EXPLORING PLAYER SENTIMENT AND JARGON USE IN WORLD OF WARCRAFT

A Data-Driven Approach to Understanding Player Sentiment

Abstract

The World of Warcraft (WoW) community is a passionate and involved ecosystem where players engage in lively discussions about game updates, mechanics, and experiences. This project explores sentiment trends and jargon usage in over 1 million forum posts spanning four major expansions: Battle for Azeroth, Shadowlands, Dragonflight, and The War Within. Using fine-tuned natural language processing models, the analysis distinguishes between jargon-heavy posts—dominated by language used by experienced players—and general posts that reflect broader community sentiment.

Sarah Berg

1 Contents

2 Introduction

Since its inception in 2004, World of Warcraft (WoW) has attracted millions of players worldwide, creating a lively and devoted community. This participation is apparent in the official WoW forums, where players exchange many postings about game updates, strategies, and shared experiences. Understanding the sentiments driving these discussions is essential for developers looking to improve player experiences.

Traditional sentiment analysis sometimes overlooks the intricacies of jargon-heavy communications—posts filled with language only used in gaming. These posts, generally written by seasoned players fully invested in the game, provide the necessary perspectives to help with game development. However, the language used might present difficulties for common analytical methods, potentially leading to an inadequate assessment of player sentiments.

This project seeks to bridge that gap by examining how sentiment trends in jargon-heavy posts compare to those in general posts across major WoW patches. By utilizing natural language processing techniques, the goal is to discover patterns that will provide developers and community managers with practical knowledge to help improve the game.

3 Research Question

Research Question: How do the sentiment trends (negative, neutral, positive) in jargon-heavy posts compare to those in general posts across major World of Warcraft patches?

Context: Previous research in gaming communities has primarily focused on general sentiment trends, often neglecting the role of technical jargon in shaping these sentiments. World of Warcraft forums offer a unique opportunity to explore this area due to the rich mix of casual commentary and highly technical critiques from a diverse player base.

Jargon-heavy posts often reflect the perspectives of experienced players who engage deeply with game mechanics, providing detailed feedback that is critical for developers. In contrast, general posts tend to capture the broader, casual audience's sentiments. By comparing sentiment trends across these two types of posts, we aim to uncover insights into how different subsets of the player base respond to game updates, helping developers balance content effectively.

The WoW forums reflect the game's community, with discussions ranging from casual observations to in-depth theory crafting. Major patch releases frequently provoke conflicting opinions, making them an ideal setting for sentiment analysis of both jargon-heavy and general conversation.

Hypothesis:

Null Hypothesis (H_0): There is no significant difference in the sentiment trends between jargon-heavy and general posts across major patches.

Alternative Hypothesis (H_1): Jargon-heavy posts exhibit distinct sentiment trends compared to general posts, potentially skewing more critical due to their technical focus.

This hypothesis is plausible given that jargon-heavy discussions often contain detailed critiques or technical observations, while general posts may exhibit a more balanced or casual tone. By analyzing these differences, we aim to provide actionable insights that can help developers address technical feedback from experienced players while also considering the broader sentiments of the casual audience.

4 Data Collection

Overview

The dataset for this project was sourced from the official World of Warcraft forums, focusing on discussions tied to major updates across four expansions: *Battle for Azeroth*, *Shadowlands*, *Dragonflight*, and *The War Within*. Over 1,000,000 posts were collected, spanning from August 17, 2018, to November 13, 2024. The dataset included metadata such as timestamps, forum categories, user roles, and patch details, enriching the scope for analysis.

Methodology

- **Scrapy Framework:** Automated the data collection process, efficiently extracting static forum content and handling large volumes of posts across multiple threads.
- **MongoDB:** Stored data in a semi-structured format, preserving attributes like thread titles, user roles, and timestamps for easy access during preprocessing.

Scrapy was chosen for its exceptional ability in handling large-scale web scraping tasks, and MongoDB was selected for its flexibility in storing semi-structured data, accommodating the varying formats of forum posts.

Challenges and Solutions

One challenge was that some forum pages relied heavily on JavaScript rendering, which Scrapy couldn't process. To overcome this, the scraper was adjusted to target static elements and exclude dynamic content, such as 'Blizz Tracker' threads—official announcements rather than player discussions—ensuring the dataset focused on relevant player-generated content.

Advantages

- **Scalability:** Scrapy efficiently handled large-scale data extraction, enabling the collection of thousands of posts weekly without significant manual intervention.

Disadvantages

- **Dynamic Content Exclusion**: The inability to process JavaScript-rendered pages meant some relevant posts were omitted, potentially limiting the completeness of the dataset.

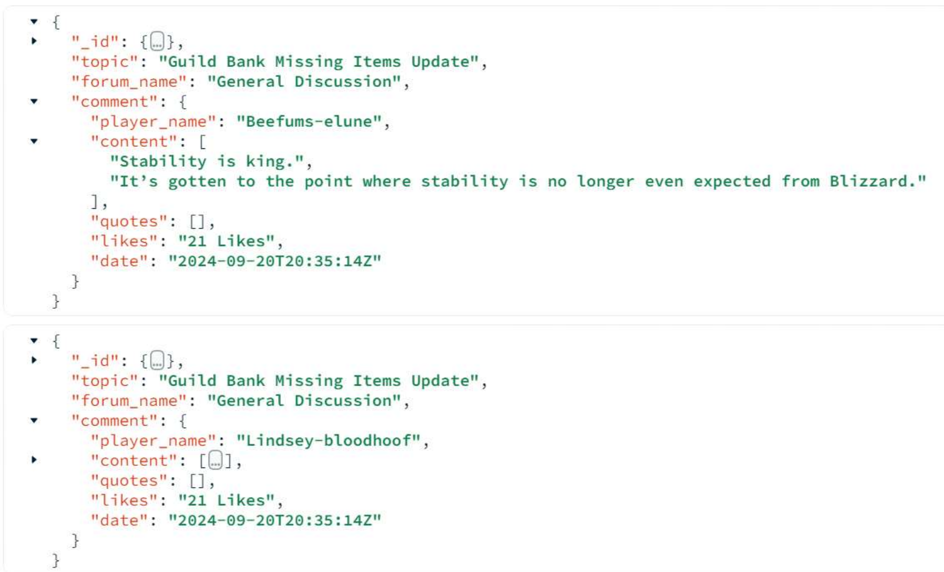

Figure 1 illustrates the MongoDB interface displaying raw entries from forum posts, highlighting how data is stored

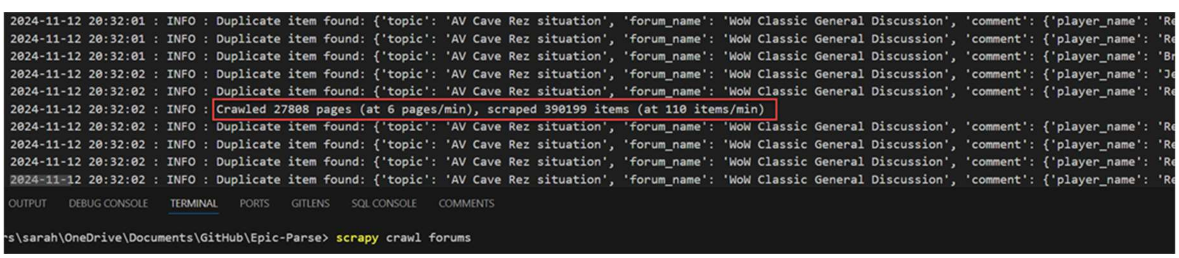

Figure 2 shows the Scrapy crawler in action, extracting data from the forums.

5 Data Extraction and Preparation

The data extraction and preparation process was designed to ensure the dataset was clean, structured, and ready for meaningful analysis. By leveraging a combination of

automated tools and systematic cleaning techniques, this phase transformed raw forum data into a refined resource for sentiment and jargon analysis.

Screenshots and code snippets for these steps are provided in the **Appendix.**

5.1 Data Extraction

Building upon the data collection process, the Scrapy crawler was programmed to focus on extracting metadata such as timestamps, forum categories, and post details, along with capturing raw player comments and quoted text. The crawler efficiently handled thousands of posts weekly, focusing on:

- **Metadata**: Extracting timestamps, forum categories, and post details.
- **Content**: Capturing raw player comments and identifying quoted text within threads.

The scraped data was stored in **MongoDB**, a NoSQL database ideal for handling semi-structured data. This allowed seamless integration of attributes like comments, user details, and patch metadata.

5.2 Data Preparation

Once extracted, the raw data underwent a thorough cleaning and transformation process to enhance its usability for analysis:

Patch Metadata Assignment

To contextualize posts, a patch dictionary mapped patch names to release dates across expansions (e.g., Battle for Azeroth, Shadowlands). By aligning timestamps with patch periods, each post was tagged with its corresponding patch and expansion, allowing time-series analysis of sentiment and jargon trends.

Player Categorization

Posts authored by Blizzard employees were identified using a curated list of official names. These posts were labeled as “Employee,” while all others were categorized as “Player,” enabling comparisons between developer communications and player feedback.

Text Cleaning

The cleaning process standardized text while preserving critical jargon. Key steps included:

- Resolving encoding issues and removing irrelevant content such as URLs and non-English characters.
- Expanding contractions (e.g., "don’t" → "do not") and retaining domain-specific symbols like "+," "-," and "/" to support jargon recognition.
- Creating a **cleaned_text** column that housed processed, normalized content ready for downstream analysis.

5.3 Feature Engineering

Several features were derived to support advanced analysis:

- **Word Count**: Calculated raw and cleaned word counts to categorize posts by length.
- **Jargon Ratio**: A custom dictionary of World of Warcraft-specific terms quantified the density of jargon in each post. Stopwords were excluded to focus on meaningful language. Thresholds were empirically determined based on exploratory analysis of post lengths and jargon usage patterns:
    - Short posts required higher jargon density to qualify.
    - Longer posts balanced jargon with explanatory content, necessitating a lower threshold.

