A content-based filtering recommender system for the livestream setting using community detection

Zach Champion

1 ABSTRACT

Video streaming websites rely heavily on recommender systems for viewer retention. Livestreaming sites in particular face unique challenges due to the time-limited nature of their content. To make matters worse, they face extreme data sparsity in how their content is defined, relying largely on the broadcast categories used by different broadcasting channels. Relative to other aspects of these platforms, recommender systems and the applicability of popular techniques to this paradigm is relatively under-studied. In particular, content-modeling based approaches that make use of community detection haven't been explored.

In this project I compare the performance of several different community detection techniques on the task of powering a content-based recommender system. I propose a pipeline that detects communities among channels on the Twitch platform based on the similarity of their content, draws recommendation candidates from these communities based on user viewing history, and produces a final weighted ranking using a channel quality metric. In the course of developing this pipeline I assess the performance of two different content modeling approaches and three community detection methods, and compare the final ranking metric with a probabilistic ranking approach.

2 INTRODUCTION

Recommender systems are a driving force of audience growth on video streaming platforms. Even as long ago as 2018, YouTube's product chief reported that 70% of video views on the website resulted from suggested videos[1]. This is especially apparent on platforms such as Amazon's Twitch that specialize in real-time broadcasting, where available content is not permanent and user attrition can result from a poorly timed visit. The ability to present an appealing list of suggested channels is critical to viewer retention, but the classic problem of identifying and ranking recommendations is complicated

[1]"YouTube's AI is the puppet master over most of what you watch," Joan Solsman, CNET [Online].

by these very same quirks of the livestreaming setting.

Twitch has its origins in gaming content, which still engages a large portion of its user base, but now includes diverse categories including talkshows, sports, cooking, fitness, music, art, and software development. Particular video games, or any of these non-gaming activities, are considered categories on the platform and used to label and sort individual broadcasts, and a broadcast can include any number of categories throughout its duration. Data on the full number of categories included in each broadcast is recorded and accessible through Twitch's API. This results in a rich history of a given channel's past activity across different categories, providing a basis to compare channels that engage in similar content.

Community detection, the study of node connection density in a network, is a popular technique to augment recommender systems in a variety of settings [1], and is frequently cited as a means to improve recommender accuracy and efficiency [2]. However, in the literature on the video livestream setting, most network analysis of this type has been dedicated to understanding the platform's social dynamics [3][4][5], livechat behavior [6][7], and spread of information [8][9]. The work on livestream recommender systems is so far scarce, and there has not yet been a study on the effectiveness of community detection techniques in this data paradigm.

Classical recommender systems fall into three main categories, each based on the involvement of "users" (customers, viewers, or other would-be consumers of a recommendation), "items" (the catalogue of products or content being recommended), or both. These categories are called collaborative filtering, content-based filtering, and hybrid filtering, respectively [10]. Community detection techniques have been applied to all three approaches in past work [1].

In the livestream setting, the system's items are typically considered to be channels that launch broadcasts rather than individual broadcasts [11]. This is due to the extremely short lifespan of a broadcast, which would make the already sparse domain of online interactions much sparser. Channels, on the other hand, provide much more historical data to learn from and use to model both user and item similarities. As Rappaz et. al. point out, however, the channel-as-item paradigm isn't without its caveats. In this paradigm, individual users consume a given item repeatedly (i.e., they watch the same channel on multiple different occasions). This gives all items a high degree of repeat consumption, and creates an unusual scenario in which evaluation necessarily involves repeat exposure to the training data:

> This differs from traditional settings where positive interactions included in the training set are discarded from the testing set [...] When splitting our dataset in the temporal dimension, around 65% of user-item interaction observed in the testing period are also present in the training set. [11]

This project explores one method to turn this peculiarity into an advantage in order to produce accurate and personalized livestream recommendations. Twitch channels are first clustered into communities based on the similarity of their broadcast category history and language. Then, given a user's past viewing history over some arbitrary period, initial recommendation candidates are generated from the community members of each of their viewed channels. These candidates are weighted by the proportion of viewing time the queried channel took up in that user's history, and then either ranked into a final set of recommendations by a channel quality metric or drawn with these weighted probabilities. I compare the performance of different combinations of these techniques -- distance metrics, community detection techniques, and ranking approaches -- based on recommendation accuracy, network coverage, and personalization.

