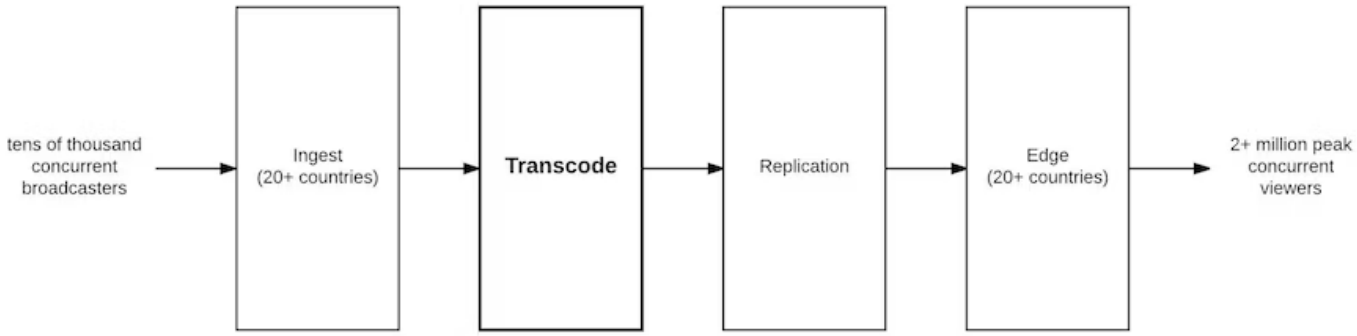
CSC 461 Multimedia Systems

Case Study Topic: Twitch

Joshua Blanch, Sam Goyal and Baraa Daoud

Twitch Compression and Transcoding

Twitch, akin to various other live streaming services, receives live stream uploads through the Real-Time Messaging Protocol (RTMP) from its broadcasters. As learned in class, RTMP is a protocol specifically designed for streaming video and audio on the internet, primarily utilized for point-to-point communication. In the context of Twitch's operational requirements, considerations must be made regarding how to efficiently scale to accommodate millions of viewers. This necessitates the adoption of HTTP Live Streaming (HLS), an HTTP-based media streaming communications protocol widely employed by most video websites.The transcoder module is responsible for converting incoming RTMP streams into the HLS format with multiple variants, such as 1080p and 720p. These variants offer varying bitrates, enabling viewers with different levels of download bandwidth to consume live video streams at optimal quality for their respective connections [1].



CDN of Twitch Figure 1 [1]

As depicted in figure 1, this provides a high-level overview of Twitch's Content Delivery Network. The process involves input broadcasts from streamers, subsequent transcoding, and replication of content to cater to millions of viewers on the platform. The transcoder receives an H.264 RTMP stream from streamers and outputs a transmuxed version of the H.264 in HLS [1].

FFmpeg Demo

FFmpeg, a widely used open-source software project, plays a crucial role in this ecosystem. It is designed to record, process, and stream video and audio. Frequently deployed by cloud encoding services for file transcoding, FFmpeg can also be employed for live stream transmuxing and transcoding processes. Twitch used this open-source library initially before switching to their in-house tool, so it is important to look into what were the downsides of FFmpeg and what Twitch is doing with their encoding.

