Dataset

**3.1 Representative Song Selection Algorithm**

3.1.1 Dataset Sources

The study utilized two primary datasets:

* JKU Dataset: [Description of JKU dataset and its characteristics]
* Audio Feature-Genre Linking Dataset: [Description of the second dataset that links songs to genres based on audio features]

3.1.2 Algorithm Objective

The algorithm was designed to select 50 representative unique videos for all genres while addressing the challenge that larger genres tend to have more overlaps with different genres and often achieve higher scores. The primary goal was to maximize representation of underrepresented genres by ensuring they also receive representative songs.

3.1.3 Algorithm Design Principles

Input Requirements:

* For each genre: a ranked list of candidate videos with scores (higher = better)

Output Goals:

* For each genre: exactly k unique videos (no video assigned to multiple genres)
* Fair conflict resolution when videos are contested by multiple genres

Key Concepts:

* Assigned Set: All videos already allocated to a genre
* Fallback Score: The score of a genre's next best unused video (used for conflict resolution)

3.1.4 Algorithm Implementation Steps

Step 1: Initial Video Assignment

* For every genre with ≥ k candidates:
  + Assign top k videos (highest scores)
  + Track position in ranked list for fallback calculations
* Exclude genres with < k candidates

Step 2: Conflict Detection

* Identify videos assigned to multiple genres
* If no conflicts exist, proceed to finalization

Step 3: Conflict Resolution For each contested video:

* Calculate fallback scores for all competing genres
* Winner Rule: Genre with lowest fallback score retains the video
  + *Rationale*: The genre with the worse alternative has greater need
* Tie-Breaking Hierarchy:
  + Higher score on contested video
  + Alphabetical order of genre name

Step 4: Loser Refill Process

* Remove contested video from losing genres
* Replace with next best unassigned video
* Continue until k videos achieved or candidates exhausted
* Update position pointers
* Return to Step 2 (check for new conflicts)

Step 5: Finalization

* Remove genres with < k videos
* Output final assignments with scores

**3.2 YouTube Data Acquisition**

3.2.1 API Utilization

* Platform: YouTube Data API v3 (Free tier)
* Extraction Target: Latest 100 comments per representative video
* Data Collection Period: [Specify timeframe]

3.2.2 Comment-Level Data Fields

The following comment metadata was extracted:

* commentId: Unique comment identifier
* authorChannelId: Comment author's channel ID
* authorDisplayName: Public display name
* authorChannelUrl: Author's channel URL
* publishedAt\_x: Comment publication timestamp
* updatedAt: Last modification timestamp
* likeCount\_x: Comment like count
* parentId: Parent comment ID (for replies)
* textDisplay: Rendered comment text
* textOriginal: Raw comment text
* videoId: Associated video identifier
* genre: Assigned music genre
* lang: Detected language

3.2.3 Video-Level Metadata

Comprehensive video information was collected:

Basic Information:

* title, description, publishedAt\_y
* channelId, channelTitle, categoryId

Content Descriptors:

* tags, liveBroadcastContent, duration
* dimension, definition, caption

Engagement Metrics:

* viewCount, likeCount\_y, commentCount, favoriteCount

Technical Attributes:

* licensedContent, privacyStatus, madeForKids
* selfDeclaredMadeForKids, uploadStatus, embeddable, license

**3.3 Data Preprocessing**

3.3.1 Language Filtering

Given that both VADER sentiment analysis and LIWC require English text for accurate processing, only English-language comments were retained for analysis. This language filtering was applied to the initial dataset of 356,973 comments collected across all genres.

3.3.2 Text Cleaning Pipeline

A systematic text preprocessing pipeline was implemented to standardize comment text:

Step 1: Whitespace Normalization

def preprocessing\_remove\_newlines\_tabs\_and\_spaces(text: str) -> str:

text\_cleaned = re.sub(r'[\n\t]+', ' ', text)

text\_final = re.sub(r'\s+', ' ', text\_cleaned).strip()

return text\_final

* Removal of newline characters (\n) and tab characters (\t)
* Consolidation of multiple consecutive spaces into single spaces
* Trimming of leading and trailing whitespace

Step 2: Case Normalization

def preprocessing\_lowercase(text: str) -> str:

return text.lower()

* Conversion of all text to lowercase for consistent processing

Step 3: Punctuation Removal

def preprocessing\_remove\_punctuation(text: str) -> str:

translator = str.maketrans('', '', string.punctuation)

return text.translate(translator)

* Complete removal of punctuation marks using Python's string.punctuation set

Combined Preprocessing Function

def preprocess\_combined(text: str) -> str:

text\_no\_punctuation = preprocessing\_remove\_punctuation(text)

text\_cleaned = preprocessing\_remove\_newlines\_tabs\_and\_spaces(text\_no\_punctuation)

preprocessed = preprocessing\_lowercase(text\_cleaned)

return preprocessed

3.3.3 Genre Filtering Based on Comment Volume

To ensure statistical reliability and meaningful analysis, genres with insufficient English comment representation were excluded from the study:

* Minimum Comment Threshold: 500 English comments per genre
* Initial Genre Count: 260 genres (from algorithm selection)
* Final Genre Count: 233 genres (retained after filtering)
* Excluded Genres: 27 genres with < 500 English comments

This filtering step was essential to maintain analytical validity, as genres with very few comments would not provide sufficient data for robust sentiment analysis and linguistic feature extraction.

**3.4 Sentiment Analysis and Linguistic Feature Extraction**

3.4.1 VADER Sentiment Analysis

Tool Selection Rationale: VADER (Valence Aware Dictionary and sEntiment Reasoner) was selected for sentiment analysis due to its optimization for social media text analysis and ability to handle informal language, slang, and emoticons commonly found in YouTube comments.

Implementation: VADER was applied to the preprocessed English comment text to generate sentiment scores across four dimensions:

* Positive Score: Proportion of positive sentiment
* Negative Score: Proportion of negative sentiment
* Neutral Score: Proportion of neutral sentiment
* Compound Score: Normalized composite score ranging from -1 (most negative) to +1 (most positive)

Language Constraint: VADER analysis was restricted to English comments only, as the tool's sentiment lexicon and rules are specifically designed for English language processing.

3.4.2 LIWC Analysis Integration

Tool Application: Linguistic Inquiry and Word Count (LIWC) was employed to extract psycholinguistic features from the comment text, providing insights into psychological processes, social relationships, and cognitive patterns expressed in user comments.

Feature Categories: LIWC analysis captured multiple linguistic dimensions including:

* Psychological processes (cognitive, emotional, social)
* Personal concerns (work, leisure, money, religion)
* Linguistic dimensions (word count, pronouns, articles)
* Grammatical categories (verbs, adjectives, prepositions)

Language Limitation: Like VADER, LIWC analysis was constrained to English comments due to the tool's English-language dictionary and processing algorithms.

3.4.3 Final Dataset Characteristics

* Total Initial Comments: 356,973 comments
* Language Filter: English comments only
* Final Genre Count: 233 genres (≥ 500 English comments each)
* Excluded Genres: 27 genres (< 500 English comments)

3.5 Additional Data Processing

[Include any other processing steps you performed, such as:]

* Statistical feature engineering
* Data aggregation methods
* Quality control measures
* Final dataset preparation