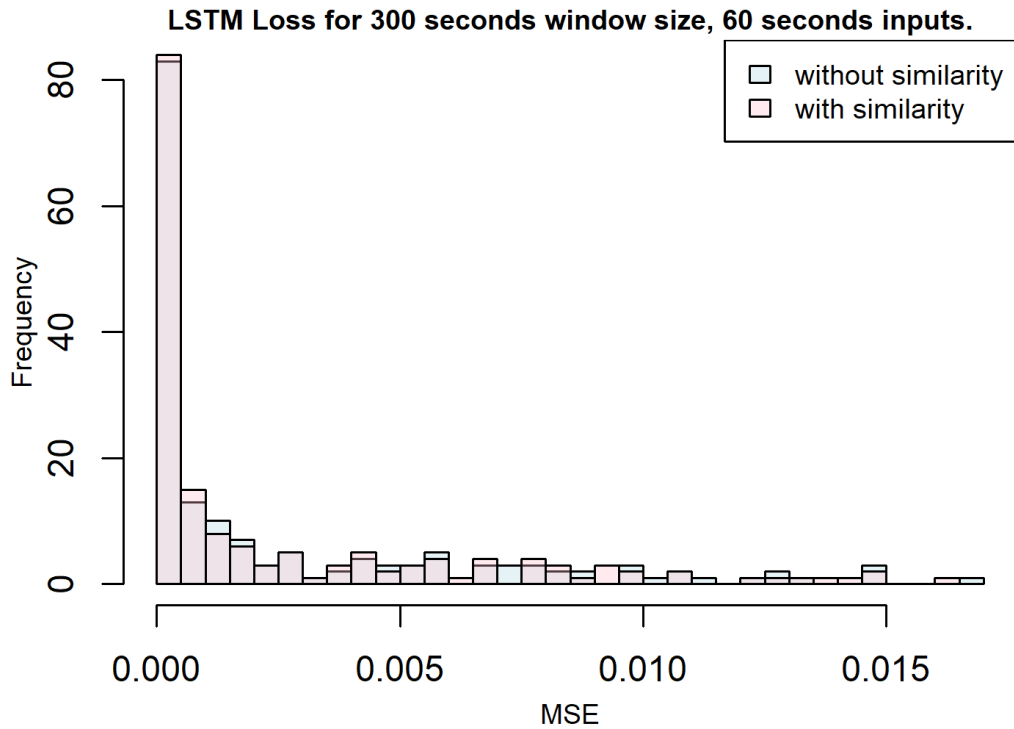


Figure IV-66: MSE loss of the LSTM model trained on 60 seconds inputs generated from a 300 seconds smoothing window.

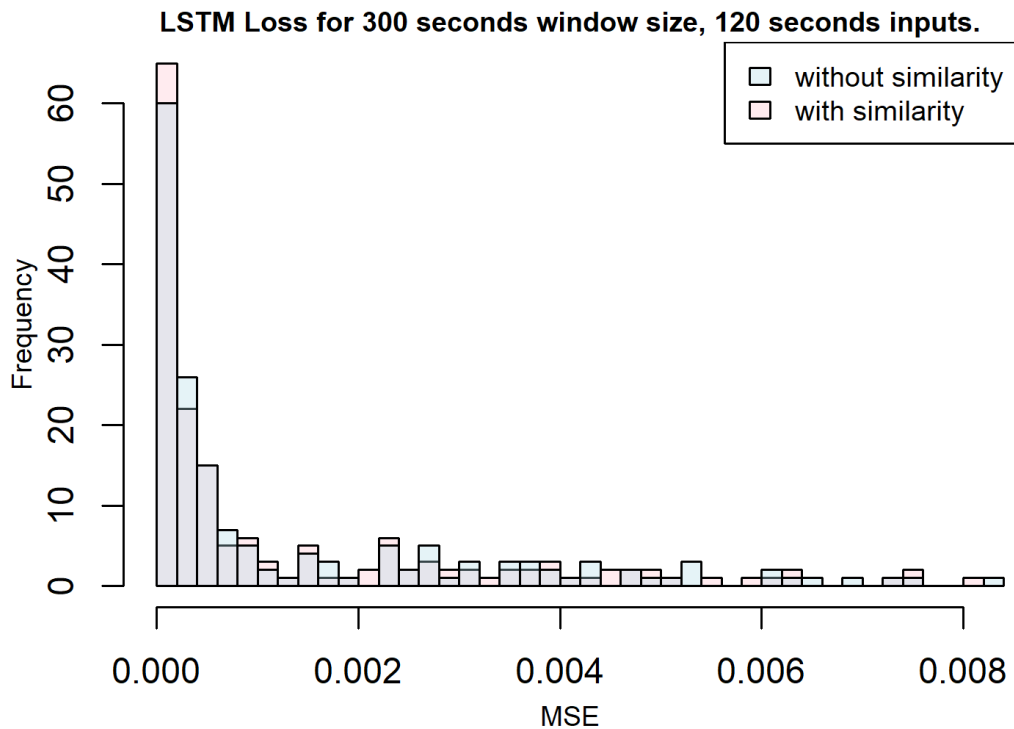


Figure IV-67: MSE loss of the LSTM model trained on 120 seconds inputs generated from a 300 seconds smoothing window.