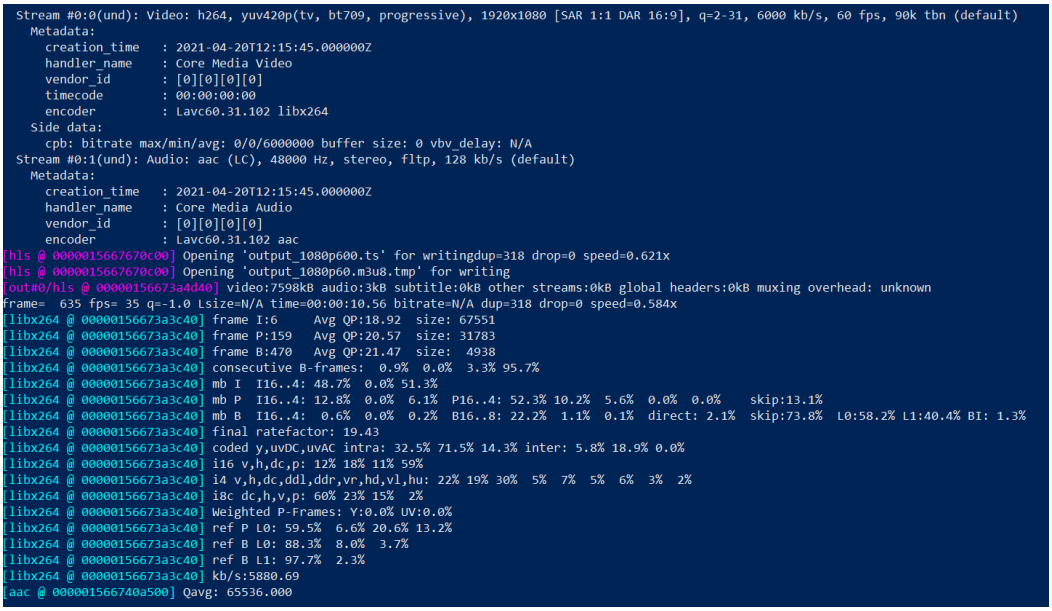
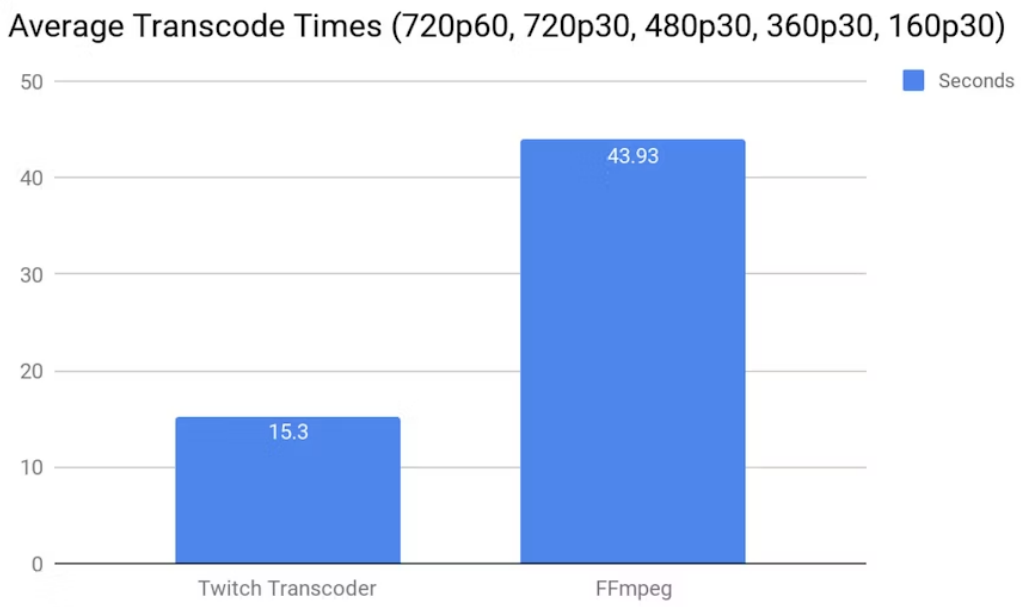


Figure 2 FFmpeg experiment on 1060p60fps

In my experiments, I utilized a 14-second video with a 4k resolution of 3840x2160p. Various parameters were specified using FFmpeg commands, such as the video codec (libx264), IDR interval, video size, FPS, target video bitrate, and H.264 profile. It's noteworthy that the processing time for these commands was approximately 10 seconds, indicating the efficiency of FFmpeg in handling such tasks.

Twitch Benchmarks vs FFmpeg



Average Transcode time twitch in house transcoder Figure 3 [2]

Twitch's proprietary in-house software, and FFmpeg, a widely-used open-source tool for video processing. TwitchTranscoder was developed to address technical challenges in transcoding and has been extensively used in Twitch's production for processing numerous concurrent live streams continuously. The benchmark tests aimed to compare the performance of TwitchTranscoder and FFmpeg in daily transcoding tasks. Both tools were fed with a Twitch live stream and a 1080p60 video file using identical presets, profiles, bitrates, and other flags. The tests involved transcoding each source into standard resolutions (720p60, 720p30, 480p30, 360p30, and 160p30). The hypothesis was that FFmpeg might transcode slower for file input and struggle with live streaming [2]. In the case of Twitch's full Adaptive Bitrate (ABR) ladder, TwitchTranscoder outperforms FFmpeg, saving 65% in execution time. This indicates the efficiency and superiority of TwitchTranscoder, especially in managing complex transcoding tasks and multiple variants.

Source IDR Problem

However, H.264's lossy compression standard results in inevitable video quality degradation during transcoding. FFmpeg is employed to generate HTTP Live Streaming (HLS) for live streams, but it poses technical challenges. The IDR frame alignment in HLS variants, particularly with different IDR intervals in the source RTMP bitstream, can lead to issues. Additionally, FFmpeg's limitation of creating N streams for N qualities is addressed by Twitch's custom transcoding software. Twitch's solution involves a 1-in-1-out transcoder, comprising a decoder, scaler, and encoder. This configuration aims to address problems related to IDR frames when a user's network changes. By using a single decoder for identical streams, Twitch eliminates redundant decoders and feeds decoded images to downstream scalers and encoders. This optimization reduces the computational expense of scaling, enhancing the overall performance of the transcoder [1].