To perform this evaluation, I've collected a year of historical broadcast data from a sample of active streamers in a variety of languages, and made use of an existing benchmark set of user viewing history data.

3 RELATED WORK

The leading publication on the subject of livestreaming recommenders is by Rappaz et. al. [11], which introduced many of this setting's peculiarities to the literature. The main emphasis of this work was addressing the issues of repeat consumption and inconsistent item availability in the livestreaming paradigm. Specifically, they argue, traditional assumptions about implicit negative and positive interactions suffer in this setting. An implicit interaction is any that is passive on the part of the user, such as viewing time, as compared with explicit interactions like qualitative ratings, purchases, or other engagements. A negative interaction in video viewing settings is the implied choice a user makes not to watch something in favor of other content. However, a user viewing a given livestream may not accurately reflect their preferences if all of their preferred channels are offline at that time. To that end, the authors devised a self-attention model that included a pre-computed availability likelihood for each channel in its training data. Also encoded in the training data is a distinction between novel and repeat interactions, along with an indicator of recency. They also performed preliminary experiments demonstrating the limitations of existing recommendation methods on livestream data. Item relevance is used to narrow the search space for a given user in the same way community detection might, but takes up a small part of their solution.

4 DATA

In this section I describe my data collection and preparation. This project made use of two datasets: historical broadcast data collected for ~14,000 channels, and user viewing history data published by Rappaz et. al. [11][13].

The historical broadcast data was collected manually by scraping a web-hosted database formed from frequent Twitch API calls over a period of eight years[2]. A channel's audience and schedule statistics as well as a list of all of its broadcasts with categories and duration was collected from a sample of the most popular channels in the ten most prevelant languages on Twitch. Search results were sorted by follower count and covered the period from March 2022 - March 2023. In order to be included, the channel had to have streamed at least once in that time period. Approximately 1,000 channels were collected for each of Arabic, French, German, Italian, Japanese, Korean, Portuguese, Russian, and Turkish, plus 2,000 from Spanish and 3,000 from English, which together represent about 70% of accounts actively broadcasting on the platform[3]. Languages were scraped separately to allow collection of more total channels. This was due to a limitation of the host website, which limits searches to 5,000 results; the feasible maximum was further limited by search results capped at 100 rows per page.

The broadcast data were then aggregated into a sparse $m \times n$ matrix, where m is the number of channels collected and n is the number of categories represented in their combined broadcast history. Each row contains the proportion of total broadcast time that channel spent on each category represented in the data. The audience and schedule statistics were collected in a separate matrix.

The second dataset used is the Twitch dataset published in Julian McAuley's "Recommender Systems and Personalization Datasets" [13]. This represents the viewing history of 15.5 million users recorded over a period of 43 days in July 2019, or a reduced benchmarking set of 100,000 users. This project uses the reduced benchmarking set. The set is structured as a series of time-ordered viewing sessions, providing the channel name and start and end time of each session rounded to 10-minute intervals. I split this data temporally into training and testing sets, representing approximately the first 30 and last 13 days of the sampling period, respectively. Each

[2] Sullygnome.com, "Twitch statistics and analytics." [Online]

[3] Sullygnome.com, "Twitch channel summary statistics." [Online]

split set was separately summarized into sparse matrices representing the proportion of viewing time each user spent on every channel in the dataset.

Finally, prior to pre-processing, the two datasets (channel broadcast and user viewing histories) were cross-referenced to include only channels present in both datasets. Any users with zero time spent watching the remainder were removed prior to training. The final size of the broadcast history following this intersection was 5073 channels and 16,553 categories. 4,422 channels were present in the user training data and 4,897 in the user test data. This was done to reduce the impact of the two sets being time-disjoint. This issue is discussed further in Section 8.

5 OVERVIEW OF COMMUNITY DETECTION METHODS

This project compares two approaches to similarity scoring and three approaches to community detection. Different combinations of these approaches were evaluated with both popular unsupervised community detection performance metrics as well as the accuracy measures of the final recommender system.

5.1 Similarity Metrics

I first compared the effectiveness of discrete and continuous representation of item features, and similarity metrics corresponding to these representations. Item features consist of the proportion of each channel's total broadcast time dedicated to the different categories present in the dataset. Continuous representation of this data weights each feature, while discrete representation reduces it to binary indicators. Similarity metrics appropriate for each of these representations were used to weight adjacency matrices, representing undirected graphs G_c and G_d, respectively.