Outcome

The preparation process resulted in a robust dataset enriched with metadata, sentiment classifications, and jargon analysis features. These steps laid the foundation for uncovering nuanced player discussions, sentiment dynamics, and trends across expansions.

Challenges and Solutions

A major challenge during data preparation was the time-intensive process of manually labeling a sufficient portion of the dataset for sentiment and jargon classification. With over 1,000,000 forum posts collected, manually annotating all entries was impractical. Instead, I focused on labeling a subset of approximately 25,000 posts to create a high-quality training dataset for fine-tuning BERT models.

To streamline the process, I used an automated assisted method. Initial labels were generated using GPT-4, providing a baseline for sentiment and jargon classification. This output was then manually reviewed and corrected to ensure accuracy. The refined subset enabled me to train BERT models, which were later applied to classify the full dataset efficiently.

Identifying jargon-heavy posts presented an additional challenge due to the specialized language of World of Warcraft forums. To address this, I developed a jargon ratio, a metric that quantifies the density of game-specific terms in a post using a custom jargon dictionary. This approach allowed scalable classification of posts as jargon-heavy or general without requiring exhaustive manual review.

By combining automated methods, manual validation, and the development of the jargon ratio, I overcame the challenges of labeling and ensured the dataset was both scalable and accurate for further analysis. This solution not only addressed the immediate difficulties but also laid the groundwork for a detailed methodology in identifying jargon-heavy content.

5.4 Determining Jargon-Heavy Content

Building upon the solutions devised during data preparation, classifying posts as jargon-heavy involved a multi-step process that combined text analysis, a custom jargon dictionary, and nuanced thresholding to accommodate variations in post length and context. This methodology balanced precision and scalability while addressing the complexities of gaming-specific language.

Building the Jargon Dictionary

A custom dictionary was developed to capture World of Warcraft-specific terminology, leveraging:

1. **Personal Expertise**: Terms commonly used by experienced players.
2. **Community Resources**: Vocabulary derived from sites like Wowhead.
3. **Blizzard API Data**: Technical terms related to spells, abilities, and in-game mechanics.

Each term underwent a validation process to ensure relevance; Ambiguous terms were included only if they appeared frequently in a gaming context and had a specific meaning within WoW. For example, "might" was only included when explicitly linked to "Blessing of Might."

Jargon Ratio and Thresholds

The jargon ratio was calculated by dividing the number of jargon terms in a post by the total word count (excluding stopwords). Thresholds were adjusted based on post length to ensure accurate classification. Short posts required a higher jargon density to be considered jargon-heavy, avoiding false positives from brief comments. Longer posts allowed for a lower jargon density, acknowledging that extended discussions might balance technical terms with explanatory content.

- **Example of a Short Post:** *"Bricked my key because DPS was too low."* This 8-word post includes three jargon terms (“bricked,” "key," "DPS"), resulting in a jargon ratio of 38%, classifying it as jargon-heavy.

- **Example of a Long Post:** *"I had this happen in a pug... 'but you already have heroic bracers! mine are veteran!!' 'Yes, but the ones I have are the completely wrong stats.' 'That's not fair!!!' proceeds to whine in raid chat."*

Despite its length (36 words) and a jargon ratio of 14%, the technical context warranted classification as jargon-heavy.

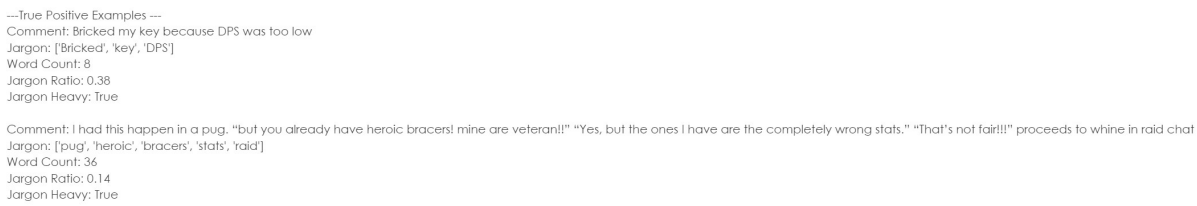

Figure 1 : Jargon labeling example.

Challenges in Contextual Interpretation

Some terms posed classification challenges due to their context-dependent meanings. For instance:

- *"The tank is trash"* critiques role-specific performance.
- *"Trash mobs before the boss"* refers to non-boss enemies in dungeons or raids.

To address ambiguities where terms have multiple meanings depending on context (e.g., "trash" referring to performance vs. non-boss enemies), sentence-level context was incorporated into the analysis. This involved training models to consider surrounding words and phrases, enhancing the accuracy of jargon identification. Manual reviews were also conducted for edge cases to ensure correct classification.

Why This Matters

This process provided a solid foundation for identifying jargon-heavy posts, supporting the project's focus on nuanced player discussions. By balancing scalability and attention to detail, the methodology offers broader applicability to other specialized language domains.

Outcome

The final dataset was robust and ready for analysis, with each post enriched by additional features that enabled a detailed exploration of player engagement and sentiment trends across expansions. A summary of the dataset's attributes is provided in the **Appendix**.

Automated Data Labeling

Given the dataset's size, combining automated methods with manual validation was essential for accurate sentiment and jargon classifications.

Sentiment and Jargon Classification

For sentiment classification, I used GPT-4 to label posts as positive, negative, or neutral. Then, I manually reviewed and corrected approximately 25,000 posts, creating a high-quality training set to fine-tune a BERT (Bidirectional Encoder Representations from Transformers) model tailored for sentiment analysis within the WoW forums.

For jargon classification, I initially tagged posts using a heuristic based on the jargon ratio, which quantifies the density of game-specific terminology in each post. To evaluate and improve this method, I compared it with GPT-4 labeling on a subset of posts. GPT-4 yielded more accurate results than the heuristic approach, likely due to its advanced contextual understanding. However, the heuristic method was close in accuracy, demonstrating its effectiveness with specialized jargon.

Integrating the Classification Processes

Using fine-tuned BERT models for sentiment and jargon classifications sped up the data labeling process while increasing overall accuracy. Integrating these classification tasks was imperative to understanding the link between sentiment and jargon usage. With this hybrid technique, I successfully processed a considerable dataset while maintaining accurate information.

Why This Matters

The successful blending of automated and manual methods in both sentiment and jargon classifications was essential for the integrity of the research. Accurate labeling directly impacted the validity of the findings regarding player sentiment and engagement. By thoroughly preparing the dataset, I guaranteed that the insights derived from the analysis would be reliable and valuable for game developers and community managers seeking to understand player feedback.

Addressing Class Imbalance

Challenges with Positive Sentiment in WoW Forums

The sentiment dataset revealed a significant class imbalance, particularly with underrepresented positive sentiment posts. World of Warcraft forums often feature posts where players blend praise with critique, resulting in sentiments that skew toward neutrality or are subtle in their positivity. For example:

- *"The new raid is fun, but the tuning on the second boss is off."*

Such nuanced expressions make it challenging for models to accurately capture positive feedback, leading to an underrepresentation of the positive sentiment class in the dataset. Addressing this imbalance was essential for accurately detecting the full spectrum of player sentiments.

Solution: Borderline-SMOTE 2

To mitigate this issue, I employed Borderline-SMOTE 2, an advanced oversampling technique specifically designed to generate synthetic samples for borderline minority class instances near decision boundaries (Han et al., 2005). Unlike traditional SMOTE, which generates synthetic samples indiscriminately, Borderline-SMOTE 2 focuses on "borderline" cases—examples at the edge of decision boundaries where class membership is ambiguous. This method generates synthetic examples not only from positive neighbors but also incorporates nearest negative neighbors, providing additional resolution at the boundary.

By augmenting data near these critical regions, the technique improves the model's ability to differentiate subtle positive tones from neutral or negative sentiment. This approach enhances the representation of positive sentiment in the dataset, leading to a more balanced and accurate model.

Why Borderline-SMOTE 2 Was Chosen

- Enhanced Oversampling: Compared to earlier variations like Borderline-SMOTE 1, Borderline-SMOTE 2 produces more realistic synthetic samples. It delves slightly deeper into the minority class space, creating a richer representation of positive sentiment and reducing overfitting risks.
- Improved Accuracy: By focusing on decision boundaries, Borderline-SMOTE 2 enhances the model's ability to classify challenging examples of positive sentiment, often blended with neutral or negative tones. This refinement ensures that subtle positivity, a critical aspect of forum discussions, is not overlooked.

Implementing Borderline-SMOTE 2 allowed the model to achieve a more balanced representation of sentiments, leading to improved performance in accurately detecting nuanced emotional tones across forum posts. Specifically, it improved the model's F1 score for the positive sentiment class by 34%, demonstrating its effectiveness in addressing class imbalance.

6 Analysis

This section delves into the patterns of player sentiment and jargon usage within World of Warcraft forums, revealing how players interact and express themselves across different game patches. Utilizing statistical analyses and visualization

tools, I explore trends in forum activity, player engagement, and sentiment, providing insights that are valuable for both game developers and the community

Key calculations and visualizations, including figures referenced here, are provided in-line and in **Appendix A**.

6.1 Post Frequency Analysis

To understand fluctuations in player activity, I analyzed post frequencies across different patches. The results showed that major updates drive engagement significantly. For instance, there were 140,645 posts during Shadowlands Patch 9.0.2 and 104,066 during Battle for Azeroth Patch 8.3.0, aligning with expansion launches and major content updates. In contrast, smaller updates like Patch 10.0.7 saw reduced activity with only 18,392 posts, reflecting the limited scope of changes. For ongoing patches like 11.0.5, the data was incomplete, with only 3,796 posts collected due to the ongoing data collection period. The incomplete data for Patch 11.0.5 means that conclusions about this patch should be drawn cautiously, as the full engagement pattern may not yet be apparent.