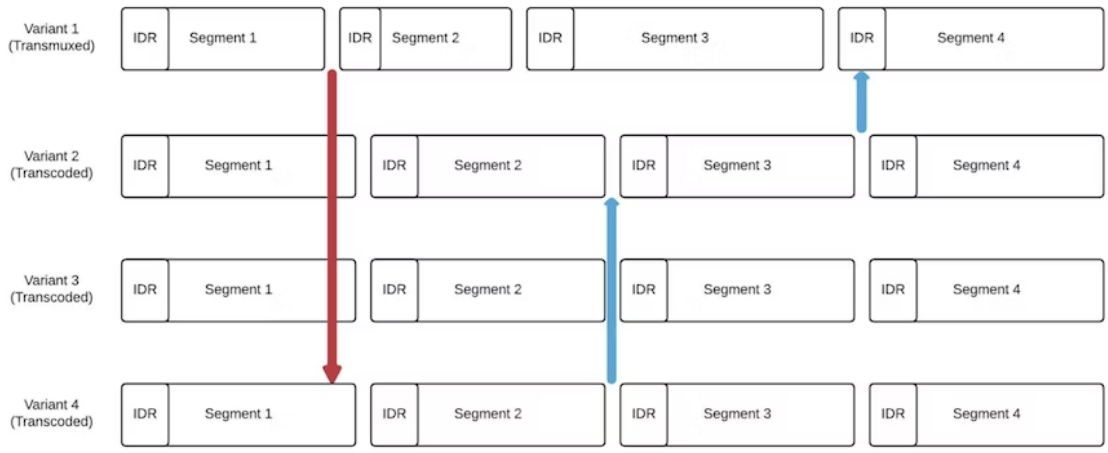


Figure 3 IDR frames

In HTTP Live Streaming (HLS), a variant is composed of segments, each commencing with an IDR frame. Alignment of IDR frames is crucial as per the HLS specification, ensuring that corresponding segments have the same Presentation Timestamp (PTS). This synchronization allows for seamless switching among variants by an HLS Adaptive Bitrate (ABR) player, facilitating adjustments based on the viewer's network conditions. While transcoding both the source and other variants results in perfectly time-aligned HLS segments by encoding IDR frames precisely every 2 seconds, the challenge arises when dealing with the source RTMP

bitstream. The IDR intervals in the source are determined solely by the broadcast software's configuration, and when transmuxing the source, there is no guarantee of alignment between the segments of the transmuxed and transcoded variants. This lack of alignment can lead to playback issues, as observed with Chromecast experiencing playback pauses when receiving HLS streams with misaligned segments. [1]

Twitch VP9 Implementation

In an article in 2018, twitch stated the plan to adopt the VP9 video compression standard, a move spurred by the apparent limitations of the long-standing H.264 format. The latter, in use since 2003, has been instrumental in supporting various applications like HDTV, Blu-ray Disc, and online streaming services such as YouTube and Twitch [3].

However, Twitch's analysis indicates that H.264 has reached its compression performance ceiling, especially for real-time encoding of gaming content at 1080p60 resolution.
In response to this challenge, Twitch plans to leverage newer-generation video standards, including VP9, known for its significant compression gains. While VP9 decoding and playback are already well-supported, this reveals that real-time encoding of gaming video content with high efficiency poses a substantial challenge due to the complexity of VP9. While VP9 decoding and playback are already well-supported, this reveals that real-time encoding of gaming video content with high efficiency poses a substantial challenge due to the complexity of VP9 [4].
Twitch has undertaken a rigorous feasibility study and has chosen FPGA (Field-Programmable Gate Array) as the hardware platform for real-time VP9 encoding. This strategic decision is expected to enable the platform to broadcast premium esports and partner channels with improved compression efficiency in the near future.