To measure similarity with continuous features, a squared euclidean similarity metric was used pairwise over the item matrix M to determine the weight of each edge $E_{(u,v)}$.

$$E_{(u,v)} = \frac{1}{1 + ||u - v||_2^2} \; \forall (u, v) \in M$$

Equation 1.

To measure similarity with discrete features, the Jaccard similarity coefficient was used pairwise over the item matrix M.

$$E_{(u,v)} = \frac{|u \cap v|}{|u \cup v|} \; \forall (u, v) \in M$$

Equation 2.

Each adjacency matrix was tuned for an appropriate value of ε to prune edge connections to an approximation of the underlying network structure. After initial trials this was determined to be a graph density of ~60%, measured by $\frac{|E|}{|V|(|V| - 1)/2}$

.

$$\text{Coverage}(C_1, \ldots, C_k) = \frac{1}{|E|} \sum_{u,v \in E} \mathbb{I}[z_u^T z_v > 0]$$

Equation 4.

$$\text{AvgClustCoef}(C1, \ldots, C_k) = \frac{1}{\sum_i |C_i|} \sum_i \frac{\text{\# existing triangles in } \mathrm{C}_i}{\text{\# possible triangles in } \mathrm{C}_i} |C_i|$$

Equation 5.

5.2 Community Detection Methods
Next, three community detection techniques were compared on each of the resulting graphs.

Spectral clustering. This method identifies connected sub-graphs of G using eigendecomposition of the graph Laplacian matrix. If a graph has k tightly connected components with sparsely connected edges between them, then this decomposition will produce k eigenvectors with corresponding eigenvalues close to 0. Because prior knowledge of the number of components is assumed, this method was tuned on the parameter k from a range of 50 to 1000 communities.

Louvain method. This greedy optimization technique identifies communities that maximize the modularity Q of a graph, a measure of the internal edge density of the graph's communities relative to links between communities. Its objective function is:

$$argmax_c \; \frac{1}{2m} \sum_{(u,v)} \left[A_{(u,v)} - \frac{k_u k_v}{2m} \right] \delta(c_u, c_v)$$

Equation 3.

where m is the sum of all graph edge weights, k represents the sum of each node's edge weights, and $\delta(\cdot)$ is an indicator function that returns 1 when node u and v are in the same community.

OPTICS. Ordering Points To Identify the Clustering Structure, an algorithm which linearly orders the points in the dataset such that the closest points spatially become neighbors. A cluster is specified to contain at least k points to be valid. This method was tuned on the minimum points, from a range of 3 to 7.

The community performance metrics for each of these similarity-detection method pairings is summarized in Table 1. The methods were compared in terms of modularity (defined above), coverage, average clustering coefficient, and number of communities. Coverage indicates the percentage of the graph's edges explained by a community.Nodes assigned to different communities that share an edge lower this score [14] (Eq. 4).

Method	Distance	Parameter	Modularity	Coverage	Clustering Coefficient	# Clusters	# Cluster/m
Louvain	Jaccard		0.6309	0.7972	0.0053	2185	2.292
Spectral	Jaccard	210	0.1969	0.3258	0.0113	210	23.8476
Louvain	Euclidean		0.1798	0.3136	0.0064	121	41.3884
Spectral	Jaccard	310	0.1794	0.1865	0.0136	310	16.1548
Spectral	Jaccard	410	0.1714	0.2113	0.0187	410	12.2146
Spectral	Jaccard	510	0.1676	0.1782	0.0204	510	9.8196
Spectral	Jaccard	610	0.1648	0.1919	0.0228	610	8.2098
Spectral	Jaccard	710	0.1536	0.1755	0.0233	710	7.0535
Spectral	Jaccard	110	0.148	0.5793	0.0084	110	45.5273
Spectral	Jaccard	810	0.1457	0.1642	0.0255	810	6.1827
Spectral	Jaccard	910	0.144	0.1534	0.0249	910	5.5033
Spectral	Jaccard	1010	0.1397	0.142	0.0251	1010	4.9584
OPTICS	Jaccard	3	0.0618	0.4574	0.1061	164	30.5366
OPTICS	Jaccard	4	0.0615	0.5269	0.0648	83	60.3373
OPTICS	Jaccard	5	0.0543	0.5865	0.0469	53	94.4906
Spectral	Euclidean	210	0.0507	0.3362	0.011	210	23.8476
OPTICS	Jaccard	6	0.0483	0.636	0.0402	38	131.7895
Spectral	Euclidean	410	0.0385	0.2526	0.0183	410	12.2146
Spectral	Euclidean	510	0.0366	0.1842	0.0208	510	9.8196