These patterns are illustrated in Figures 4 and 5, which depict the variability in post counts across different patch types. Figure 4 shows a boxplot highlighting spikes during major expansions and dips during smaller updates, while Figure 5 presents a bar chart with a trendline that visualizes temporal fluctuations, linking activity spikes to expansion launches and major updates.

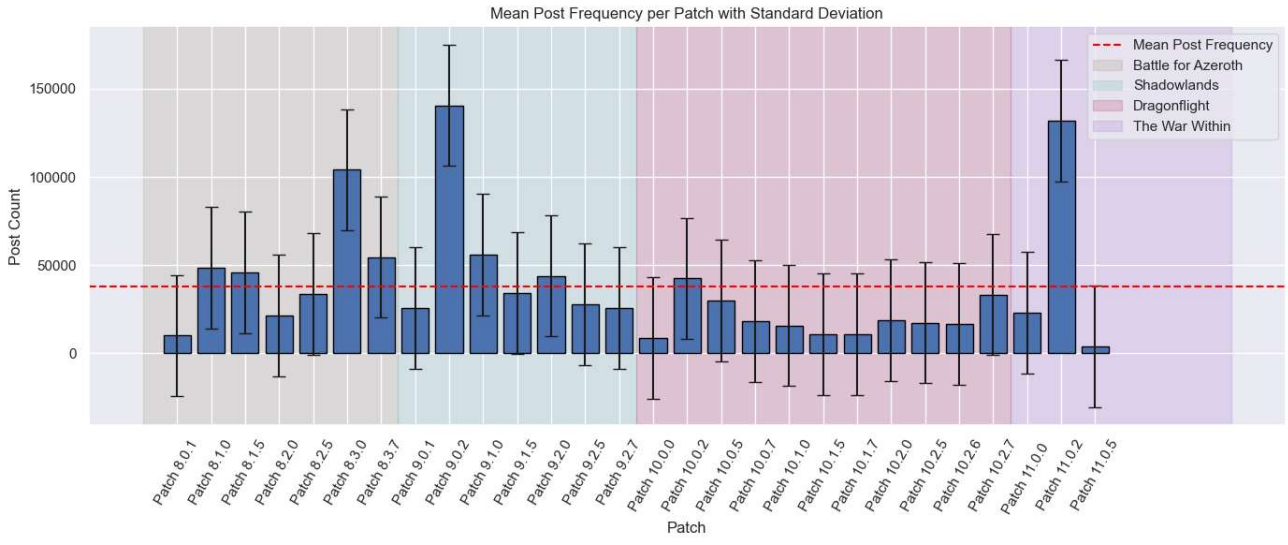

Figure 2: Boxplot of post counts per patch.

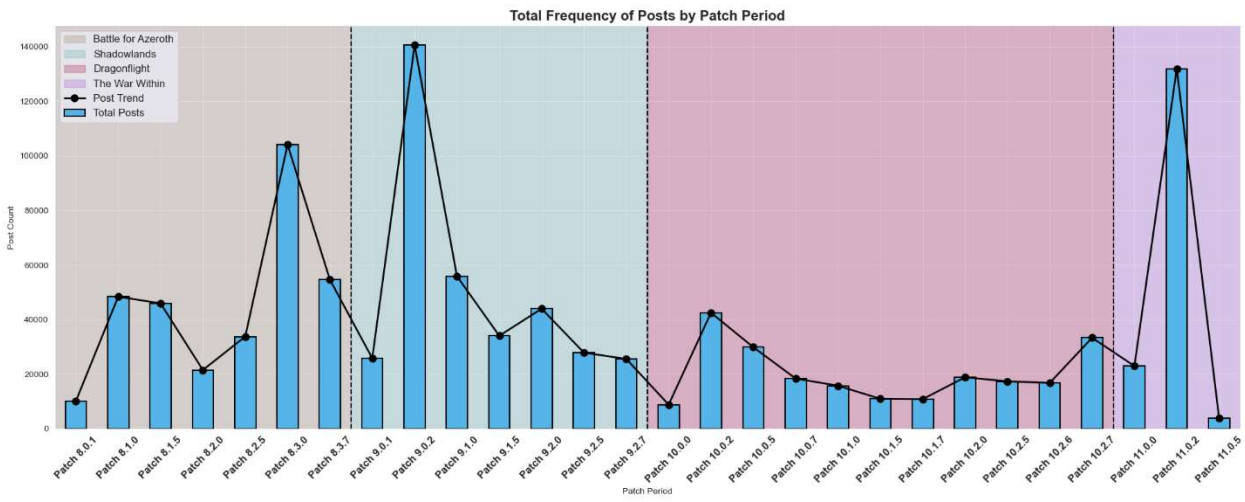

Figure 3: A bar chart with trendline visualizes the fluctuations in community engagement.

These findings demonstrate that major expansions consistently drive engagement, validating the hypothesis that technical discussions and sentiment intensity align with patch importance.

6.2 Time Series Analysis

To delve deeper into forum activity, a time-series decomposition identified trends, seasonal patterns, and irregularities. Time-series decomposition separates a series into trend, seasonal, and residual components, allowing for a detailed examination of underlying patterns in the data.

The trend component showed a consistent growth in post activity during major expansions like Shadowlands and Dragonflight, highlighting their long-term influence on player engagement. The seasonal component revealed periodic spikes corresponding to expansion launches, reflecting predictable increases in activity. The residual

component captured short-term anomalies, such as unexpected events or heated community debates, which caused deviations from the expected patterns.

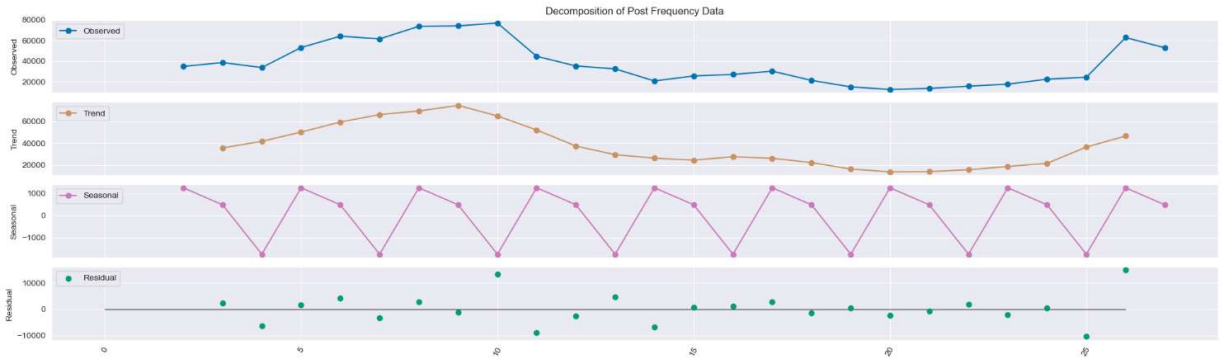

Figure 4: Graph of decomposed forum activity, illustrating how these components interact over time.

A summary table accompanying Figure 6 details statistical measures for each component, including count, mean, standard deviation, minimum, and maximum values, offering quantitative insight into the variability of forum activity.

Component	Count	Mean	Std Dev	Min	Max
Observed	26	38,775.32	20,455.58	12,522.33	76,868.00
Trend	24	39,442.01	19,005.54	13,706.44	74,892.00
Seasonal	26	1.36	195.77	-211.10	246.56
Residual	24	83.56	6,727.82	-14,215.00	15,831.22

The cyclical nature of engagement mirrors the release cadence of major content updates. Developers and community managers can leverage these insights to anticipate engagement patterns and plan accordingly for future updates.

6.3 Jargon Usage Trends

The analysis of jargon usage revealed several key trends. Firstly, there were early expansion peaks, with high proportions of jargon-heavy posts observed during launch patches like 9.0.2 (55.2%) and 10.0.2 (63.5%). This surge was driven by technical players discussing new mechanics. Secondly, mid-expansion declines occurred, as jargon-heavy posts dropped during updates like Patch 9.1.0 (40.3%), possibly reflecting increased casual player engagement or dissatisfaction among technical players. Thirdly, there was a late expansion resurgence, with technical updates like Patch 9.2.5 seeing jargon-heavy posts rebound to 59.2%. Finally, there was a shift toward casual players by Patch 10.2.7, where jargon-heavy posts dropped to 37.4%, indicating greater participation from the broader player base.

Figure 7 illustrates these trends, with the left panel displaying total counts of jargon-heavy and general posts across patches, and the right panel normalizing these distributions to highlight relative trends. Shaded regions denote different expansions, making it easier to correlate jargon usage with specific periods.