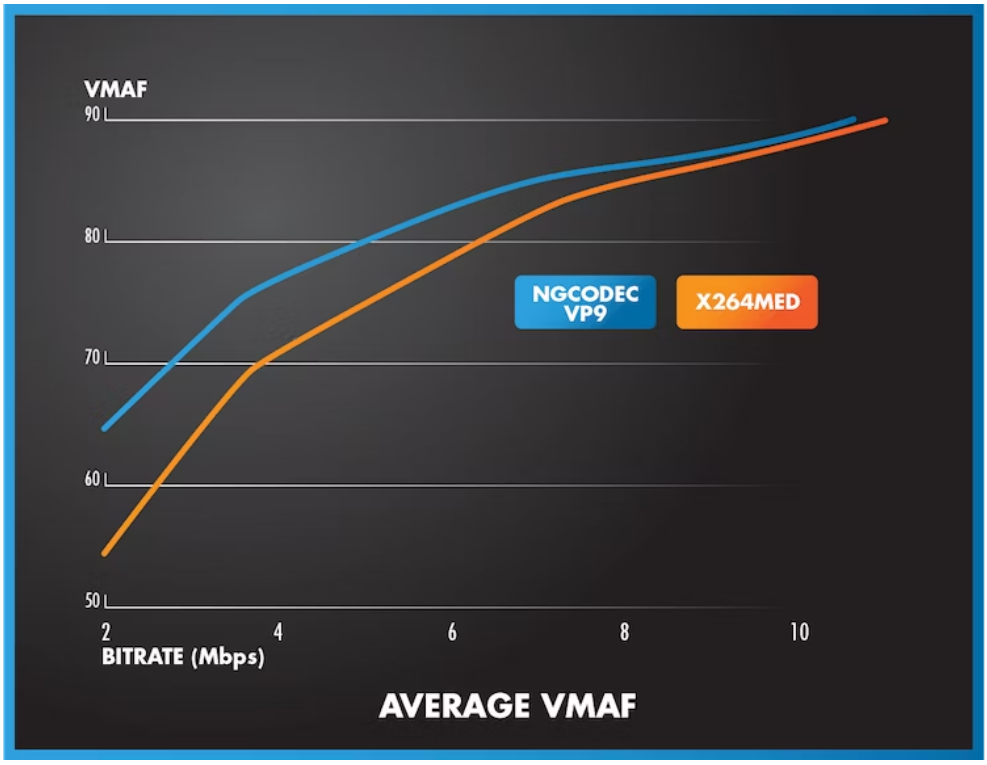


Figure 2 [3]

Key features explored include larger prediction block sizes, wider transform choices, and a full mode search for rate-distortion optimization. Additionally, Twitch discusses the significance of rate control optimizations, specifically the implementation of advanced adaptive quantization (AQ) and rate control algorithms in the NGCodec VP9 encoder. These technologies aim to smartly allocate bits based on scene complexity and achieve consistent video quality, especially during challenging scenes or scene changes in gaming content. Overall, the transition to VP9, supported by FPGA technology, promises a more efficient and superior streaming experience for Twitch users [3].

Twitch Search and Storage

Video System:

- **Ingest and Transcode:**
    - Twitch's video system starts with the ingestion of content from broadcasters. This is typically in the form of RTMP (Real-Time Messaging Protocol) streams, which are commonly used for live streaming.
    - The transcode system plays a crucial role in video processing. It takes the incoming RTMP streams and transcodes them into multiple HLS (HTTP Live Streaming) streams. HLS is an adaptive streaming protocol that allows Twitch to deliver video content at varying qualities based on the viewer's network conditions.
- **VOD Archiving:**
    - The Video on Demand (VOD) aspect involves archiving the incoming video streams for later retrieval. This creates a repository of recorded content, enabling users to watch streams they missed during the live broadcast.
    - Archiving likely involves storing different versions of the content, including various resolutions and bitrates to support adaptive bitrate streaming for VOD playback.

Differentiated Storage:

- **Live Content Storage:**
    - Live content, being dynamic and requiring real-time accessibility, might be stored in storage systems optimized for low-latency access. This storage needs to support the ongoing ingestion and distribution of live streams without compromising performance.
    - Storage systems for live content are likely designed for high throughput and low latency, ensuring a smooth and uninterrupted streaming experience for viewers.
- **Archived VOD Storage:**
    - Archived VODs, on the other hand, may be transitioned to lower-cost, long-term storage solutions. This is a common practice to optimize costs, especially considering that archived content may not be accessed as frequently as live content..
    - Long-term storage solutions could include technologies like object storage or cold storage, which provide cost-effective options for retaining large volumes of data over an extended period.

Live Content Storage:

- **Dynamic and Real-Time Accessibility:**
    - Live content, such as ongoing streams, needs to be stored in a storage system optimized for dynamic and real-time accessibility. Viewers are actively engaging with the content as it is being generated, and any delay or latency in accessing the live stream can negatively impact the user experience.
- **Low-Latency Access:**
    - The storage system for live content must be capable of low-latency access to support the continuous streaming experience. Low-latency access ensures that viewers receive the most up-to-date content with minimal delay.
- **High Throughput:**
    - Since live content involves continuous ingestion and distribution of video streams, the storage system must handle high throughput. This means the system should efficiently manage the rapid input and output of video data without bottlenecks.
- **Temporary Storage:**
    - Live content storage is often temporary, serving as a buffer during the streaming session. Once the live stream concludes, the content is typically archived or transitioned to long-term storage for later access as a VOD.

Highlights and Extended VOD Preservation:

- **Indefinite Storage:**
    - Highlights are curated, shorter segments extracted from past broadcasts. Unlike the default storage duration for past broadcasts (7 days for regular broadcasters), highlights are saved indefinitely. This means that once a portion of a stream is highlighted, it remains accessible on the creator's channel without the risk of automatic deletion.
- **Preserving Key Moments:**
    - Creators can use the Highlighter tool to select specific segments from their past broadcasts, preserving key moments, impressive gameplay, or interesting interactions with the audience. By doing so, they ensure that these moments are not lost when the default VOD storage period expires.
- **Preventing Automatic Deletion:**
    - Since highlights are not subject to the same retention policies as regular past broadcasts, they provide a way to bypass the automatic deletion timeline. Most Twitch Partners, Prime, and Twitch Turbo users enjoy a 60-day storage period for past broadcasts, but highlights surpass this limitation by being saved indefinitely.