Table 1: A comparison of community detection performance metrics during parameter tuning. Results include three detection methods in combination with two similarity scores.

The average clustering coefficient is a combination of each community's coefficient weighted by community size. The coefficient measures the number of closed (i.e., fully connected) triplets of nodes within the community relative to the total number of triplets [14] (Eq. 5).

The number of communities is a consideration particular to the goals of community detection. In this case, having an extremely small number of communities relative to the number of channels would provide poor recommender results. If a scored ranking metric were employed, then all users would be very likely to receive the same set of recommendations regardless of what they watched previously. If final selections are drawn randomly from the community candidates, then performance would differ very little from a random sampling of the whole channel search space. For this reason I favored methods that resulted in at least $m/5$ communities.

At this stage, community detection performance was used to determine the best parameterization for spectral clustering and OPTICS. The differing results between different methods and similarity metrics is revisited in Section 7 in the context of overall recommender performance. Spectral clustering was determined to offer the best balance of scores with 210 clusters when trained with the Jaccard coefficient, and 410 when using Euclidean similarity. Optics performed best with three minimum points using Jaccard, and five using Euclidean.

6 RANKING METRIC

In addition to collaborative filtering, content filtering, and hybrid filtering, a number of techniques have been explored to weigh item quality as a consideration in recommendation filtering [15][16]. A common challenge in this approach is effectively measuring item quality. In the livestreaming setting, there are several quantitative measures available that approximate a channel's quality in the form of sustained audience growth and account activity.

The goal of a recommendation system is to predict an item's inclusion in a user's future behavior. In the livestreaming setting in particular, predicted future items can include both interactions with novel items and repeat interactions with items already in the user's training data. Items that garner repeat and regular viewership in general can therefore be considered to be of high quality. One indirect measure of this is a channel's follower count, which indicates the number of users who have opted to be notified of the channel's live status (either through push notifications or inclusion on their Twitch home page). This can be viewed as an explicit indication of interest in repeat viewing. This measure alone doesn't tell the full story, however. Many anomalous accounts have a high number of followers while seldom streaming, or with a relatively low number of average concurrent viewers. This phenomenon results from channels that, for example, have independent fame on other platforms but do not often use Twitch, or channels that have purchased followers through illicit means. Either of these scenarios could skew a recommender's overall quality by including channels with a low probability of being live and with an overestimated likelihood to retain the viewer.

Therefore, I propose a weighted quality metric that accounts for channel momentum, average viewership, and level of activity. Momentum is measured by follower gain during the sample period Δf rather than total followers at the time of training, while average viewership and level of activity are measured simultaneously by the channel's "Watch time" statistics, W. This roughly equates to the total minutes streamed mutliplied by the channel's average viewership during the period under consideration. The final quality metric is their follower gain divided by watch time, $\frac{\Delta f}{W}$. In addition to providing a more detailed snapshot of a channel's level of activity and ability to retain viewership, this metric also hedges against popularity bias when compared with scoring a channel on followership alone.

To evaluate the effectiveness of this metric, recommenders were trained using both this ranking approach and sampling from a uniform distribution of candidates.

		Probability-Based				Ranked Score-Based			
Method	Metric	ACC	Novel ACC	Repeat ACC	Personalization	ACC	Novel ACC	Repeat ACC	Personalization
Spectral	Euclidean	1.14%	0.32%	1.59%	16.51%	0.75%	0.15%	1.04%	22.64%
	Jaccard	1.03%	0.33%	1.4%	16.38%	0.71%	0.14%	0.98%	22.47%
OPTICS	Euclidean	1.61%	0.27%	2.55%	16.32%	1.27%	0.17%	2.04%	21.69%
	Jaccard	0.93%	0.25%	1.35%	16.63%	0.69%	0.11%	1.03%	23.35%
Louvain	Jaccard	2.88%	0.29%	4.79%	15.13%	2.52%	0.11%	4.28%	24.51%
	Euclidean	1.42%	0.34%	2.14%	16.99%	1.02%	0.16%	1.58%	17.94%
Random sample from full item space		0.10%	0.10%	0.09%	16.82%				

Table 2: A comparison of mean accuracy performance between recommenders built using different combinations of community detection, similarity metric, and ranking approach.