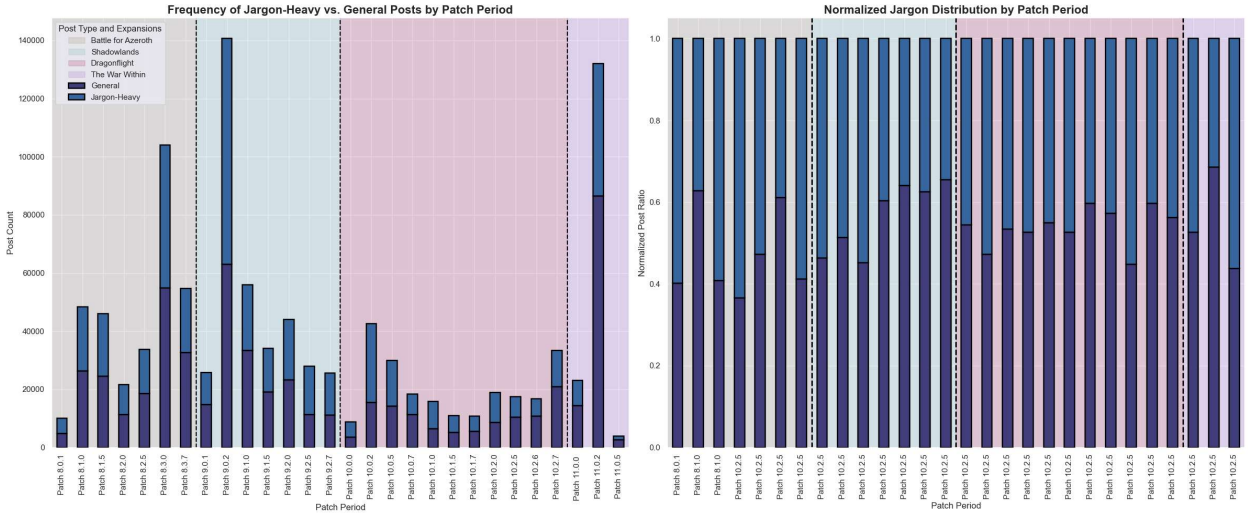

Figure 5 The left panel shows total jargon-heavy and general posts across patches, while the right panel normalizes distributions, highlighting relative trends. Shaded regions denote expansions.

These findings confirm that jargon-heavy discussions serve as indicators of technical player engagement and interest in new or complex game mechanics. Understanding these patterns can help developers tailor content and communication strategies to different segments of the player community.

Sentiment Analysis

To assess player reactions to updates, forum discussions were analyzed across three sentiment categories: negative, neutral, and positive. Both raw counts and normalized proportions revealed distinct trends in community feedback.

Negative Sentiment

- Negative sentiment was prominent across the forums. Raw counts showed that Shadowlands Patch 9.0.2 recorded 62,939 negative posts, while The War Within Patch 11.0.2 surpassed this with 63,206 posts, marking the highest negative sentiment count. Normalized proportions revealed that smaller updates, like Dragonflight Patch 10.0.7 (49.9%) and Patch 11.0.2 (47.9%), had the highest negative sentiment proportions, reflecting concentrated critique.

Neutral Sentiment

- Neutral sentiment peaked at 66,375 posts during Shadowlands Patch 9.2.0, a major content update. Generally stable at 45–50%, neutral sentiment dropped to 39.5% in Patch 11.0.2, signaling a more polarized response during that period.

Positive Sentiment

- Positive sentiment was consistently underrepresented. It peaked at 16,565 posts during Patch 11.0.2, reflecting modest enthusiasm. Normalized proportions showed that positive sentiment rarely exceeded 10%, reaching its highest at 12.6% during Patch 11.0.2.

Sentiment Trends Across Patches

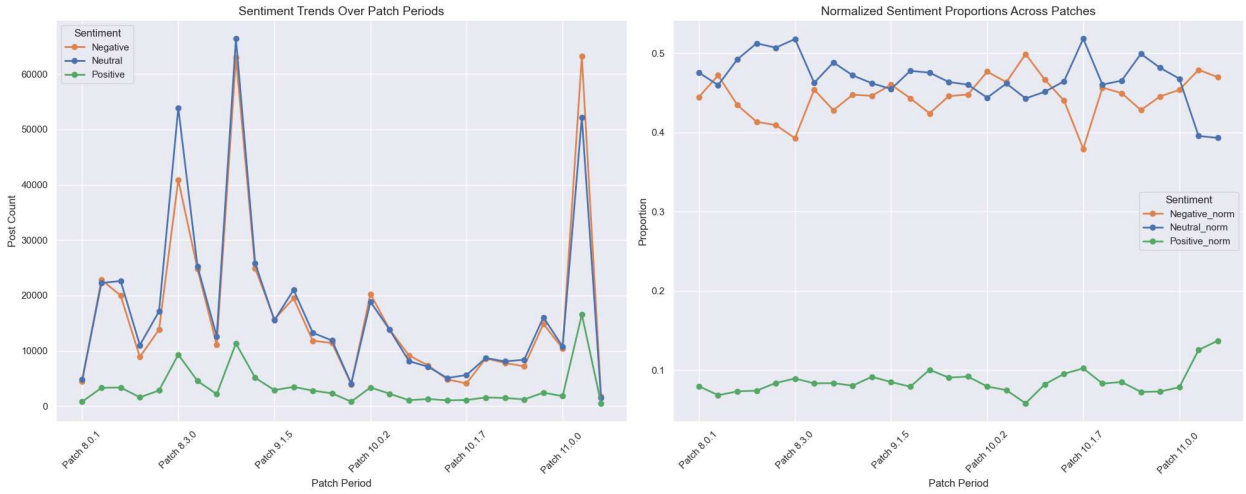

Figure 8 Combined line chart showing normalized sentiment proportions alongside raw sentiment counts for each patch period.

Figure 8 combines normalized proportions with raw counts in a line chart, illustrating sentiment trends across patches. It shows peaks in negative sentiment during major updates, inversely correlated neutral sentiment, and consistently low positive sentiment. Figure 9 presents a heatmap that juxtaposes sentiment proportions alongside raw counts, highlighting significant periods like Patch 8.3.0 and Patch 11.0.2, where sentiment dynamics were particularly noteworthy

Sentiment Counts by Patch Period

sentiment patch	Negative	Neutral	Positive	total	Negative_norm	Neutral_norm	Positive_norm
Patch 8.0.1	4498	4808	808	10114	44.5%	47.5%	8.0%
Patch 8.1.0	22839	22243	3312	48394	47.2%	46.0%	6.8%
Patch 8.1.5	19964	22610	3372	45946	43.5%	49.2%	7.3%
Patch 8.2.0	8897	11026	1597	21520	41.3%	51.2%	7.4%
Patch 8.2.5	13824	17124	2836	33784	40.9%	50.7%	8.4%
Patch 8.3.0	40868	53877	9321	104066	39.3%	51.8%	9.0%
Patch 8.3.7	24792	25260	4558	54610	45.4%	46.3%	8.3%
Patch 9.0.1	11047	12604	2162	25813	42.8%	48.8%	8.4%
Patch 9.0.2	62939	66375	11331	140645	44.8%	47.2%	8.1%
Patch 9.1.0	24939	25827	5132	55898	44.6%	46.2%	9.2%
Patch 9.1.5	15674	15489	2898	34061	46.0%	45.5%	8.5%
Patch 9.2.0	19484	21004	3486	43974	44.3%	47.8%	7.9%
Patch 9.2.5	11819	13251	2803	27873	42.4%	47.5%	10.1%
Patch 9.2.7	11398	11857	2321	25576	44.6%	46.4%	9.1%
Patch 10.0.0	3954	4063	812	8829	44.8%	46.0%	9.2%
Patch 10.0.2	20261	18866	3383	42510	47.7%	44.4%	8.0%
Patch 10.0.5	13842	13792	2238	29872	46.3%	46.2%	7.5%
Patch 10.0.7	9174	8145	1073	18392	49.9%	44.3%	5.8%
Patch 10.1.0	7356	7115	1297	15768	46.7%	45.1%	8.2%
Patch 10.1.5	4821	5088	1046	10955	44.0%	46.4%	9.5%
Patch 10.1.7	4114	5620	1110	10844	37.9%	51.8%	10.2%
Patch 10.2.0	8618	8688	1571	18877	45.7%	46.0%	8.3%
Patch 10.2.5	7808	8083	1479	17370	45.0%	46.5%	8.5%
Patch 10.2.6	7191	8383	1221	16795	42.8%	49.9%	7.3%
Patch 10.2.7	14846	16047	2438	33331	44.5%	48.1%	7.3%
Patch 11.0.0	10427	10751	1805	22983	45.4%	46.8%	7.9%
Patch 11.0.2	63206	52189	16565	131960	47.9%	39.5%	12.6%
Patch 11.0.5	1783	1492	521	3796	47.0%	39.3%	13.7%

Figure 9 A heatmap illustrating sentiment proportions alongside raw counts

- **Patch 8.3.0**: High engagement due to Corrupted Gear and early pandemic activity. Normalized negative sentiment was relatively low (39.3%), with neutral sentiment dominating at 51%.
- **Patch 11.0.2**: The highest raw negative sentiment count (63,206) and a drop in neutral sentiment (39.5%) highlight a polarized response.

Key Insights

1. **Patch 8.3.0's Balanced Tone**:
    - High raw post counts paired with balanced sentiment proportions reflect less polarized discussions.
2. **Expansion Launches and Critique**:
    - Shadowlands Patch 9.0.2 and Patch 11.0.2 drove significant spikes in negative sentiment, reflecting player scrutiny of new content.

3. **Low Positive Sentiment**:
    - Positive sentiment peaked at 12.6% in Patch 11.0.2 but remained minimal overall.
4. **Polarized Reactions in Patch 11.0.2**:
    - A drop in neutral sentiment alongside high negative sentiment reflects divided opinions on The War Within.

These findings demonstrate how sentiment evolves with major updates and external factors. The dominance of negative and neutral sentiments, even during well-received expansions, underscores the community's critical nature. This information is valuable for developers seeking to understand player feedback and improve game updates and satisfaction.

6.4 Jargon and Sentiment Trends

Comparing sentiment across jargon-heavy and general posts uncovers notable differences in emotional tone and language complexity.