- **Customizable and Manageable:**
    - Creators have control over the title and categorization of each highlight. This allows for easy organization and navigation of the content library. The ability to manage and customize highlights makes it convenient for creators to present their best content to viewers without worrying about time constraints.
- **Accessible Archives:**
    - Highlights essentially serve as curated archives of a creator's best and most engaging content. Viewers can navigate to a creator's channel and explore these highlights at any time, providing a way to revisit memorable moments or catch up on content they might have missed during the live broadcast.
- **Viewer Engagement:**
    - As viewers discover interesting highlights, they may be more likely to engage with the channel and explore additional content. This ongoing engagement benefits both creators and viewers by fostering a sense of community and appreciation for the content

Understanding Twitch's Search Infrastructure:

In the vibrant realm of live streaming, Twitch has carved a niche as the ultimate platform for gamers, artists, and content creators.Twitch's search experience is powered by a sophisticated system comprising several key components: an API, a retrieval system, a ranking system, and a content ingestion pipeline.

a. API Orchestration:

The API acts as the gatekeeper, managing user requests and ensuring seamless communication between various systems. It oversees experimentation, event tracking, and interactions with retrieval and ranking modules.

Challenges:

Experimentation Handling: Twitch continuously experiments with features, requiring robust API management to maintain stability during changes.
Event Tracking Efficiency: Efficient event tracking is vital for understanding user behavior, demanding optimized mechanisms for accurate data collection.

b. Retrieval System:

Powered by Amazon OpenSearch service, this system identifies matches across live streams, channels, categories, and videos. It forms the foundation of Twitch's content discovery, providing the initial set of results for user queries.

Challenges:

Lexical Match Precision: Fine-tuning algorithms is essential for precise matches, ensuring users get results closely aligned with their search intent.
Dynamic Content Indexing: Real-time indexing of new content is crucial. Implementing efficient mechanisms for content ingestion and indexing reduces latency for newly created streams.

c. Ranking System:

After retrieving potential matches, machine learning models within the ranking system step in. These models re-rank results based on user interactions, such as click-through rates and viewing duration, ensuring the most relevant and engaging content rises to the top.

Challenges:

User Interaction Data Quality: Accurate user interaction data is paramount. Addressing issues related to bot interactions or fake engagement metrics ensures genuine user feedback informs the ranking system. *Model Adaptability:* Machine learning models must adapt to changing user behaviors. Continuous model training using up-to-date data and evolving algorithms is essential for maintaining relevance.

Machine Learning-Powered Recommendations:

Twitch's recommendation system personalizes user experiences, generating high-quality recommendations based on viewers' preferences. To achieve this, Twitch utilizes machine learning algorithms that adapt to users' evolving habits.
These algorithms are supported by a robust infrastructure, including the Machine Learning Feature Store (MLFS), Real-Time Feature tools, Model Registry, and Model Deployment System.

Machine Learning Feature Store (MLFS): MLFS is a system that manages and shares features across teams. Features are aspects or characteristics of data that machine learning algorithms use to

make predictions or recommendations. MLFS ensures these features are stored centrally and can be accessed by different teams. Challenges related to MLFS include ensuring that the stored features are relevant and updated in real-time to improve the accuracy of recommendations.

Real-Time Feature: Real-Time Feature is a tool built on AWS Kinesis data streams. It enables the processing of streaming features with minimal delay or latency. The challenge here is to ensure seamless integration and synchronization between Real-Time Feature tools and MLFS. This integration is essential for providing up-to-the moment insights, enhancing the accuracy of recommendations.

Model Registry: The Model Registry serves as the source of truth for model metadata. It is crucial for maintaining consistency between the models used for training and the ones deployed in production. Any discrepancies in the integration between training pipelines (where models are developed) and production services (where models are used) can lead to inconsistencies in the quality of recommendations.

Model Deployment System: The Model Deployment System refers to the continuous integration/continuous deployment (CI/CD) pipeline used for deploying machine learning models. CI/CD pipelines streamline the process of deploying models into production. However, there is a need for further enhancements in integration testing and rollback mechanisms. Integration testing ensures that the deployed models work seamlessly with the existing systems, and rollback mechanisms allow reverting to previous versions of models if issues arise, ensuring optimal safety and efficiency.

Improving Twitch Search Functionality

Twitch, as a leading live streaming platform, is constantly seeking ways to enhance user experience, particularly in the realm of search functionality. As the platform caters to a diverse global audience, ensuring that users can easily discover relevant streams is of paramount importance.

1. Real-Time Language Detection:

a. Concept:

Twitch faces the challenge of catering to a diverse global audience, leading to streams in multiple languages. Real time language detection is essential for ensuring that users can filter streams based on spoken languages, enhancing their viewing experience. Twitch can achieve this by implementing the Google Cloud Speech-to-Text API, a service that converts spoken language into written text. The API's ability to detect spoken languages in live streams provides Twitch with accurate language data, ensuring users can

search for streams in their preferred languages. Twitch does indeed have language tags that allow streamers to specify the language they are speaking during their streams. These language tags help viewers find content in their preferred language. Not all streamers manually tag their streams with language information. While some conscientious streamers may diligently tag their content with the appropriate language, others might overlook this step or forget to update it when they switch languages during their streams. This variation in tagging behaviour can result in an inconsistent user experience when viewers are searching for streams in specific languages.

