Processing API Data as a Stream: A Report

Khuong Thanh-Gia-Hieu
Alex-Răzvan Ispas
Github: `https://github.com/ktgiahieu/largescaledistri_project`

Abstract

In this report, we present a comprehensive analysis of clustering Twitch chat messages from the most viewed streams across five categories using sentiment analysis and K-means clustering. Starting with the collection of chat messages through Spark Streaming, we preprocess and analyze the data to obtain sentiment polarity and subjectivity scores. We then apply the K-means clustering algorithm to group the chat messages based on these features. Our investigation covers various aspects of the clustering process, including data preprocessing, feature extraction, and model evaluation. A thorough comparison of the resulting clusters is conducted, highlighting the challenges in distinguishing the categories based on sentiment features alone.

1. Introduction

This project aims to analyze the Twitch chat messages from the most viewed streams of five categories: Fortnite, Just Chatting, Chess, Valorant, and Counter-Strike [5].

The analysis includes sentiment detection and clustering based on sentiment polarity and subjectivity. The project is implemented in Python, using the Twitch API and PySpark for data processing and analysis [9].

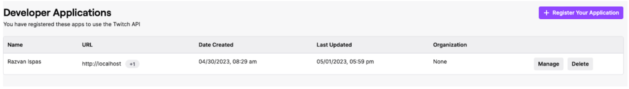
2. Methodology

2.1. Obtaining Access on Twitch Developers

In order to connect to the Twitch API, you will need to create an account on the Twitch developer platform and register an application to use the API, following these steps:

1. Create an account on the Twitch developer website.

2. Register for API access [8].



Developer Applications
You have registered these apps to use the Twitch API

Name	URL	Date Created	Last Updated	Organization	
Razvan Ispas	http://localhost +1	04/30/2023, 08:29 am	05/01/2023, 05:59 pm	None	Manage Delete

Figure 1. Register for API access

3. After your application is accepted, press "Manage" and set the URL to `localhost` with the port `3000`.

4. Enable two-factor authentication [3].

2.2. Connecting to Twitch API

To connect to the Twitch API, you will need the following credentials:

- Your Twitch nickname
- The application key
- Your secret key

The `connect_twitch_to_socket.py` script provides a user interface that guides you through the process of adding your credentials. After completing these steps for the first time, your credentials will be saved in `twitch_info.json` and `oauth-keys.json` [10].

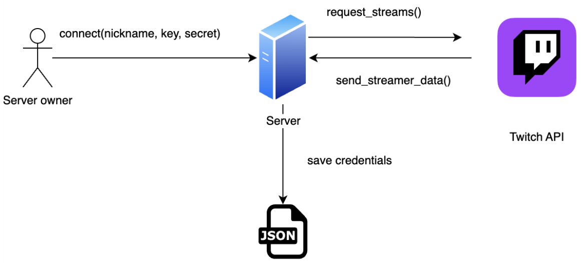


Figure 2. Twitch API connection diagram.

The process then connects to the Twitch API and retrieves the data for the most viewed stream of each of the five categories. It creates a socket for each stream's chatroom and listens for incoming chat messages. The script also creates a listening socket at localhost:9999, which waits for a Spark Streaming client to connect before broadcasting the concatenated chat messages to the client.

2.3. Analyzing Retrieved Data

To connect a listener to the server, use the `twitch_chat_analysing.py` script. Once connected, the listener retrieves the data stream via the socket and begins sentiment detection with a window size of 60. It then performs the following analyses on the chat messages:

1. Clean the Twitch chat messages by removing elements that do not represent words [4].

2. For each category, use the TextBlob library to detect sentiment polarity and subjectivity [7].

3. Merge the game, sentiment (polarity and subjectivity), and its vectorized version for each entry in the stream [2].

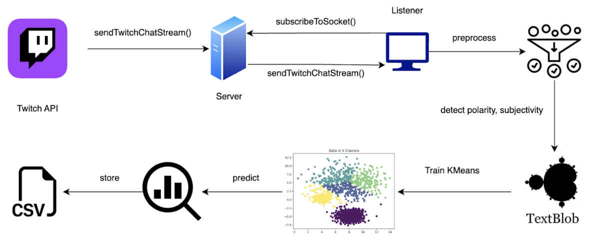


Figure 3. Data processing procedure of the project.