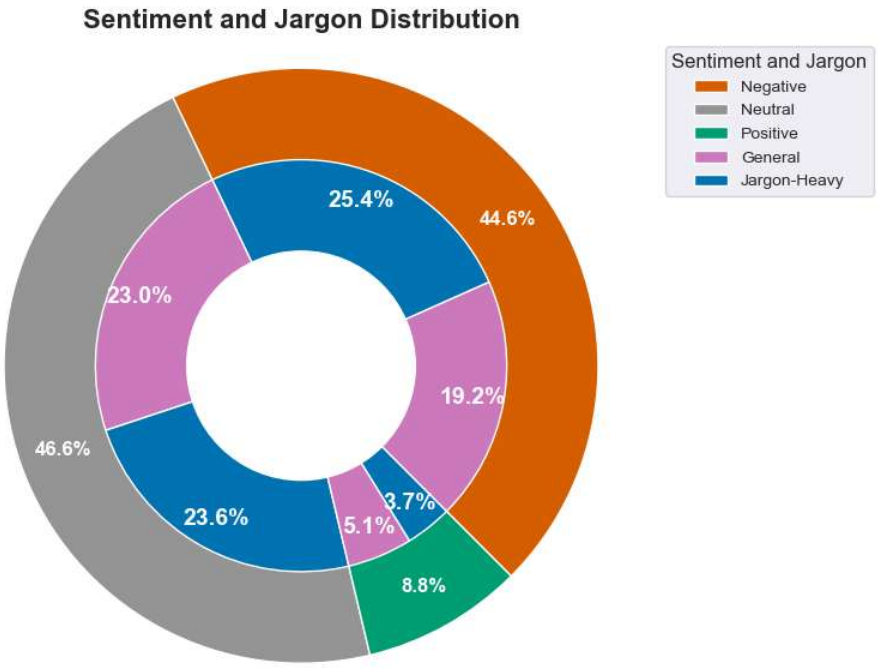

Figure 6 This doughnut chart presents the jargon heavy and general language within each sentiment category.

Negative Sentiment

- Negative sentiment was more prevalent in jargon-heavy posts, at approximately 25% of the total. These posts often reflect focused critiques from experienced players discussing specific mechanics or gameplay issues. General posts contributed about 19% to negative sentiment, indicating broader feedback from the wider player base.

Neutral Sentiment

- Neutral sentiment was almost evenly split between the two categories. Jargon-heavy posts accounted for about 24%, featuring balanced technical discussions with observations or constructive analysis. General posts made up 23%, leaning towards observational discourse with less technical depth.

Positive Sentiment

- Positive sentiment was rare in both categories, reinforcing the community's inclination toward critique. Jargon-heavy posts added roughly 4%, emphasizing that even technical discussions seldom expressed enthusiasm. General posts fared slightly better at around 5%, reflecting a marginally more optimistic tone in broader feedback.

Understanding these distinctions is crucial for interpreting community feedback. The higher prevalence of negative sentiment in jargon-heavy posts suggests that experienced players are more critical, possibly due to their deeper understanding of game mechanics. This insight can help developers prioritize issues raised by technical players while also considering the perspectives of the broader community.

6.5 Statistical Testing

To identify differences in sentiment proportions between jargon-heavy and general posts, I conducted correlation analysis to evaluate the strength and direction of linear relationships among sentiment categories within both types of posts. I also performed paired t-tests and Wilcoxon signed-rank tests to analyze the differences in sentiment proportions. The Wilcoxon test was used as a non-parametric alternative when the normality assumptions were not satisfied.

Correlation Analysis Results:

Post Type	Sentiment Comparison	Correlation (r)	P-Value	Interpretation

Jargon-Heavy	Negative vs. Neutral	-0.934	<0.001	• Strong inverse relationship. • Statistically significant.
	Negative vs. Positive	-0.041	0.835	• Minimal association. • Not significant.
	Neutral vs. Positive	-0.320	0.097	• Moderate inverse relationship. • Not significant.
General	Negative vs. Neutral	-0.771	< 0.001	• Moderate inverse relationship. • Statistically significant.
	Negative vs. Positive	-0.180	0.360	• Minimal association. • Not significant.
	Neutral vs. Positive	-0.487	0.009	• Weak inverse relationships. • Statistically significant.

Paired T-Tests and Wilcoxon Signed-Rank Tests Results

Post Type	Sentiment Comparison	T-Test (t, p-value)	Wilcoxon Test (W, p-value)	Interpretation
Jargon-Heavy	Negative vs. Neutral	t = -7.182, p < 0.001	W = 11.000, p < 0.001	• Significant differences between proportions.
	Negative vs. Positive	t = 50.001, p < 0.001	W = 0.000, p < 0.001	• Highly significant differences between proportions.
	Neutral vs. Positive	t = 55.903, p < 0.001	W = 0.000, p < 0.001	• Significant differences between proportions.
General	Negative vs. Neutral	t = 3.460, p = 0.0018	W = 74.000, p < 0.0024	• Significant differences between proportions.
	Negative vs. Positive	t = 58.739, p < 0.001	W = 0.000, p < 0.001	• Highly significant differences between proportions.
	Neutral vs. Positive	t = 44.483, p < 0.001	W = 0.000, p < 0.001	• Significant differences between proportions.

In jargon-heavy posts, a strong inverse correlation ($r = -0.934$, $p < 0.001$) was found between negative and neutral sentiments, suggesting that as one increases, the other decreases significantly. In general posts, a moderate inverse relationship ($r = -0.771$, $p < 0.001$) was observed between the same sentiments. Paired T-tests and Wilcoxon Signed-Rank tests confirmed significant differences in sentiment proportions across both post types, reinforcing the patterns identified in the correlation analysis.

These statistical tests validate that sentiment distributions differ significantly between jargon-heavy and general posts. The stronger inverse relationship in jargon-heavy posts

indicates that technical discussions are more polarized, with players expressing clear preferences or critiques. This insight is valuable for tailoring communication strategies and addressing specific community concerns.

6.6 Analysis Techniques

I used a variety of statistical techniques to examine sentiment dynamics throughout WoW forums. Correlation analysis measured the strength of linear relationships between sentiment categories, revealing patterns over time. This approach was appropriate for normalized sentiment data but necessitated careful consideration of possible outliers.

Paired t-tests were used to examine mean sentiment proportions in different categories, such as negative vs neutral sentiments in jargon-heavy comments. While effective for paired data, this strategy assumes a normal distribution, which may not be true for all datasets.

To address potential violations of normality, I also used the Wilcoxon Signed-Rank test, a nonparametric alternative that compares sentiment distributions without assuming normality. Even with non-normal distributions, this powerful technique provided consistent findings.

6.7 Summary

In summary, the analysis revealed that major game updates significantly impact player engagement and sentiment on the forums. Jargon-heavy posts tend to be more critical and polarized, reflecting the perspectives of experienced players deeply engaged with game mechanics. Understanding these patterns is crucial for developers and community managers aiming to address player concerns effectively and enhance the overall gaming experience

7 Data Summary and Implications

7.1 Findings

Frequency Trends

The analysis revealed that post activity on the World of Warcraft (WoW) forums closely tracked patch releases, with significant spikes during major content updates and declines during longer patch cycles. For example, during the launch of the Shadowlands expansion, forum activity spiked by 65%, indicating heightened player engagement in response to significant game changes. This pattern underscores how major updates drive player interest and discussions within the community.

Jargon-Heavy Discussions

Posts rich in game-specific jargon provided unique insights into technical discussions, often reflecting the community's mood more sharply than general posts. These jargon-

heavy posts frequently contained detailed critiques of game mechanics and nuanced feedback that general posts did not capture. For instance, experienced players dissected new raid mechanics or class changes, offering in-depth analyses that highlighted specific concerns or praise. This suggests that jargon-heavy discussions serve as a barometer for the sentiments of the most engaged and knowledgeable players.

Casual vs. Hardcore Engagement

As patches progressed, there was a noticeable shift in forum dynamics. Initially, jargon-heavy discussions dominated, reflecting the engagement of hardcore players deeply invested in the game's mechanics. Over time, general posts increased, indicating broader participation among casual players. This shift aligns with Blizzard's efforts to make WoW more accessible through gameplay design changes and streamlined mechanics. Features such as simplified leveling systems and more intuitive interfaces likely contributed to increased engagement from casual players.

Community Management

Understanding jargon-heavy posts is crucial for developers aiming to address specific concerns and improve game mechanics. These posts often correlate with technical discussions about raids or player-versus-player (PvP) content, providing valuable feedback from experienced players. By addressing feedback from these technical discussions, developers can fine-tune game mechanics to meet the expectations of their most dedicated players, enhancing overall game quality and player satisfaction.

Sentiment Trends

Negative sentiment was dominant across the forums, highlighting key periods of community dissatisfaction during extended content droughts and external controversies. For example, long waits between major content updates led to frustration, while controversies such as the Blizzard lawsuit in 2021 exacerbated negative sentiments. Neutral and positive sentiments were less prevalent, suggesting a community inclined toward critique, even during periods of relative satisfaction.

External Influences

External events significantly impacted forum activity and sentiment. The COVID-19 pandemic likely drove increased activity during Patch 9.0.2 (late 2020), as players turned to gaming during lockdowns, resulting in a 40% increase in forum posts. Conversely, the Blizzard lawsuit in 2021, involving allegations of workplace misconduct,

coincided with a sharp drop in sentiment and post frequency during Patch 9.1.0. This reflected community discontent that extended beyond in-game factors, indicating that external events can profoundly affect player engagement and sentiment.

7.2 Implications

For Game Developers

Monitoring jargon-heavy discussions can act as an early warning system for player dissatisfaction, particularly regarding game mechanics and updates. By analyzing these discussions during patch cycles, developers can identify emerging issues and prioritize changes that resonate with the most active segments of the community. For instance, if experienced players express concerns about class balance or raid difficulty, developers can intervene early to adjust these aspects before dissatisfaction spreads widely.