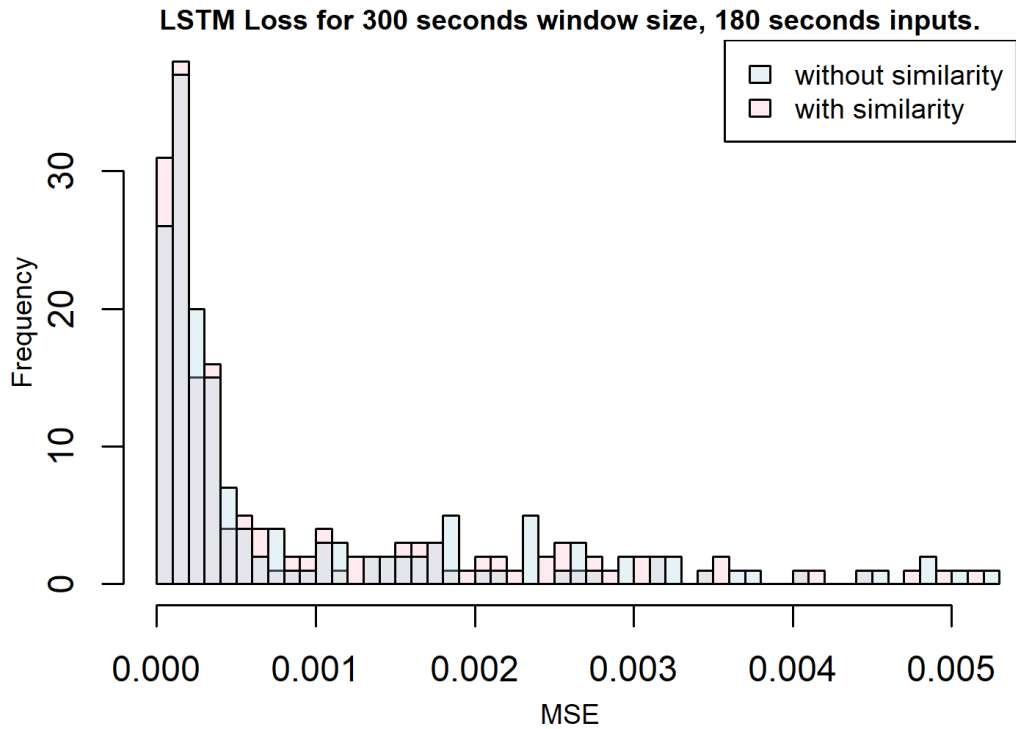


Figure IV-68: MSE loss of the LSTM model trained on 180 seconds inputs generated from a 300 seconds smoothing window.

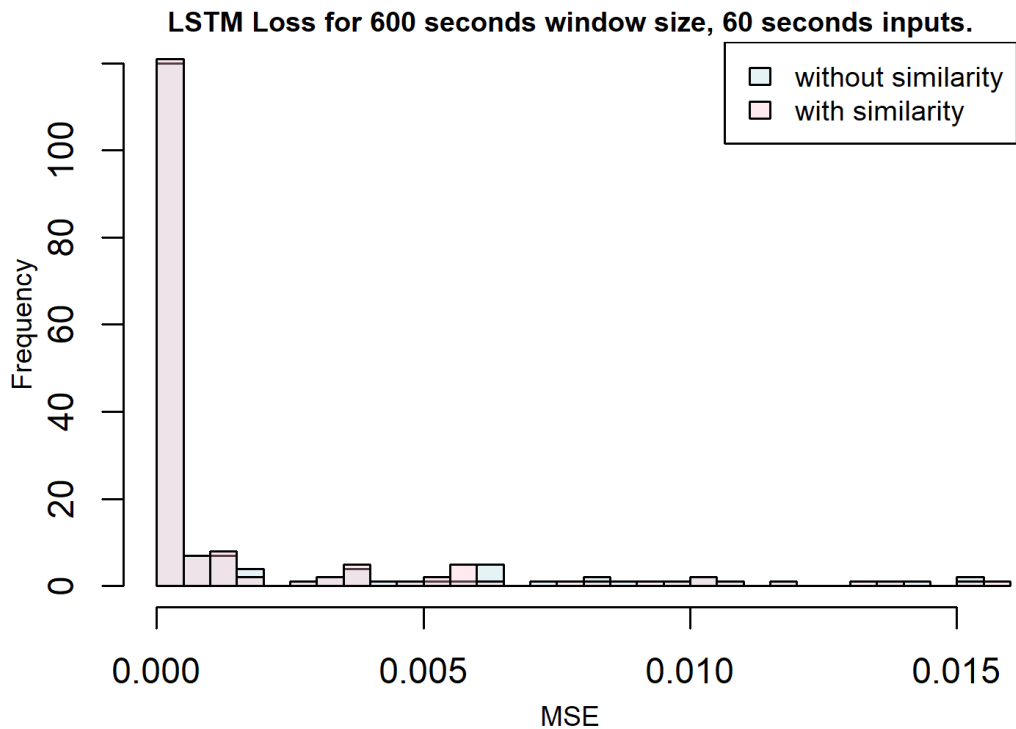


Figure IV-69: MSE loss of the LSTM model trained on 60 seconds inputs generated from a 600 seconds smoothing window.