Real-time language detection provides several benefits over manual tagging:

1. **Accuracy:** Real-time language detection algorithms can accurately identify the language being spoken in a stream, ensuring that viewers are presented with streams in their desired language without relying on potentially inaccurate manual tags.
2. **Consistency:** Real-time detection ensures consistent and accurate language information across all streams, regardless of whether the streamer remembered to tag their content correctly. This consistency enhances the overall user experience and search accuracy.
3. **Convenience:** Viewers don't need to rely on streamers to tag their streams with the correct language manually. Real-time language detection automates the process, making it more convenient for both streamers and viewers.
4. **Dynamic Language Switching:** Some streamers may switch between languages during their streams based on viewer requests or other factors. Real-time language detection can adapt to these language switches in real-time, providing viewers with up-to-date language information.
5. **Multilingual Streams:** In cases where streamers speak multiple languages during their streams, real-time language detection can accurately identify and label each language segment, allowing viewers to filter streams based on their specific language preferences.

b. Implementation Steps:

Integration of Google Cloud Speech-to-Text API:

Twitch integrates the Google Cloud Speech-to-Text API into its streaming platform, enabling real-time transcription of live stream audio.
Data Transmission: Live audio from streams is sent to the API, where it undergoes transcription, effectively converting spoken content into text.

Language Identification:

The API not only transcribes spoken content but also identifies the language being spoken, leveraging its language detection capabilities.

API Response Processing: Twitch receives the detected language information from the API response, associating it with the corresponding stream.

Benefits:

Accurate Language Tags: Streams are accurately labeled with the detected language, improving search accuracy for multilingual users.

Real-Time Identification: Users can filter streams based on languages spoken in real-time, ensuring they find content in their preferred languages.

2. Semantic Search and Contextual Analysis:

a. Concept:

Semantic search techniques, including Word Embeddings (Word2Vec) and BERT, enhance search accuracy by understanding the context and nuances of user queries and stream titles. Word Embeddings map gaming terminology, capturing semantic relationships, while BERT comprehends the contextual nuances of gaming language, aligning search results closely with user intent.

b. Implementation Steps:

Utilizing Word Embeddings (Word2Vec):

Twitch builds a custom Word2Vec model using gaming-related text corpora, which could include game guides, forums, or gaming websites.

Training the Model: The model learns to represent gaming terminology in a vector space, where semantically related words are positioned nearby.

Mapping User Queries and Stream Titles: When a user enters a search query, the query terms are mapped to vectors using the trained Word2Vec model. Stream titles are preprocessed, and their constituent words are similarly mapped to vectors.

Matching Algorithm: Twitch uses cosine similarity or Euclidean distance to measure the similarity between the user query vector and the vectors representing stream titles. Titles with vectors closest to the user query vector are considered the most relevant and are displayed in the search results.

Implementing BERT (Bidirectional Encoder Representations from Transformers):

Twitch uses a pretrained BERT model on gaming-related text data to understand the contextual nuances of gaming language.

Contextual Encoding: BERT encodes both user queries and stream titles contextually, understanding the relationships between words in the query and title.

Semantic Matching: BERT-encoded user queries and stream titles are compared using similarity metrics (e.g., cosine similarity) to identify the most contextually relevant titles.

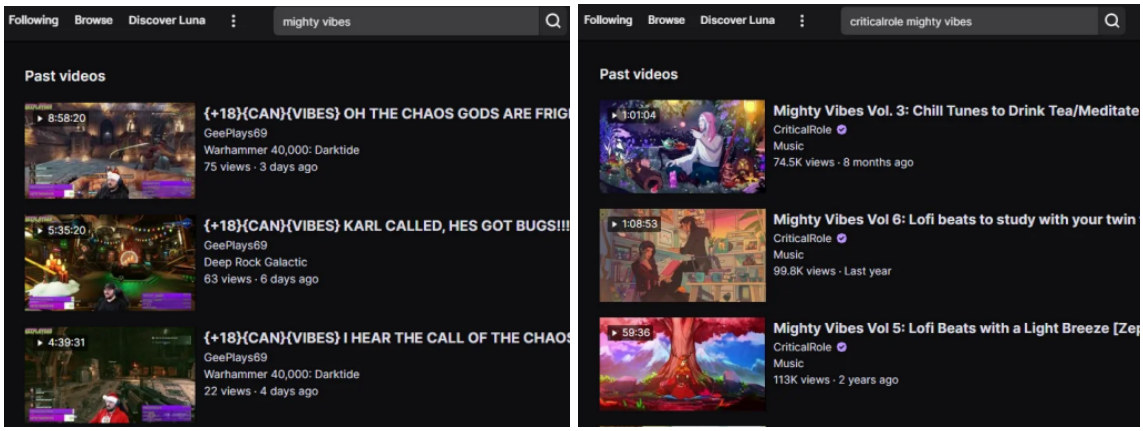
Fine-Tuning (Optional): Twitch can fine-tune the BERT model on specific gaming-related queries and titles to adapt the model to the unique nuances of gaming language used on the platform.