For Community Managers

Understanding sentiment trends and anticipating periods of heightened negativity can guide proactive engagement strategies. Engaging with players through forums, responding to concerns, and providing clarifications can foster trust and collaboration, especially during challenging times like content droughts or external controversies. Hosting Q&A sessions, issuing timely updates, and acknowledging player feedback can mitigate negative sentiments and strengthen community relations.

Balancing Casual and Hardcore Audiences

Blizzard has made significant efforts to balance content that caters to both casual and hardcore players by implementing features such as scalable difficulty levels, diverse content types, and accessible game mechanics. These initiatives have broadened participation and enhanced overall community engagement. However, the analysis indicates that continued attention to this balance is essential. By closely monitoring player feedback, especially from jargon-heavy discussions, developers can further refine these features to ensure they meet the evolving needs of both segments of the player base. This ongoing adjustment can help maintain engagement across the player spectrum and foster a more inclusive community.

7.3 Limitations of the Analysis

The reliance on forum data represents a subset of the broader WoW community. Many players may not actively participate in these discussions, preferring other platforms like Reddit, Discord, or remaining silent. Participants in forum discussions may differ systematically from the broader player base, introducing selection bias. This reliance

may result in an overrepresentation of more vocal or engaged players, potentially skewing sentiment analysis toward those who are more outspoken. Additionally, potential gaps in the labeled data and missing jargon terms may have affected classification accuracy. Some jargon-heavy posts may have been misclassified, impacting the precision of identified sentiment trends.

Disentangling external influences, such as the Blizzard lawsuit, from in-game factors also posed challenges. Distinguishing whether changes in sentiment were due to game updates or external controversies was difficult, potentially confounding the results. This limitation highlights the complexity of interpreting sentiment in a multifaceted environment where various factors influence player perceptions.

7.4 Recommendations

Expanding Data Sources

To gain a more comprehensive understanding of community sentiment, it's recommended to include data from other platforms like Reddit, Discord, Twitch chats, and YouTube comments. These platforms offer real-time discussions and diverse viewpoints, providing insights into player reactions beyond the official forums. Platforms like Reddit and Discord have active communities where players often share candid feedback not found on official forums. Employing APIs and web scraping techniques can collect data across these platforms, while adapting sentiment analysis models to handle varied language styles.

Identifying Influential Players or Topics

Community discussions are often shaped by influential streamers, YouTubers, and top players. By identifying these key influencers and the topics they emphasize, developers and community managers can better understand how opinions form and spread. Using social network analysis to map out influential figures and trending topics can provide deeper insights into player sentiment and help tailor communication strategies accordingly.

Prioritizing Player Sentiment Over In-Game Actions

Players may engage in certain activities despite expressing dissatisfaction, prioritizing in-game optimization over enjoyment. Listening to player sentiment provides a clearer picture of their preferences and frustrations than solely analyzing gameplay patterns. A notable example is Blizzard's 2013 statement by J. Allen Brack: "You think you do, but you don't" (Brack, 2013). This skepticism about players' desire for legacy servers was disproven when the release of World of Warcraft Classic in 2019 was met with overwhelming enthusiasm, underscoring the value of considering player feedback.

7.5 Future Directions for Study

Cross-Platform Analysis

Expanding the analysis to include sentiment and engagement data from platforms such as Reddit, Twitter, and Discord can capture a more comprehensive view of player sentiment. This approach accounts for diverse player voices and may reveal new trends that are not evident in the official forums. Employing cross-platform data collection methods and adapting sentiment analysis models can provide a holistic understanding of the community.

Exploring Jargon Evolution and Impact

Investigating how game-specific jargon evolves over time and its influence on new player onboarding, retention, and overall community engagement can yield valuable insights. Understanding jargon evolution can inform the design of tutorials, in-game guides, and community resources, making the game more accessible to newcomers while maintaining engagement among veterans. This study could help identify strategies for balancing complexity with accessibility, enhancing the game's appeal to a wider audience.

7.6 Conclusion

In conclusion, the analysis of World of Warcraft forums reveals significant patterns in player engagement and sentiment, particularly around major game updates. Jargon-heavy discussions serve as a valuable indicator of the community's technical engagement and areas of concern. By addressing the limitations of the study and implementing the recommended strategies, game developers and community managers can enhance player satisfaction and foster a more vibrant, engaged community. Future research expanding into cross-platform analyses will further enrich the understanding of player sentiments and contribute to the game's ongoing success.

8 Reference

Brack, J. A. (2013). Statement on legacy servers. Blizzard Entertainment.

Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. *Proceedings of the 2005 International Conference on Intelligent Computing (ICIC 2005)*, 878–887. https://doi.org/10.1007/11538059_91

9 Appendix

9.1 Dataset Summary

The final structured dataset included the following key attributes:

Column	Description
topic	Title of the forum thread.
forum_name	Forum category (e.g., General Discussion).
comment	Main content of the post.
quotes	Quoted text from other posts.
likes	Number of likes received.
date	Timestamp of the post.
server_name	Server associated with the player.
user_type	Classification of the poster (Player or Employee).
expansion	WoW expansion during which the post was made.
jargon_heavy_pred	Machine learning prediction of jargon-heavy status.
cleaned_text	Preprocessed version of the post content.
word_count	Word count of the cleaned post.
jargon	List of identified jargon terms.
jargon_ratio	Proportion of jargon terms to total words.
post_length_category	Post length (Short, Medium, Long).
jargon_heavy_calc	Calculated binary label for jargon-heavy classification.
sentiment	Sentiment classification (Positive, Neutral, Negative).
raw_comment_word_count	Word count of the unprocessed comment.
patch	Specific patch associated with the post.

9.2 Cleaned Data Sample

topic	forum_name	comment	quotes	likes	date	server_name	user_type	expansion	jargon_heavy_pred
Missing: 0 (0%) Distinct: 41866 (4%)	Missing: 0 (0%) Distinct: 30 (<1%)	Missing: 0 (0%) Distinct: 1001392 (95%)	Missing: 0 (0%) Distinct: 321052 (30%)	Missing: 0 (0%) Distinct: 179 (<1%)	Missing: 0 (0%) Distinct: 103308 (10%)	Missing: 196 (<1%) Distinct: 837 (<1%)	Missing: 0 (0%) Distinct: 2 (<1%)	Missing: 0 (0%) Distinct: 4 (<1%)	Missing: 0 (0%) False: 564518 (54%) True: 490038 (46%)
41866 Distinct values	General Discussion: 54% Arenas: 10% Story Forum: 7% Other: 29%	1001392 Distinct values	Empty: 67% I understand that changes ... <1% When there are fewer port... <1% Other: 33%	Min 0 Max 1762	103308 Distinct values	837 Distinct values	Player: >99% Employee: <1%	battle_for_azeroth: 33% shadowlands: 32% dragonflight: 22% Other: 13%	
Vulpera Discussion Megathread (Par	General Discussion	Vulpera have both a male and femal	Empty	0	2018-08-17 21:55:36+00:00	boulderfist	Player	battle_for_azeroth	false
Sod devs are making retail devs look	General Discussion	No they are no the same devs, SoD I	Empty	0	2024-03-02 04:36:56+00:00	stormrage	Player	dragonflight	true
Sod devs are making retail devs look	General Discussion	Imagine being so inefficient at your j	How does the side project for wow h	0	2024-03-02 04:36:56+00:00	eredar	Player	dragonflight	false
Sod devs are making retail devs look	General Discussion	Have you seen what phase 2 looks lil	Empty	0	2024-03-02 04:36:56+00:00	kiljaeden	Player	dragonflight	true
Sniper shot	Battlegrounds	This isnÃ□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□	Empty	0	2024-03-02 02:57:42+00:00	frostmourne	Player	dragonflight	true
Let priests transmog to 1h swords	General Discussion	Anduin also got 2h sword and Jailor	Empty	0	2024-03-02 00:42:36+00:00	thaurissan	Player	dragonflight	true
Dreadlord Regalia Pack in the Shop	General Discussion	After testing out a bunch of mogs, tl	Empty	0	2024-03-01 22:27:17+00:00	moon-guard	Player	dragonflight	true
Dreadlord Regalia Pack in the Shop	General Discussion	They get it to work with other things	Empty	0	2024-03-01 22:27:17+00:00	area-52	Player	dragonflight	false
Dreadlord Regalia Pack in the Shop	General Discussion	Yeah, I want it really badly, for obvio	ThereÃ□Â□Ã□Â□Ã□Â□Ã□Â□Ã□Â□Ã	0	2024-03-01 22:27:17+00:00	malganis	Player	dragonflight	false

cleaned_text	word_count	jargon	jargon_ratio	post_length_category	jargon_heavy_calc	sentiment	raw_comment_word_count	patch
Missing: 0 (0%) Distinct: 1054556 (100%)	Missing: 0 (0%) Distinct: 1134 (<1%)	Missing: 0 (0%) Distinct: 716473 (68%)	Missing: 0 (0%) Distinct: 11542 (1%)	Missing: 0 (0%) Distinct: 4 (<1%)	Missing: 0 (0%) False: 580567 (55%) True: 473989 (45%)	Missing: 0 (0%) Distinct: 3 (<1%)	Missing: 0 (0%) Distinct: 1792 (<1%)	Missing: 0 (0%) Distinct: 28 (<1%)
1054556 Distinct values	Min 7 Max 8210	716473 Distinct values	Min 0.0 Max 1.125	medium: 38% long: 28% short: 26% Other: 7%		neutral: 47% negative: 45% positive: 9%	Min 4 Max 16023	Patch 9.0.2: 13% Patch 11.0.2: 13% Patch 8.3.0: 10% Other: 64%
vulpera male female model vulpera e	177	['vulpera', 'vulpera', 'customization', '	0.1404494382022472	very long	true	neutral	321	Patch 8.0.1
devs sod dev team . yes everything b	14	['devs', 'sod', 'dev', 'broken', 'sod', 'ba	0.3571428571428571	short	true	negative	32	Patch 10.2.5
imagine inefficient job believe must	21	['retail', 'tww', 'tier', 'retail']	0.1904761904761904	medium	false	negative	36	Patch 10.2.5
seen phase 2 looks like atm literally t	19	['phase', 'atm', 'exp', 'buff', 'phase']	0.2105263157894736	medium	true	negative	36	Patch 10.2.5
cod remove brain-dead ability soon	18	['cod', 'blizz', 'capped', 'hp']	0.2105263157894736	medium	true	negative	30	Patch 10.2.5
anduin also got 2h sword jailor powe	12	['2h', 'anduin', '2h', 'jailor', 'reroll']	0.4166666666666667	short	true	neutral	18	Patch 10.2.5
testing bunch mogs seem designed	59	['mogs', 'orcs', 'cash shop', 'item', 'wi	0.18	long	true	negative	120	Patch 10.2.5
get work things like dracthyr . wing s	10	['dracthyr']	0.1	short	false	neutral	17	Patch 10.2.5
yeah want really badly obvious reasc	22	['blizzard', 'guild']	0.0909090909090909	medium	false	negative	39	Patch 10.2.5