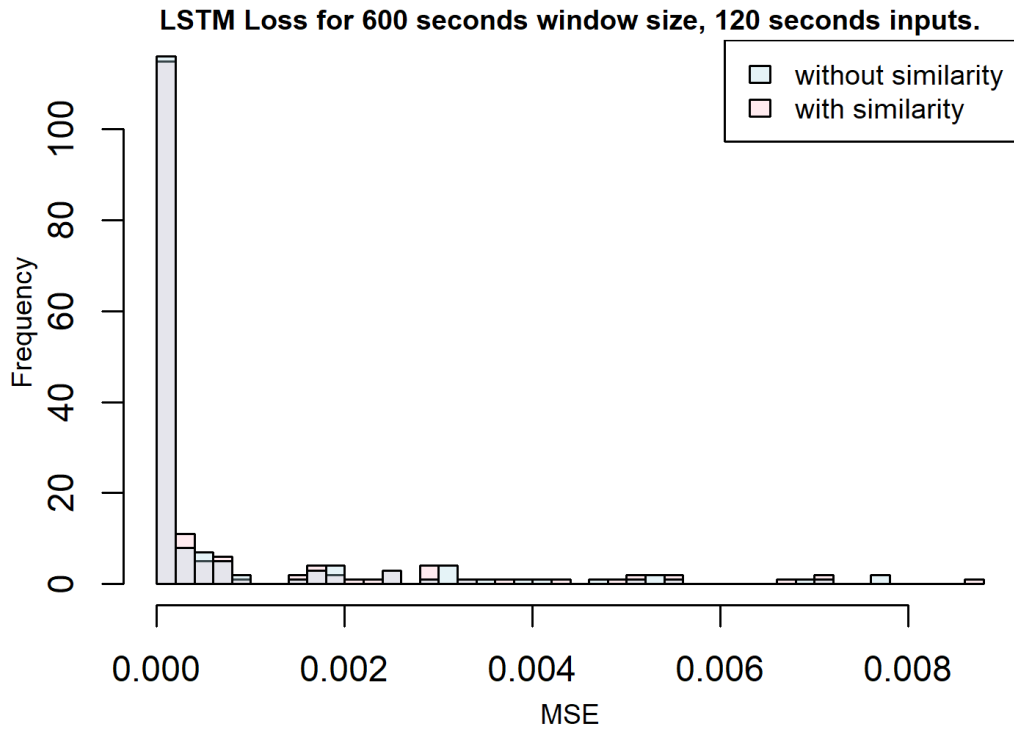


Figure IV-70: MSE loss of the LSTM model trained on 120 seconds inputs generated from a 600 seconds smoothing window.

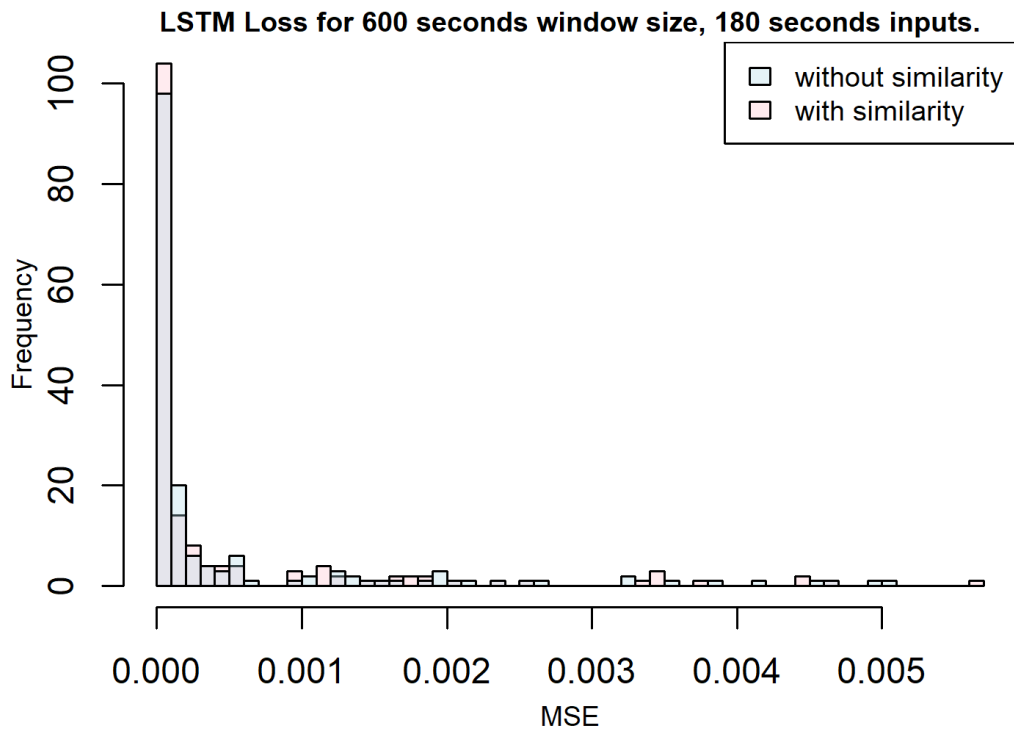


Figure IV-71: MSE loss of the LSTM model trained on 180 seconds inputs generated from a 600 seconds smoothing window.