7 RECOMMENDER EVALUATIONS

7.1 Training Method

Following parameter tuning, the final community labels produced by each method pairing were used to train and evaluate twelve different recommender configurations: all permutations of the three community detection techniques, the two similarity metrics, and a score-based and proability-based ranking approach.

The training data split from the user viewing history dataset was used to generate a set of recommendations for each user. For every channel in their viewing history, a set of candidates was formed based on the entire contents of each channel's community. The candidates drawn from each community were weighted by the proportion of the user's viewing history the queried channel took up.

These weights were applied either to the channels' quality metric, or to their probability distribution. In the final step, a list of five results were taken from the top ranked channels or randomly drawn.

7.2 Performance Metrics

Each recommender configuration was compared on the following:

- The overall accuracy of a set of recommendations was measured as the percentage of the user's test data that appeared in the set. I.e., if 4 of the recommendations are present in a test point of length 8, the system achieves an accuracy of 50%.
- Accuracy is then separately measured as a percentage of new channels predicted, i.e. channels not present in that user's training data, and repeat channels predicted. This evaluates the model's ability to both recommend appealing repeat content and introduce new interests to users.
- Finally, the recommenders are scored on their degree of personalization. This measures the similarity between recommendations given to all users; a higher score indicates more diversity of recommendations, while a lower score indicates a larger number of repeat recommendations.

7.3 Overall Results

Overall, the system's accuracy performance is disappointing. The results follow the expected pattern of performing better on repeat recommendations than both novel recommendations and the overall average, but a maximum overall accuracy of 2.88% indicates that there is a key flaw in the logic of the pipeline. Compared with a random sample of the entire item space, however, the two sampling approaches do appear to offer non-zero value.

It's possible that issues with the data collection contributed to this -- see Section 8.1 for more discussion -- but, the final stages of sampling and ranking from candidate recommendations is no doubt in need of rework. Indeed, it may also benefit from better incorporation of existing work on the role community detection should play in the pipeline [1].

7.4 Discrete vs. Continuous Similarity Scores

This split ended up having a significant impact on the performance of the community detection techniques (Table 1), but a much more ambiguous effect on the overall performance of the recommender. While the best-scoring recommender overall did use the Jaccard similarity metric and discretized item data, the accuracy differences are not large enough to be significant, and the two worst performing overall use the same metric.

The community detection methods all unambiguously performed best on the discretized data, however, which produced a much sparser adjacency matrix and may have therefore had a clearer community structure.

7.5 Performance of Community Detection Techniques

The Louvain method outperformed both other techniques in both clustering performance and overall recommender accuracy. Given that it optimizes cluster assignments based on modularity, it's unsurprising to see it perform among the best on this metric. It also provides good edge coverage, but would seem to produce arguably too many clusters overall, with an average of two channels per cluster.

However, that high number of clusters did lead to the best recommender accuracy. The difference is not significant enough to consider important, but in a pipeline that doesn't have poor overall performance this is a result worth further investigation.

7.6 Random Sampling Versus Scored Metric

It's clear from these results that channel quality scoring does not improve recommender performance -- it gives worse results than a weighted uniform distribution. However, given the overall poor performance of both ranking techniques, the issue with the pipeline lies deeper than this final step.

One area the ranked approached outshone random sampling was in recommendation personalization. This also runs counter to my expectations -- given multiple users draw from the same communities, I assumed a ranked score approach would result in more frequently returning overlapping results than a random draw.

8 DISCUSSION

8.1 Challenges and Limitations

The data used for this project suffers from one significant weakness that can easily be remedied in a followup trial, likely resulting in improved results. Specifically, the user interaction and historical broadcast datasets are disjoint in time (July 2019 vs. 2022-23). This was a result of discovering the user interaction dataset late in the data collection process; initially it was assumed that the historical broadcast data would be the only dataset available for the project.