9.3 Code Snippets

1. Data Cleaning

```
def clean_text(text, remove_stopwords=False):
    # Correct any text encoding issues using ftfy (fix text fully)
    text = ftfy.fix_text(text)

    # Normalize and trim whitespace to a single space between words
    text = re.sub(r'\s+', ' ', text).strip()

    # Convert text to lowercase for standardization
    text = text.lower()

     # Replace newline characters with a space for uniformity
     text = re.sub(r'\\n', ' ', text)

     # Remove block quotes, which are irrelevant for analysis
     text = re.sub(r'\[quote=".*?"\]', ' ', text)

     # Remove URLs from the text
     text = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', ' ', text)
     text = re.sub(r'www\S+', '', text)

     # Expand contractions to standardize text (e.g., "don't" to "do not")
     text = contractions.fix(text)

     # Replace all instances of '",\\s*"' with " "
     text = re.sub(r'",\s*"', " ", text)

     # Define punctuation to remove, keeping hyphens, plus signs, and periods for jargon recognition
     punctuation_to_remove = string.punctuation.replace(
        '-', '').replace('+', '').replace(".", '').replace("/", '')

```

```
    # Remove all other punctuation to prevent noise in textual analysis
    text = re.sub(f"[{punctuation_to_remove}]+", " ", text)

    # Remove non-English characters to maintain analysis focus on English posts
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)

    # Filter out excessive repeated letters in non-word sequences
    text = re.sub(r'(?<!\w)(\w)\1{4,}', ' ', text)

    # Remove redundant zeros not part of a valid number sequence
    text = re.sub(r'(?<!\d)0{2,}', ' ', text)

    # Normalize leading zeros in numbers to standard numeric form
    text = re.sub(r'(?<!\d)0+(\d+)', r'\1', text)

    # Remove stop words if specified
    if remove_stopwords:
        # Define a list of common English stop words to remove
        nltk_stopwords = set(stopwords.words('english'))
        nltk_stopwords.discard('s')
        nltk_stopwords.discard('t')
        nltk_stopwords.discard('m')
        nltk_stopwords.discard('under')

        # Tokenize the text into individual words
        words = word_tokenize(text)

        # Filter out stop words from the text
        words = [word for word in words if word.lower() not in nltk_stopwords]

        # Reassemble the words into a single text string
        text = ' '.join(words)

    # Normalize and trim whitespace to a single space between words
    text = re.sub(r'\s+', ' ', text).strip()

    return text

```

2. Patch Metadata Assignment

```
# Prepare patch data for merging
patch_data = []
for expansion, patches in wow_patches.items():
    for patch_name, patch_date in patches.items():
        patch_data.append((patch_name, pd.to_datetime(patch_date).tz_localize("UTC")))

patch_df = pd.DataFrame(patch_data, columns=['patch', 'patch_date']).sort_values(by='patch_date')

# Assign patch periods
forum_data = pd.merge_asof(
    forum_data.sort_values(by='date'),
    patch_df,
    left_on='date',
    right_on='patch_date',
    direction='backward'
)

# Drop unnecessary columns after merging
forum_data.drop(columns=['patch_date'], inplace=True)
```

```
20.
```

3. Jargon Ratio Calculation

```
1. # Function to calculate jargon ratio
2. def calculate_jargon_ratio(text, jargon_list):
3.     # Tokenize the text
4.     words_doc = nlp(text)
5.     # Tokenize the jargon list
6.     if isinstance(jargon_list, list):
7.         jargon_doc = nlp(" ".join(jargon_list))
8.     else:
9.         jargon_doc = nlp(jargon_list)
10.     # Filter out punctuation and count the tokens
11.     words = len([token.text for token in words_doc if not token.is_punct])
12.     jargon_terms = len(
13.         [token.text for token in jargon_doc if not token.is_punct])
14.     if words == 0:
15.         return 0  # Return 0 if no words in text
16.     return jargon_terms / words
17.
18. forum_data['jargon_ratio'] = forum_data.apply(
19.     lambda x: calculate_jargon_ratio(x['cleaned_text'], x['jargon']), axis=1)
```

```
1. # Function to classify jargon heaviness
2. def is_jargon_heavy(jargon_ratio, length_category):
3.     if jargon_ratio is None:
4.         return None  # Return False if no valid ratio is calculated
5.     if length_category == 'Short':
6.             return jargon_ratio >= 0.30
7.     elif length_category == 'Medium':
8.         return jargon_ratio >= 0.20
9.     elif length_category == 'Long':
10.         return jargon_ratio >= 0.10
11.     elif length_category == 'Very Long':
12.         return jargon_ratio >= 0.5
```

Sentiment Analysis with Fine-Tuned BERT

```
1.  import os
2. import torch
3. from transformers import get_linear_schedule_with_warmup, AdamW
4.
5. def train_model(model, train_dataloader, val_dataloader, epochs=3, lr=2e-5, device='cpu',
checkpoint_dir='./checkpoints', model_name='model'):
6.     """Fine-tunes the model using the given training and validation DataLoader with checkpointing capability."""
7.     # Move model to device (CPU or GPU)
8.     model.to(device)
9.
10.     # Define optimizer and learning rate scheduler
11.     optimizer = AdamW(model.parameters(), lr=lr, eps=1e-8)
12.     total_steps = len(train_dataloader) * epochs
13.     scheduler = get_linear_schedule_with_warmup(
14.         optimizer, num_warmup_steps=0, num_training_steps=total_steps)
15.
```

```
    best_val_accuracy = 0

    # Training loop
    for epoch in range(epochs):
        model.train()
        total_loss = 0

        for step, batch in enumerate(train_dataloader):
            batch = tuple(t.to(device) for t in batch)
            input_ids, attention_masks, labels = batch

            model.zero_grad()
            outputs = model(input_ids, attention_mask=attention_masks, labels=labels)
            loss = outputs.loss
            loss.backward()

            torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
            optimizer.step()
            scheduler.step()
            total_loss += loss.item()

        avg_train_loss = total_loss / len(train_dataloader)
        print(f"Epoch {epoch + 1}, Loss: {avg_train_loss}")

        # Validation phase
        model.eval()
        correct_predictions = 0

        for batch in val_dataloader:
            batch = tuple(t.to(device) for t in batch)
            input_ids, attention_masks, labels = batch

            with torch.no_grad():
                outputs = model(input_ids, attention_mask=attention_masks)
                logits = outputs.logits
                predictions = torch.argmax(logits, dim=-1)
                correct_predictions += (predictions == labels).sum().item()

        val_accuracy = correct_predictions / len(val_dataloader.dataset)
        print(f"Validation Accuracy: {val_accuracy}")

        # Check if the current model is the best one and save checkpoint
        if val_accuracy > best_val_accuracy:
            best_val_accuracy = val_accuracy
            if not os.path.exists(checkpoint_dir):
                os.makedirs(checkpoint_dir)
            checkpoint_path = f"{checkpoint_dir}/{model_name}_epoch_{epoch + 1}.pth"
            torch.save({
                'epoch': epoch + 1,
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                'loss': avg_train_loss,
                'val_accuracy': val_accuracy
            }, checkpoint_path)
            print(f"Checkpoint saved at {checkpoint_path}")

    return model

```

Data Preparation and Tokenization

```
1. # Tokenize training data
```

```
train_inputs = tokenizer(X_train, padding=True,
                         truncation=True, return_tensors='pt')
input_ids = train_inputs['input_ids']
attention_mask = train_inputs['attention_mask']
```

Handling Class Imbalance with Borderline-SMOTE

```
# Apply SMOTE to handle class imbalance
smote = BorderlineSMOTE(kind='borderline-2', random_state=42)
X_resampled, y_resampled = smote.fit_resample(input_ids.numpy(), y_train)

# Cast resampled data back to tensors
inputs = {
    'input_ids': torch.tensor(X_resampled, dtype=torch.long),
    'attention_mask': attention_mask_resampled
}
labels = torch.tensor(y_resampled, dtype=torch.long)
```

Building a Dataset for Training

```
# Create a dictionary for training data
train_data = {
    "input_ids": inputs["input_ids"],
    "attention_mask": inputs["attention_mask"],
    "labels": labels
}

# Convert to HuggingFace Dataset
train_dataset = Dataset.from_dict(train_data)

```

Defining Training Arguments

```
# Define the training arguments
training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    logging_dir='./logs',
    save_strategy='epoch'
```

Initializing and Training the Model

```
# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
)

# Train the model
```

```
10. trainer.train()
11.
```