The models' comparison experiment shows that the larger the window size is, the closer TSCs to ASR sentiment values are. A longer sequence used in the LSTM could achieve a better temporal sentiment representation. The experiment also suggests combining a long input sequence and a short smoothing window size is more likely to improve the final accuracy using the similarity feature (higher accuracy improvement). The experiment also shows the effectiveness of using the TSCs sentiment to predict the ASR sentiment for online videos. The small MSE results prove the sentiment mapping approach in this study works well.

## **V Summary**

This study proposed a general approach to converting online comments into sentiment labels. I demonstrate efficient strategies to collect existing comments from multiple users directly from YouTube. I also developed a comprehensive data pipeline to crawl online TSCs data automatically through months of experiments. The outcomes involve two large-scale TSCs datasets extracted from two different sets of keywords. Then I combined the sentiment analysis with the NLP techniques to successfully map the temporal data into sentiment sequence. My sentiment modeling experiment shows the close connection between the TSCs' converted sentiment scores and the live streams' original sentiment.

The above experiments suggest my data collection frameworks work well, and the sentiment mapping approach is suitable for most online streams. Although I only conducted my study on one video distributed platform, the broad coverage of my video contents and topics should represent a large portion of the live stream population. I believe the work could be extended to other sites as well. The goal of exploring the low-cost, efficient, and accurate sentiment mapping strategy for online time-sync data is achieved.

### **V.A Discussion**

This section will summarize the limitations, conclusions, and potential future works of the study. I will discuss each aspect in detail and provide a deep insight into further improvement of the project in future.

#### **V.A.1 Data Collection**

The top four categories contributed to my LST-YF20 dataset is "Entertainment," "Gaming," "People & Blog," and "Howto & style." In 2016, Google reported that the top four content categories watched by YouTube users are comedy, music, entertainment/pop culture, and "how-to" (ThinkwithGoogle, 2016), which is different from the result in my dataset. The reason behind this shift probably results from the blooming of live streaming on influential video-sharing and social media platforms.



When more individuals were involved in the streaming, personal activities got more exposure to the public world. This phenomenon provides researchers an opportunity to understand the dynamic interactions between authors and streaming content on different types of videos.

From the research perspective, most studies prefer videos that contain dense TSCs over sparse ones. It is simply because the more TSCs appear in a fixed period, the more accurate information they can carry to reflect the implicit features of the video contents. I noticed that using different search terms in my pipeline may vary the returning result significantly. For example, the search term "apple" initially returns 2.4 times more videos than other search terms in the LST-YF20 experiment but is later surpassed by "pear" within 30 days. Selecting the correct search term may eventually accelerate the data collection process. However, one drawback of this strategy is, the variations of the TSCs are limited and may cause data bias in future studies.

We should notice, different online media platforms may contain different types of video in general. The user base of such websites may differ from others, which may cause a problem in data bias. Some previous works (Liao et al., 2018, Ma et al., 2019, Yang, Wang, et al., 2019, W. Wang et al., 2020) only collect data from non-English speaking platforms such as BiliBili and AcFun, which geographically focus on the East Asia audience groups. Extending the language settings to other regions of the world may significantly reduce getting bias results. It is also worth noticing that BiliBili and AcFun are heavily influenced by anime culture among teenagers in China. Researchers using these TSCs data should be aware of such influence and adjust their research accordingly.

My data pipeline proves it is possible to create a decent amount of TSCs data directly from online social media platforms on low-budget devices. Compared with similar project (Uechi, 2021), we do not require a professional online cluster with thousands of crowd-sourcing clients working globally. With only eight runs in 20 days, I gathered 105 million TSCs, equal to one-fifth of the total number of TSCs crawled by

the previous crowd-sourcing methods in a longer run. Meanwhile, the agility of my search term configuration enables me to get even larger TSCs datasets with only a few keywords expansion.

Thus, I conclude that my data mining strategies can produce promising TSCs datasets needed in research from quality and quantity perspectives. My experiment paves the way to further optimizing data mining TSCs data and contributes to future TSCs research.

### **V.A.2 Data Acquisition**

Crawling live comments on a large scale is highly limited for several reasons.

YouTube does not provide an easy way to retrieve and restore chat messages once video streamings are finished. Compared with Bilibili, YouTube does not have a specialized API for pulling historical live comments for videos. Some live channels on YouTube allow viewers to replay the chat, but many broadcasters disallow this option, limiting the choice of crawling live comments from such channels. The current strategy of recording those comments is crawling live comments during streaming and manually aligning them with the video timeline afterward, significantly limiting the comments sample size. On the other hand, crawling comments from other channels will also double the mining time, eventually slowing down the speed and magnitude of crawling.