This issue results in poorer recommender performance because many channels don't overlap between the two time periods -- ultimately removing them from the training data -- and the differing size and broadcasting habits of channels between the two periods will skew results. This issue was mitigated slightly by a) collecting channels sorted by follower count, resulting in a higher-than-average number of channels with broadcasting history that overlaps with 2019 and b) removal of non-intersecting channels from the training data.

The solution is straightforward: randomly sample a sufficient quantity of channels from the list of all those present in the user interaction dataset, and collect stream data from the latter's sample period of July 2019.

8.2 Opportunities for Further Work

Additional benchmarks for performance comparison. Arguably the biggest weakness of this project is a lack of ground truth comparison for the recommender's performance in the form of similar measures using established techniques on the same data. Because of this it's unclear at which point in the pipeline performance suffers most -- it could be that the issues introduced by the data collection would spoil outcomes in a variety of existing recommenders, or they could reveal that there is a fundamental flaw in the design of my pipeline.

I expect the greatest source of issue to come from the final aggregation of candidates and their ranking. A more robust sampling approach from each community under consideration could further tune the candidates to the user in question. For example, determining candidates with the highest similarity score against the whole set of channels in that user's history instead of simply weighting and ranking (or sampling from) the entire set.

Finally, the measurement of more non-accuracy performance measures could shed light on the strenghts of the system, such as diversity, novelty, and surprise. Zhou et. al. argue:

> The focus on similarity is compounded by the metrics used to assess recommendation performance. A typical method of comparison is to consider an algorithm’s accuracy in reproducing known user opinions that have been removed from a test dataset. An accurate recommendation, however, is not necessarily a useful one: real value is found in the ability to suggest objects users would not readily discover for themselves, that is, in the novelty and diversity of recommendation. [17]

<u>More realistic data insights</u>. In a real-world scenario, Twitch has access to substantial additional data to that described in this project. Users provide explicit feedback in the form of following, donating to, and paying for a subscription to a channel, as well as chatroom engagement during broadcasts. Additional insight can be gained from a user's viewing duration prior to taking any of these actions. Broadcasting channels would also provide more robust information as items if data on the site's collaborative features were available. Explicit connections exist in the form of channel hosts and raids -- wherein one channel broadcast's another's content to their viewers while offline -- use of the shout-out feature in the channel's chatroom, multi-streams broadcast together, and shared team membership.

<u>Better channel quality measures</u>. An important result of the above could be a better picture of what makes a quality channel -- not just one that gains audience over time and streams often, but which regularly gains paid subscribers and has a highly engaged chatroom. A better quality measure could prove to make the scoring approach more viable.

<u>Combination with existing studies on item availability</u>. The techniques outlined here do not address the crucial impact item availability has on a livestream recommender system's assumptions. In weighting a user's recommendation rankings by the proportion of view time taken up by each channel in their history, we're allowing the assumption of equal item availability to sneak in under the guise of treating viewing history as an accurate reflection of the user's real preferences. In reality, it may be the case that any given user may have preferred to devote 100% of their viewing time to a single channel that was only infrequently available. The self-attention approach devised by Rappaz et. al. to address this issue can be used as the final piece of the recommder pipeline rather than a scored ranking or weighted probabilistic approach. Conversely, an adaptation of the authors' availability encoding scheme could be applied to methods outlined in this project.

9 CONCLUSION

In this project, I developed a content-based recommendation system based on community detection among livestreaming channels on Twitch. The system was evaluated using several configurations of similarity metrics and community detection techniques, and incorporated a weighted item quality metric for ranking recommendations. While the results are an improvement over simple random sampling, further analysis is needed to address accuracy weaknesses.

10 REFERENCES

[1] F. Gasparetti, G. Sansonetti, A. Micarelli. "Community detection in social recommender systems: a survey." *Appl Intell*, vol. 51, pp. 3975–3995 (Nov. 2020). doi: https://doi.org/10.1007/s10489-020-01962-3.