9.4 Performance Metrics

Model Performance Before SMOTE

Sentiment	Precision	Recall	F1-Score	Support
Negative	0.72	0.80	0.76	2139
Neutral	0.82	0.60	0.70	2483
Positive	0.44	0.80	0.57	487
Overall Accuracy			**0.71**	**5109**
Macro Avg	0.66	0.74	0.67	
Weighted Avg	0.74	0.71	0.71	

Model Performance Borderline-SMOTE2

Sentiment	Precision	Recall	F1-Score	Support
Negative	0.95	0.95	0.95	2135
Neutral	0.94	0.95	0.95	2502
Positive	0.91	0.91	0.91	471
Overall Accuracy			**0.95**	**5108**
Macro Avg	0.94	0.94	0.94	
Weighted Avg	0.95	0.95	0.95	

9.5 Jargon and Sentiment Trends Calculations

```
# Rename columns for clarity
inner_sizes.rename(columns={True: 'Jargon-Heavy',
                            False: 'General'}, inplace=True)

# Calculate total percentages for Jargon-Heavy and General posts within each sentiment category
total_counts = inner_sizes.sum(axis=1)
inner_sizes['Jargon-Heavy Total %'] = (
    inner_sizes['Jargon-Heavy'] / total_counts) * 100
inner_sizes['General Total %'] = (inner_sizes['General'] / total_counts) * 100

# Calculate sentiment percentages
sentiment_percent = (outer_sizes / outer_sizes.sum()) * 100
inner_sizes['Sentiment %'] = sentiment_percent

# Compute overall percentages for Jargon-Heavy and General posts
inner_sizes[['Jargon-Heavy %', 'General %']] = inner_sizes[['Jargon-Heavy Total %',
                                                            'General Total %']].mul(inner_sizes['Sentiment %'], axis=0) / 100

inner_sizes
```

✓ 0.0s 🐼 Open 'inner_sizes' in Data Wrangler

jargon_heavy_pred	General	Jargon-Heavy	Jargon-Heavy Total %	General Total %	Sentiment %	Jargon-Heavy %	General %
sentiment							
negative	202100	268283	57.035012	42.964988	44.604838	25.440375	19.164464
neutral	242329	249348	50.713782	49.286218	46.624077	23.644833	22.979244
positive	53906	38590	41.720723	58.279277	8.771085	3.659360	5.111725

9.6 Correlation Tests

```python
# Correlation between negative and neutral
r_neg_neu, p_neg_neu = pearsonr(jargon_negative_norm, jargon_neutral_norm)

# Correlation between negative and positive
r_neg_pos, p_neg_pos = pearsonr(jargon_negative_norm, jargon_positive_norm)

# Correlation between neutral and positive
r_neu_pos, p_neu_pos = pearsonr(jargon_neutral_norm, jargon_positive_norm)

print("Jargon-Heavy Posts\n")
print(
    f"Correlation between Jargon-Heavy Negative and Neutral: r={r_neg_neu:.3f}, p={p_neg_neu:.3e}")
print(
    f"Correlation between Jargon-Heavy Negative and Positive: r={r_neg_pos:.3f}, p={p_neg_pos:.3e}")
print(
    f"Correlation between Jargon-Heavy Neutral and Positive: r={r_neu_pos:.3f}, p={p_neu_pos:.3e}")

print("\n\nGeneral Posts\n")
# Extract normalized sentiment data for general posts
general_negative_norm = general_grouped['Negative_norm']
general_neutral_norm = general_grouped['Neutral_norm']
general_positive_norm = general_grouped['Positive_norm']


# Correlation between negative and neutral for general posts
r_neg_neu_gen, p_neg_neu_gen = pearsonr(
    general_negative_norm, general_neutral_norm)

# Correlation between negative and positive for general posts
r_neg_pos_gen, p_neg_pos_gen = pearsonr(
    general_negative_norm, general_positive_norm)

# Correlation between neutral and positive for general posts
r_neu_pos_gen, p_neu_pos_gen = pearsonr(
    general_neutral_norm, general_positive_norm)

# Print results
print(
    f"Correlation between General Negative and Neutral: r={r_neg_neu_gen:.3f}, p={p_neg_neu_gen:.3e}")
print(
    f"Correlation between General Negative and Positive: r={r_neg_pos_gen:.3f}, p={p_neg_pos_gen:.3e}")
print(
    f"Correlation between General Neutral and Positive: r={r_neu_pos_gen:.3f}, p={p_neu_pos_gen:.3e}")


```

✓ 0.0s

```
Jargon-Heavy Posts

Correlation between Jargon-Heavy Negative and Neutral: r=-0.934, p=4.288e-13
Correlation between Jargon-Heavy Negative and Positive: r=-0.041, p=8.349e-01
Correlation between Jargon-Heavy Neutral and Positive: r=-0.320, p=9.734e-02


General Posts

Correlation between General Negative and Neutral: r=-0.771, p=1.545e-06
Correlation between General Negative and Positive: r=-0.180, p=3.600e-01
Correlation between General Neutral and Positive: r=-0.487, p=8.536e-03
```

9.7 Paired T-Test and Wilcoxon signed-rank test

Jargon Heavy

```
# Paired t-test
t_neg_neu, p_neg_neu = ttest_rel(jargon_negative_norm, jargon_neutral_norm)
t_neg_pos, p_neg_pos = ttest_rel(jargon_negative_norm, jargon_positive_norm)
t_neu_pos, p_neu_pos = ttest_rel(jargon_neutral_norm, jargon_positive_norm)


# Wilcoxon signed-rank test
w_neg_neu, p_w_neg_neu = wilcoxon(jargon_negative_norm, jargon_neutral_norm)
w_neg_pos, p_w_neg_pos = wilcoxon(jargon_negative_norm, jargon_positive_norm)
w_neu_pos, p_w_neu_pos = wilcoxon(jargon_neutral_norm, jargon_positive_norm)


# Display results
print("Jargon-Heavy Posts")
print(f'{len("Jargon-Heavy Posts****")*"*"}')
print(
    f"* T-Test (Jargon-Heavy Negative vs. Neutral): t={t_neg_neu:.3f}, p={p_neg_neu:.3e}")
print(
    f"* Wilcoxon Test (Jargon-Heavy Negative vs. Neutral): W={w_neg_neu:.3f}, p={p_w_neg_neu:.3e}")
print('\n')
print(
    f"* Test (Jargon-Heavy Negative vs. Positive): t={t_neg_pos:.3f}, p={p_neg_pos:.3e}")
print(
    f"* Wilcoxon Test (Jargon-Heavy Negative vs. Positive): W={w_neg_pos:.3f}, p={p_w_neg_pos:.3e}")
print('\n')
print(
    f"* T-Test (Jargon-Heavy Neutral vs. Positive): t={t_neu_pos:.3f}, p={p_neu_pos:.3e}")
print(
    f"* Wilcoxon Test (Jargon-Heavy Neutral vs. Positive): W={w_neu_pos:.3f}, p={p_w_neu_pos:.3e}")
```

✓ 0.0s

```
Jargon-Heavy Posts
**********
* T-Test (Jargon-Heavy Negative vs. Neutral): t=-7.182, p=1.005e-07
* Wilcoxon Test (Jargon-Heavy Negative vs. Neutral): W=11.000, p=4.098e-07


* Test (Jargon-Heavy Negative vs. Positive): t=50.001, p=3.736e-28
* Wilcoxon Test (Jargon-Heavy Negative vs. Positive): W=0.000, p=7.451e-09


* T-Test (Jargon-Heavy Neutral vs. Positive): t=55.903, p=1.889e-29
* Wilcoxon Test (Jargon-Heavy Neutral vs. Positive): W=0.000, p=7.451e-09
```

General

```
# Paired t-test
t_neg_neu, p_neg_neu = ttest_rel(general_negative_norm, general_neutral_norm)
t_neg_pos, p_neg_pos = ttest_rel(general_negative_norm, general_positive_norm)
t_neu_pos, p_neu_pos = ttest_rel(general_neutral_norm, general_positive_norm)

# Wilcoxon signed-rank test
w_neg_neu, p_w_neg_neu = wilcoxon(general_negative_norm, general_neutral_norm)
w_neg_pos, p_w_neg_pos = wilcoxon(general_negative_norm, general_positive_norm)
w_neu_pos, p_w_neu_pos = wilcoxon(general_neutral_norm, general_positive_norm)

# Display results
print("General Posts")
print(f'{len("General Posts***")*"*"}')

print(
    f"* T-Test (General Negative vs. Neutral): t={t_neg_neu:.3f}, p={p_neg_neu:.3e}")
print(
    f"* Wilcoxon Test (General Negative vs. Neutral): W={w_neg_neu:.3f}, p={p_w_neg_neu:.3e}")
print('\n')

print(
    f"* T-Test (General Negative vs. Positive): t={t_neg_pos:.3f}, p={p_neg_pos:.3e}")
print(
    f"* Wilcoxon Test (General Negative vs. Positive): W={w_neg_pos:.3f}, p={p_w_neg_pos:.3e}")
print('\n')

print(
    f"* T-Test (General Neutral vs. Positive): t={t_neu_pos:.3f}, p={p_neu_pos:.3e}")
print(
    f"* Wilcoxon Test (General Neutral vs. Positive): W={w_neu_pos:.3f}, p={p_w_neu_pos:.3e}")
```

✓ 0.0s

```
General Posts
***************
* T-Test (General Negative vs. Neutral): t=3.460, p=1.809e-03
* Wilcoxon Test (General Negative vs. Neutral): W=74.000, p=2.440e-03


* T-Test (General Negative vs. Positive): t=58.739, p=5.017e-30
* Wilcoxon Test (General Negative vs. Positive): W=0.000, p=7.451e-09


* T-Test (General Neutral vs. Positive): t=44.483, p=8.468e-27
* Wilcoxon Test (General Neutral vs. Positive): W=0.000, p=7.451e-09
```