Due to the limitation of storage capacity, large-scale video data mining is challenging. Since most videos are recorded at 30 frames per second, storage requirements raise rapidly as more videos put pressure on the local disk. Therefore, only a small portion of videos can be stored and analyzed locally.

### **V.A.3 Speech Recognition**

Although extracting audio features directly from live streams sounds like a straightforward approach. Video streams are recorded from multiple devices in practice and implemented with appealing musical effects. Post-edited clips may introduce unnecessary noise into the audio channel, lower the quality of the audio

signal, and interrupt the feature extraction procedure. Using the vanilla audio features without careful selection may encounter problems.

The quality of speech feature recognition depends on speech clarity and ASR model accuracy. Both offline ASR-generated text and automatic captions might misrepresent the spoken content due to mispronunciations, accents, dialects, or background noise. Having a noisy environment or background music in streams may reduce the final speech recognition quality. The accuracy of the ASR engines will also influence the final result of speech recognition. If an ASR model could only recognize a limited number of words, using it for a general speech recognition task will only provide a limited result. We can avoid this problem by first replacing the original feature recognition model with a better one, then manually selecting specific videos with little environmental sound and good clarity of the speaker's voice.

#### **V.A.4 Sentiment Annotation**

First, live comments contain not only textual information but also emojis. The traditional linguistic approach for sentiment analysis often overlooks such information during processing. However, emojis are popular in live comments and convey a solid subjective feeling of the author. Neglecting emojis can result in missing crucial emotional information and reduce the accuracy of overall sentiment measurement.

Another disadvantage of using a general sentiment lexicon for live comments is overlooking internet slang. Since live commenting is more or less like a form of real-time online communication, internet slang and abbreviation frequently appear in comments and quickly evolve each year. For example, slang such as "LOL" and "LMAO" is often observed on YouTube Chats. A general-purpose lexicon does not capable of interpreting such slang and its variants into sentiments.

Moreover, not every video contains sufficient live comments for analysis. Only videos from the popular channel have enough viewers to comment. For example, only gaming videos are popular on Twitch and have enough live comments. Collecting live comments from a single platform may introduce video and audience sampling bias.

This partial data may limit further generalization of the sentiment model.

Live comments are not evenly distributed - some videos may have missing annotations. Data imputing must be done to fix gaps between annotations. If dramatic sentiment change exists between the annotation intervals, then imputing will introduce bias into the dataset. Otherwise, a semi-supervised labeling algorithm for training data with small-size labeling must be performed to predict and fill the label gaps.

Live comments are not memory-less. The internal feeling of viewers may last for a long time, sometimes after the video event is finished. Therefore taking future sentiment changes into current sentiment evaluation is crucial for producing an accurate result. But how to choose a correct window size for future sentiment is still unknown.

#### **V.A.5 Sentiment Modeling**

The sentiment modeling results show a close connection between the converted TSCs with the ASR sentiment values. A larger smoothing window size could significantly improve the accuracy of predicting ASR sentiment through TSCs sentiment. A larger window could also capture long-term temporal patterns in the TSCs and achieve higher prediction accuracy. Human feeling could not change rapidly in a short time, which may explain why using a larger window could improve the accuracy of sentiment mapping. Additionally, applying the more extended sequence as inputs for the shallow LSTM network could improve the mapping quality. I believe the LSTM could identify some long-term temporal features by feeding with long data series. After applying the semantic similarity features to the LSTM model, the prediction accuracy improves on all parameter settings. And the smaller the window size is, the more significant accuracy improvement is. The phenomena could be explained by the out-of-sync issue of the TSCs and ASR sentiment sequence. When audiences' attention is drawn to a particular topic, they no longer comment on the video contents. In this case, data smoothed with a small window size could be very sensitive to the video topic. Some video events may be so emotional that the sentiment effect lasts

longer among the comments.

#### **V.A.6 Semantic Distance**

I used Ward's distance to measure the semantic distance between the ASR and TSCs text clusters for each timestamp. Ward's distance equation is a unique case of the Lance-Williams dissimilarity measurement. Other forms of dissimilarity between text clusters, such as Gower's method (WPGMC) or Centroid method (UPGMC), can be applied instead. The optimal measure is yet to be identified and may result in higher accuracy in representing semantic distance. Additional analysis should be done to compare these measurements in future studies.

The semantic distance was calculated on each timestamp's ASR and TSCs text sequence. The number of texts is determined by the intensity of online comments and video dialogue. This approach usually does not work well due to the inconsistency of the flow of words. Expanding text coverage to nearby timestamps could resolve the problem. However, this process will introduce unexpected semantic correlations between the two text clusters and influence the subsequent studies. Finding an optimal cluster size by analyzing the intensity of neighboring text is essential.