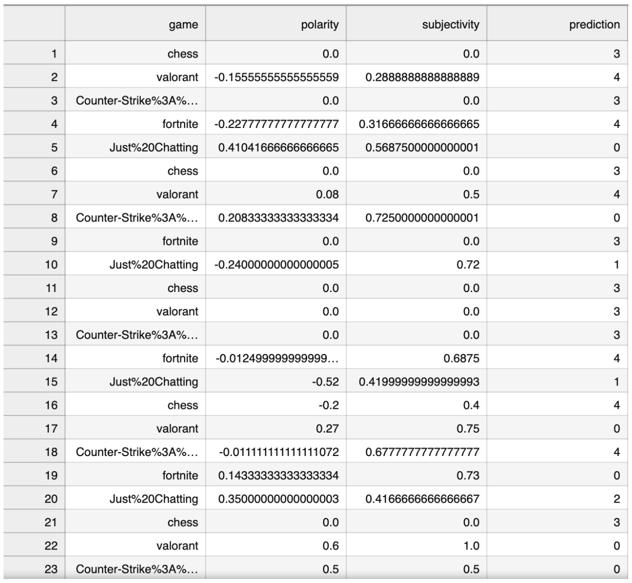
2.4. Category Clustering

After collecting sentiment data for each entry, the polarity and subjectivity values are used to train a KMeans model in the streaming context. The K-means clustering algorithm is applied to the vectorized sentiment data to group the chat messages into five distinct clusters [1]. The model is configured with k=5 (the same number of centroids as the number of categories) and a decay factor of 0.1 to update the cluster centroids as new data arrives. The model is then used to predict the cluster for each entry. Finally, all data is saved in a DataFrame, which includes the following:

- Sentiment polarity and subjectivity values

- KMeans cluster predictions

- Merged game, sentiment, and vectorized data for each entry [6].



	game	polarity	subjectivity	prediction
1	chess	0.0	0.0	3
2	valorant	-0.15555555555555559	0.2888888888888889	4
3	Counter-Strike%3A%…	0.0	0.0	3
4	fortnite	-0.22777777777777777	0.31666666666666665	4
5	Just%20Chatting	0.41041666666666665	0.5687500000000001	0
6	chess	0.0	0.0	3
7	valorant	0.08	0.5	4
8	Counter-Strike%3A%…	0.20833333333333334	0.7250000000000001	0
9	fortnite	0.0	0.0	3
10	Just%20Chatting	-0.24000000000000005	0.72	1
11	chess	0.0	0.0	3
12	valorant	0.0	0.0	3
13	Counter-Strike%3A%…	0.0	0.0	3
14	fortnite	-0.012499999999999…	0.6875	4
15	Just%20Chatting	-0.52	0.41999999999999993	1
16	chess	-0.2	0.4	4
17	valorant	0.27	0.75	0
18	Counter-Strike%3A%…	-0.011111111111111072	0.6777777777777777	4
19	fortnite	0.14333333333333334	0.73	0
20	Just%20Chatting	0.35000000000000003	0.4166666666666667	2
21	chess	0.0	0.0	3
22	valorant	0.6	1.0	0
23	Counter-Strike%3A%…	0.5	0.5	0

Figure 4. Example DataFrame of the predictions from K-means clustering

2.5. Real-time visualization

To see a demo of the real-time visualization, visit our Github repo. To visualize the clustering results in real-time, we implement an asynchronous Python script in the `visualization.ipynb` notebook, using Matplotlib and Seaborn libraries [9]. The script reads the updated ’output.csv’ file over time and generates a scatter plot of the chat messages’ sentiment values.

The real-time visualization allows for a better understanding of the clustering results and how they change over time[3].

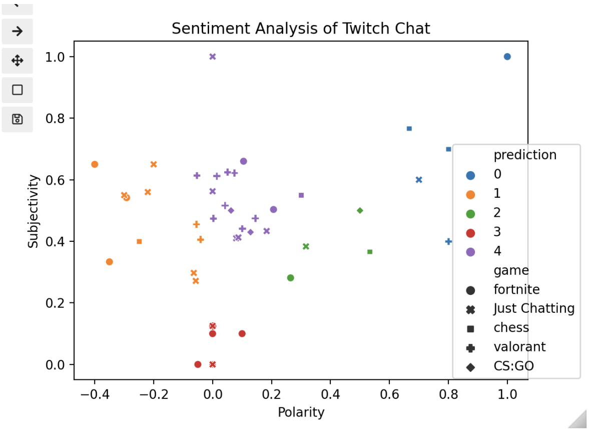


Figure 5. Visualization of the clusterings.

3. Results and Discussion

After running the analysis, it was observed that the clusters learned by the K-means model did not clearly differentiate between the categories. The sentiment polarity and subjectivity were found to be quite similar across the different categories, leading to overlapping clusters. This can be attributed to the following factors:

- Shared sentiment patterns across categories: Users express emotions similarly regardless of the content they are watching.
- Noise in chat data: Emotes, abbreviations, and non-standard language usage affect sentiment analysis results.
- Limited features for clustering: Only sentiment polarity and subjectivity are used, which may not be sufficient to differentiate categories.

4. Conclusion

This project aimed to analyze and cluster Twitch chat messages from the most viewed streams of five categories using sentiment analysis and K-means clustering. Although the clusters learned by the model did not clearly differentiate between the categories, the project provided insights into the challenges of clustering chat data and really helps us understand the procedure of working with Spark Streaming and APIs as a whole.

References

[1] Learning long-term dependencies with gradient descent is difficult.

[2] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC Press, 2014.

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, volume 96, pages 226–231, 1996.

[4] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[5] Tamal Lahiri, Niloy Ganguly, and Pawan Goyal. Twitching the right way: A snapshot of the twitch streaming ecosystem. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 66–77, 2018.

[6] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[7] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 63–70, 2002.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, volume 57, pages 61–66, 2010.

[10] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, and Ali ... Ghodsi. Apache spark: A unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.