Benefits:

Fine-Tuning and Contextual Relevance: Regular updates and fine-tuning of Word Embeddings and BERT models ensure that gaming-specific language nuances are captured effectively, delivering highly relevant results even for niche or specialized queries.

Adaptable Algorithms: Implementing natural language processing enhancements can handle diverse query structures, addressing complex user queries, which may include slang or gaming jargon.

3. Learning to Rank (LTR) Algorithms:



a. Concept:

LTR algorithms optimize search result ranking based on user interactions. These algorithms use machine learning techniques to predict the likelihood of a user engaging with a particular stream. By training models using historical user data and various features, including click-through rates, viewing duration, user engagement history, and streamer popularity, Twitch dynamically adjust search rankings, promoting the most relevant and engaging streams.

b. Implementation Steps:

Feature Extraction:

Features such as click-through rates, viewing duration, user engagement history, streamer popularity, and content category are extracted from user interactions.

Data Collection: Twitch collects historical user data and relevant features to train the machine learning models. **Training the Model:**

Machine learning models, such as gradient boosting machines or neural networks, are trained using the historical user data and extracted features.

Prediction: These models predict the likelihood of a user engaging with specific streams based on the features provided.

Real-Time Ranking:

When a user performs a search, the LTR model evaluates the candidate streams and ranks them based on the predicted user engagement probability.

Dynamic Adjustment: The rankings are dynamically adjusted in real-time, ensuring the most relevant streams are prominently displayed.

Continuous Feedback Loop:

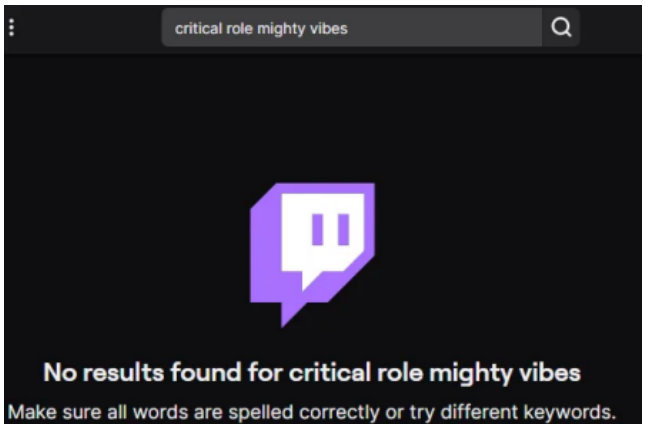
User feedback, such as likes, dislikes, and time spent on a stream, is continuously collected. **Retraining the Model:** This feedback is used to retrain the LTR model, ensuring it adapts to changing user preferences over time.

Benefits:

Dynamic Ranking and User Engagement: Real-time adjustments ensure the most relevant streams are prominently displayed, improving user experience. Continuous model training using up-to-date data and evolving algorithms is essential for maintaining relevance.

Optimized User Experience: Ranking adjustments based on real-time user interactions ensure users discover content aligned with their interests, increasing overall platform usability.

4. Improved Keyword Matching:



a. Issue:

Users face challenges when there are spaces between words in search queries.

b. Solution:

Enhance tokenization and implement phrase matching algorithms for accurate search results.

c. Benefits:

Users can search for specific stream titles or channel names without worrying about exact spacing between words.

Twitch Live-Streaming Using Starlink

For streamers and viewers living in rural communities, they have limited options when it comes to connecting to the internet if the infrastructure is not set up. But recently there's a new option which is to connect to Starlink which provides an internet service by forming a constellation of small satellites in low Earth orbit, creating a web of interconnected nodes that beam internet signals to ground-based user terminals, commonly referred to as satellite dishes.

The advantage is it allows users to connect to the internet in remote or rural areas where traditional infrastructure might be scarce or unavailable. This opens up opportunities for streamers to broadcast from diverse and unconventional locations, tapping into new and unique content possibilities.

1. Current Twitch Network Protocol:

Twitch currently uses HTTP Live Streaming (HLS) as its default communication protocol which is the industry standard and has widespread support and implementation. Developed by Apple, HLS enables adaptive streaming, allowing seamless playback on a variety of devices with different network conditions. The key principle behind HLS is its ability to dynamically adjust video quality based on the viewer's internet speed and device capabilities. The following features are the main reasons for why HLS is good for streaming:

1. Adaptive Bitrate (ABR) Streaming:

   Adaptive Bitrate (ABR) dynamically adjusts the quality of the video stream in real-time, depending on the viewer's internet speed and device capabilities. This ensures a seamless viewing experience by automatically switching between different quality levels without interruptions.

2. Segmentation:

   Twitch utilizes HTTP Live Streaming (HLS) to deliver live video content to viewers. The streaming process involves breaking down the video stream into smaller segments, typically lasting a few seconds each. These segments are encoded at various quality levels, allowing for adaptive streaming based on the viewer's network conditions.