#### **V.B Conclusion**

This project proposed a data pipeline collecting TSCs data directly from online social media platforms, specifically YouTube. I experimented with automatic data collection and produced two large-scale TSCs datasets, namely LST-YF20 and LST-YT1000. My result proved the efficiency of the proposed data pipe and the quality of collected TSCs data. To illustrate the difference between the previous works to my approach, I compared the most recent TSCs datasets in the literature. I am aware that several limitations exist for the data pipeline. For example, a predefined search term list is required before data collection, and the data collection speed depends on the popularity of search terms. However, I believe, with small adjustments, the data pipeline can significantly contribute to the field by constantly providing the most recent TSCs to researchers. I also demonstrated a way of using mobile devices at a low cost to get an extensive collection of TSCs data. It potentially could contribute to creating a data

mining network shared by researchers around the globe. In the future, I plan to share this tool with the research community under an open-access license to benefit researchers with constrained budgets and resources. Moreover, I explored the influence of adopting different smoothing windows and networks on the ASR sentiment prediction. I conclude my dataset could potentially be helpful for researchers working in the sentiment analysis field. I hope my work will also contribute to the advent of autonomous temporal data collection for multimedia studies.

### **V.C Future Work**

The study demonstrates a way of converting online time-sync data into an accurate sentiment map and creating two large-scale TSCs datasets. The work can be extended to further analysis of the internal emotional connection between the video contents and the audience. The dataset generated by the data pipeline can also be used in other multimedia tasks, such as video event detection, video topic modeling, and audience reaction prediction. Beyond that, I'm interested in creating more TSCs datasets using the tools I've developed in this study. Because the data pipeline can work on multiple mobile devices, creating distributed data mining networks seems interesting. More researchers will benefit from the project. I'm also interested in considering visual features in future studies. Recently research in transfer learning has provided a way of extracting semantic information from the visual signals. I believe if combining this information with the comments topic, I could eventually achieve a more accurate sentiment mapping. There is a lot of potential for future extension of this study, and I am eager to continue working on it.

## VI Study Timetable

This section presents the progress of the study. I started the literature review during the winter of 2020. After reading several papers, I realized that current multimedia research relies heavily on manually labeled video datasets. Creating a large-scale TSCs dataset is very expensive and challenging. Integrating online comments with automated scraping could generate high-quality TSCs datasets with little effort. I first prototyped a few programs to experiment with the different stages of data mining and later formalized the idea of creating an automatic data mining system for online streaming platforms in spring 2021. The project grew fast after I added more codes to it. Then I spent months testing the code and stabilized the code base in the summer of 2021. It was challenging since I rapidly experimented with different data mining strategies and techniques. In summer 2021, I started a large-scale data collection with my data pipeline. Several months were spent on the data collection, and I developed a graphic interface for more accessible interactions with the data pipeline. By the end of 2021, I got two large datasets: LST-YF20 and LST-YT1000. The data cleaning took me months to finish in 2022. Later, auxiliary data were added and aligned with my TSCs datasets. In March 2022, I started testing various network architectures. Then I selected one architecture with balanced performance and computational time based on my data size. The final result was analyzed and documented in the summer of 2022. Table 19 shows the detailed timetable for this study.

Stage	Task	Timeline
Literature review	Problem statement	Winter 2020 - Spring 2021
Prototyping	Test mining strategies	Spring 2021 - Summer 2021
Proposal	Write proposal	Spring 2021 - Spring 2021
Framework development	Merge components	Summer 2021 - Spring 2022
Data pipeline test	Small-scale test	Summer 2021 - Summer 2021
Data collection	Large-scale mining	Summer 2021 - Winter 2021
Data processing	Data cleaning	Winter 2021 - Spring 2022
Machine learning	Validate accuracy	Spring 2022 - Summer 2022
Report	Finish the study	Summer 2022

Table 19: The timetable of this study.

## BIBLIOGRAPHY

- Aminikhanghahi, S., & Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2), 339–367.
- Bakshi, R. K., Kaur, N., Kaur, R., & Kaur, G. (2016). Opinion mining and sentiment analysis. *2016 3rd international conference on computing for sustainable global development (INDIACom)*, 452–455.
- Beaver, I. (2019). Pycontractions: Intelligently expand and create contractions in text leveraging grammar checking and word mover's distance. *PyPI*. <https://pypi.org/project/pycontractions/>
- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: Analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."
- Brooke, J., Tofiloski, M., & Taboada, M. (2009). Cross-linguistic sentiment analysis: From english to spanish. *Proceedings of the international conference RANLP-2009*, 50–54.
- Captions YouTube Data API. (2021). Retrieved April 4, 2021, from <https://developers.google.com/youtube/v3/docs/captions>
- Chenlo, J. M., & Losada, D. E. (2014). An empirical study of sentence features for subjectivity and polarity classification. *Information Sciences*, 280, 275–288.
- Download YouTube Captions for Greasemonkey. (2014). Retrieved April 4, 2021, from <https://userscripts-mirror.org/scripts/show/50003>
- Duan, C., Cui, L., Ma, S., Wei, F., Zhu, C., & Zhao, T. (2020). Multimodal matching transformer for live commenting. *arXiv preprint arXiv:2002.02649*.
- Ekman, P. (1992). An argument for basic emotions. *Cognition & emotion*, 6(3-4), 169–200.
- Evangelopoulos, G., Zlatintsi, A., Potamianos, A., Maragos, P., Rapantzikos, K., Skoumas, G., & Avrithis, Y. (2013). Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention. *IEEE Transactions on Multimedia*, 15(7), 1553–1568.
- Facebook research wav2letter. (2021). Retrieved April 4, 2021, from <https://github.com/facebookresearch/wav2letter>