[2] M. Azaouzi, D. Rhouma, L.B. Romdhane. "Community detection in large-scale social networks: state-of-the-art and future directions." *Social Netw Anal Min,* vol. 9, iss. 23. 2019. doi: https://doi.org/10.1007/s13278-019-0566-x

[3] J. Cai, C. Guanlao, D. Y. Wohn. "Understanding Rules in Live Streaming Micro Communities on Twitch." Jun. 2021. Presented at the ACM Intl. Conference on Interactive Media Experiences. doi: https://doi.org/10.1145/3452918.3465491

[4] Z. Lu, M. Annett, D. Wigdor. "Vicariously Experiencing it all Without Going Outside: A Study of Outdoor Livestreaming in China." (Nov. 2019) *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, pp. 1-28. doi: https://doi.org/10.1145/3359127

[5] B. C. B. Churchill and W. Xu, "The Modem Nation: A First Study on Twitch.TV Social Structure and Player/Game Relationships," 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), Atlanta, GA, USA, 2016, pp. 223-228, doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.43.

[6] R. Koskimaa, T. Välisalo, J. Lindroos. "Understanding Twitch Esports Communities through Livestream Chat Analysis." Nov. 2022. Presented at the Esports Research Network Conference, Jönköping, Sweden. pp. 139-141. [Online]. Available: https://esportsresearch.net/wp-content/uploads/2023/01/ERNC22_Jonkoping_Book_of_Abstracts.pdf#page=139

[7] F. Barbieri, L. Espinosa-Anke, M. Ballesteros, J. Soler-Company, H. Saggion. "Towards the understanding of gaming audiences by modeling Twitch emotes." (Sep. 2017) Presented at the Third Workshop on Noisy User-generated Text

(W-NUT 2017), Copenhagen, Denmark. Stroudsburg (PA): ACL; 2017. p. 11-20. [Online]. Available: https://repositori.upf.edu/handle/10230/33289

[8] N. Ruiz-Bravo, L. Selander, M. Roshan. "The Political Turn of Twitch – Understanding Live Chat as an Emergent Political Space." Presented at the 55th Hawaii International Conference on System Sciences (Jan. 2022). uri: http://hdl.handle.net/10125/79723

[9] V. Diwanji, A. Reed, A. Ferchaud, J. Seibert, V. Weinbrecht, N. Sellers. "Don't just watch, join in: Exploring information behavior and copresence on Twitch." *Computers in Human Behavior*, vol. 105, Apr. 2020. doi: https://doi.org/10.1016/j.chb.2019.106221

[10] P. B. Thorat, R. M. Goudar, S. Barve. "Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System." *Intnl. Journal of Computer Applications,* vol. 110, no.4. Jan., 2015. doi: 10.5120/19308-0760.

[11] J. Rappaz, J. McAuley, K. Aberer. Recommendation on Live-Streaming Platforms: Dynamic Availability and Repeat Consumption. Presented at RecSys 2021. [Online]. Available: https://cseweb.ucsd.edu/~jmcauley/pdfs/recsys21b.pdf.

[12] X. Su et al., "A Comprehensive Survey on Community Detection With Deep Learning," in *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2021.3137396.

[13] J. McAuley. 2021. "Recommender Systems and Personalization Datasets: Twitch," RecSys. [Online]. Available: https://cseweb.ucsd.edu/~jmcauley/datasets.html#twitch

[14] O. Shchur, S. Gunnemann. "Overlapping Community Detection with Graph Neural Networks." Presented at the 1st Intnl. Workshop on Deep Learning on Graphs (Sep. 2019). doi: https://doi.org/10.48550/arXiv.1909.12201

[15] S. Cho, M. Lee, J. Kim, B. Kim and E. Choi. "A Novel Weight for Recommendation: Item Quality." Presented at the 2008 International Conference on Computational Sciences and Its Applications, Perugia, Italy. pp. 39-46. doi: 10.1109/ICCSA.2008.58.

[16] Y. Guan, S. Cai, M. Shang. "Recommendation algorithm based on item quality and user rating preferences." *Front. Comput. Sci.*, vol. 8, pp. 289–297 (Dec. 2013). https://doi.org/10.1007/s11704-013-3012-7

[17] T. Zhou, Z. Kuscsik, J. Liu, M. Medo, J.R. Wakeling, Y. Zhang. "Solving the apparent diversity-accuracy dilemma of recommender systems." *Applied Physical Sciences*, (Feb. 2010). doi: https://doi.org/10.1073/pnas.1000488107