3. Buffering:

Buffering is an integral part of the HLS streaming experience on Twitch. It allows the client-side player to store a small portion of the video in advance, helping to mitigate interruptions caused by fluctuations in network conditions. This buffering mechanism smoothens playback and minimizes the impact of momentary drops in internet speed.

2. Challenges with Starlink:

While HLS generally performs well, certain challenges arise when using it with Starlink, SpaceX's satellite-based internet service.

1. Latency:

   Starlink introduces higher latency compared to traditional terrestrial internet. The round-trip time for data to travel to and from low Earth orbit satellites can result in delays, impacting the real-time nature of live streaming on platforms like Twitch.
2. Packet Loss and Jitter:

   Satellite-based internet, including Starlink, is susceptible to packet loss and increased jitter due to the nature of communication between Earth and satellites. This can lead to a less stable connection, affecting the reliability of the streaming experience.
3. Intermittent Connectivity:

   Starlink's connectivity can be intermittent, causing brief periods of signal loss. HLS relies on consistent and reliable internet connections, and interruptions in connectivity can result in buffering and disruptions in the streaming experience.
4. Satellite Constellation Dynamics:

   The dynamic nature of the Starlink satellite constellation, with satellites orbiting the Earth, introduces variability in signal strength and quality. This variability can impact the continuous and smooth delivery of HLS segments. For instance, Satellite Handovers lead to consistent ping spikes, packet loss, and packet delays.

3. Solution and Limitations:

WebRTC (Web Real-Time Communication) is a protocol designed for real-time data exchange, including audio and video streaming. It introduces some of the issues lacking in HLS but introduces its own new features for overcoming issues associated with satellite-based internet services like Starlink.

1. Low Latency:

   WebRTC is designed for low-latency communication, making it well-suited for real-time applications like video streaming. The direct peer-to-peer communication eliminates the need for intermediaries, reducing latency compared to traditional protocols. This is particularly beneficial for interactive applications, such as live streaming on platforms like Twitch.
2. Improved Packet Loss Handling:

   WebRTC is optimized for real-time communication and includes mechanisms to handle packet loss more gracefully. Techniques like Forward Error Correction (FEC) and retransmission help mitigate the impact of packet loss, providing a more robust streaming experience.
3. Dynamic Adaptation:

   WebRTC adapts to changing network conditions dynamically, adjusting parameters such as bitrate and codec to optimize streaming quality. This adaptability is crucial for maintaining a consistent streaming experience, especially in situations where the available bandwidth may vary.
4. Consistent Connectivity:

   WebRTC is designed to handle intermittent connectivity more effectively. It includes mechanisms for reconnecting and re-establishing connections, reducing the impact of brief interruptions in the network.

While WebRTC helps improve network connectivity for live-streaming, it is still limited due to the physical reality of using satellites such as extreme weather, signal reflection, network congestion, and so on that cannot be fixed with pure software. However as technology progresses and Starlink expands it's network, it will take off the burden on software to improve the network connectivity and make it smooth and error proof.

Conclusion:

By implementing these innovative solutions, Twitch could significantly enhance its search functionality. Real-time language detection ensures accurate language tags, empowering users to find content in their preferred languages effortlessly. Semantic search techniques, powered by Word Embeddings and BERT, enable precise and contextually relevant matches, aligning

closely with user intent. Learning to Rank (LTR) algorithms and improved keyword matching further enhance user flexibility and engagement. These implementations are pivotal in creating a seamless and satisfying Twitch experience, tailored to each user's preferences and requirements.

Contribution

Name	**Contribution**
Sam	Twitch Search and Storage
Josh	Twitch Compression and Transcoding
Baraa	Twitch Live-Streaming Using Starlink and formating

References

[1] “Live Video Transmuxing/Transcoding: Ffmpeg vs Twitchtranscoder, Part I.” *Twitch Blog*, 10 Oct. 2017, blog.twitch.tv/en/2017/10/10/live-video-transmuxing-transcoding-f-fmpeg-vs-twitch-transcoder-part-i-489c1c125f28/.

[2] “Live Video Transmuxing/Transcoding: Ffmpeg vs Twitchtranscoder, Part II.” *Twitch Blog*, 23 Oct. 2017, blog.twitch.tv/en/2017/10/23/live-video-transmuxing-transcoding-f-fmpeg-vs-twitch-transcoder-part-ii-4973f475f8a3/.

[3] “How VP9 Delivers Value for Twitch’s Esports Live Streaming.” *Twitch Blog*, 19 Dec. 2018, blog.twitch.tv/en/2018/12/19/how-v-p9-delivers-value-for-twitch-s-esports-live-streaming-35db26f6322f/.

[4] “VP9.” *Wikipedia*, Wikimedia Foundation, 30 Oct. 2023, en.wikipedia.org/wiki/VP9.

https://help.twitch.tv/s/article/video-on-demand?language=en_US

https://blog.twitch.tv/en/2015/12/18/twitch-engineering-an-introduction-and-overview-a23917b71a25/

https://help.twitch.tv/s/article/how-does-search-work?language=en_US