- Famin, W., Guangyi, L., Qi, L., Ming, H., Biao, C., Weidong, H., Hui, Z., & Le, Z. (2019). Deep semantic representation of time-sync comments for videos. *Journal of Computer Research and Development*, 56(2), 293.
- Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4), 82–89.
- Francisco, V., & Gervás, P. (2006). Automated mark up of affective information in english texts. *International Conference on Text, Speech and Dialogue*, 375–382.
- Fraser, C. A., Kim, J. O., Shin, H. V., Brandt, J., & Dontcheva, M. (2020). Temporal segmentation of creative live streams. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–12.
- Handle contractions in text preprocessing - nlp. (2022). *DEV Community*. Retrieved June 13, 2022, from <https://dev.to/edualgo/handle-contractions-in-text-preprocessing-nlp-21p>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Hokuto, T. (2020). GitHub - taizan-hokuto/pytchat: library for youtube chat. <https://github.com/taizan-hokuto/pytchat>
- Hutto, C., & Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the international AAAI conference on web and social media*, 8(1), 216–225.
- Kaggle. (2021). Retrieved September 28, 2021, from <https://www.kaggle.com/uetchy/vtuber-livechat>
- Kaldi. (2011). Retrieved April 4, 2021, from <http://kaldi-asr.org/>
- Korvas, M., Plátek, O., Dušek, O., Žilka, L., & Jurčiček, F. (2014). Free English and Czech telephone speech corpus shared under the CC-BY-SA 3.0 license. *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2014)*, To Appear.
- Kruse, R., Nauck, D., & Borgelt, C. (1999). Data mining with fuzzy methods: Status and perspectives. *Proceedings of The European Congress on Intelligent Techniques and Soft Computing (EUFIT-99)*.
- Liao, Z., Xian, Y., Yang, X., Zhao, Q., Zhang, C., & Li, J. (2018). Tscset: A crowd-sourced time-sync comment dataset for exploration of user experience improvement. *23rd International Conference on Intelligent User Interfaces*, 641–652.

- Liu, B. (2010). Sentiment analysis: A multi-faceted problem. *IEEE Intelligent Systems*, 25(3), 76–80.
- Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1), 1–167.
- Ma, S., Cui, L., Dai, D., Wei, F., & Sun, X. (2019). Livebot: Generating live video comments based on visual and textual contexts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 6810–6817.
- Meta. (2016). Taking into account live video when ranking feed [Accessed: 2022-1-30]. <https://about.fb.com/news/2016/03/news-feed-fyi-taking-into-account-live-video-when-ranking-feed/>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Mohammad, S., & Turney, P. (2010). Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, 26–34.
- Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word–emotion association lexicon. *Computational intelligence*, 29(3), 436–465.
- Mondovo. (2020). The Most Searched Words on Google – Top Keywords | Mondovo. <https://www.mondovo.com/keywords/most-searched-words-on-google>
- Mozilla/DeepSpeech. (2021). Retrieved April 4, 2021, from <https://github.com/mozilla/DeepSpeech>
- Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning. *ICML*.
- NVIDIA/NeMo. (2021). Retrieved April 4, 2021, from <https://github.com/NVIDIA/NeMo>
- Online, F. (2021). YouTube to stop showing ‘Dislike’ count on all videos; here’s why. <https://www.financialexpress.com/industry/technology/youtube-to-stop-showing-dislike-count-on-all-videos-heres-why/2367809/>
- Ortiz-Ospina, E. (2019). The rise of social media. <https://ourworldindata.org/rise-of-social-media#licence>
- Paivio, A. (2013). *Imagery and verbal processes*. Psychology Press.

- Parrott, W. G. (2001). *Emotions in social psychology: Essential readings*. psychology press.
- Pearl, L., & Steyvers, M. (2010). Identifying emotions, intentions, and attitudes in text using a game with a purpose. *Proceedings of the naacl hlt 2010 workshop on computational approaches to analysis and generation of emotion in text*, 71–79.
- Ping, Q. (2018). Video recommendation using crowdsourced time-sync comments. *Proceedings of the 12th ACM Conference on Recommender Systems*, 568–572.
- Plutchik, R. (1980). A general psychoevolutionary theory of emotion. *Theories of emotion* (pp. 3–33). Elsevier.
- Poria, S., Cambria, E., Bajpai, R., & Hussain, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion*, 37, 98–125.
- Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. (2011). The kaldi speech recognition toolkit [IEEE Catalog No.: CFP11SRW-USB]. *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.
- Proksch, S.-O., Lowe, W., Wäckerle, J., & Soroka, S. (2019). Multilingual sentiment analysis: A new approach to measuring conflict in legislative speeches. *Legislative Studies Quarterly*, 44(1), 97–131.
- Pundak, G., & Sainath, T. (2016). Lower frame rate neural network acoustic models.
- Pyinstaller 5.1. (2020). Retrieved May 22, 2022, from <https://pyinstaller.org/en/stable/>
- Rambocas, M., & Pacheco, B. G. (2018). Online sentiment analysis in marketing research: A review. *Journal of Research in Interactive Marketing*.
- RapidAPI. (2020). Youtube v3 api documentation. Retrieved December 28, 2021, from <https://rapidapi.com/ytdlfree/api/youtube-v31/>
- Restream Team. (2021). 56 stats for your live streaming strategy in 2022 [Accessed: 2022-1-30]. <https://restream.io/blog/live-streaming-statistics/>
- Roberts, R. C. (1988). What an emotion is: A sketch. *The philosophical review*, 97(2), 183–209.
- Sadoski, M., & Paivio, A. (1994). A dual coding view of imagery and verbal processes in reading comprehension.

- Saura, J. R., Palos-Sanchez, P., & Grilo, A. (2019). Detecting indicators for startup business success: Sentiment analysis using text data mining. *Sustainability*, *11*(3), 917.
- Streamlink. (2021). Streamlink 2.1.0 documentation. Retrieved March 25, 2021, from <https://streamlink.github.io/>
- Svendsen, T. (2004). Pronunciation modeling for speech technology. *2004 International Conference on Signal Processing and Communications, 2004. SPCOM'04.*, 11–16.
- Taizan-hokuto. (2021). Library for youtube chat. Retrieved March 24, 2021, from <https://github.com/taizan-hokuto/pytchat>
- Tampermonkey for Chrome. (2021). Retrieved April 4, 2021, from <http://www.tampermonkey.net>
- The Packer.com. (2019). The Packer Fresh Trends 2019. <http://digitaledition.qwinc.com/publication/?m=40749&i=577447&p=0>
- ThinkwithGoogle. (2016). Top four content categories on YouTube. <https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/top-content-categories-youtube/>
- Thomala, L. L. (2021). Bilibili: Average maus 2021. <https://www.statista.com/statistics/1109108/bilibili-average-monthly-active-users/>
- Tim-smart userscripts. (2021). Retrieved April 4, 2021, from <https://github.com/tim-smart/userscripts>
- Uechi, Y. (2021). Vtuber 500m: Large scale virtual youtubers live chat dataset. <https://github.com/holodata/vtuber-livechat-dataset>
- Van den Burg, G. J. J., & Williams, C. K. I. (2020). An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*.
- Vosk offline speech recognition api. (2020). Retrieved April 4, 2021, from <https://alphacephei.com/vosk/>
- Wang, L., Zhang, J., Wang, M., Tian, J., & Zhuo, L. (2020). Multilevel fusion of multimodal deep features for porn streamer recognition in live video. *Pattern Recognition Letters*, *140*, 150–157.
- Wang, W., Chen, J., & Jin, Q. (2020). Videoic: A video interactive comments dataset and multimodal multitask learning for comments generation. *Proceedings of the 28th ACM International Conference on Multimedia*, 2599–2607.

- Wang, Z., Zhou, J., Ma, J., Li, J., Ai, J., & Yang, Y. (2020). Discovering attractive segments in the user-generated video streams. *Information Processing & Management*, 57(1), 102130.
- Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301), 236–244.
- Wiebe, J., Bruce, R., & O’Hara, T. P. (1999). Development and use of a gold-standard data set for subjectivity classifications. *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, 246–253.
- Wu, B., Zhong, E., Tan, B., Horner, A., & Yang, Q. (2014). Crowdsourced time-sync video tagging using temporal and personalized topic modeling. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 721–730.
- Wu, Z., Zhou, Y., Wu, D., Zhou, Y., & Qin, J. (2019). Crowdsourced time-sync video recommendation via semantic-aware neural collaborative filtering. *International Conference on Web Engineering*, 171–186.
- Xian, Y., Li, J., Zhang, C., & Liao, Z. (2015). Video highlight shot extraction with time-sync comment. *Proceedings of the 7th International Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking*, 31–36.
- Yang, W., Wang, K., Ruan, N., Gao, W., Jia, W., Zhao, W., Liu, N., & Zhang, Y. (2019). Time-sync video tag extraction using semantic association graph. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(4), 1–24.
- Yang, W., Gao, W., Zhou, X., Jia, W., Zhang, S., & Luo, Y. (2019). Herding effect based attention for personalized time-sync video recommendation. *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 454–459.
- Youtube data api overview. (2020). <https://developers.google.com/youtube/v3/getting-started?hl=en#quota>
- Youtube live streaming ingestion protocol comparison. (2020). Retrieved April 4, 2021, from <https://developers.google.com/youtube/v3/live/guides/ingestion-protocol-comparison>
- Zucco, C., Calabrese, B., & Cannataro, M. (2017). Sentiment analysis and affective computing for depression monitoring. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 1988–1995.