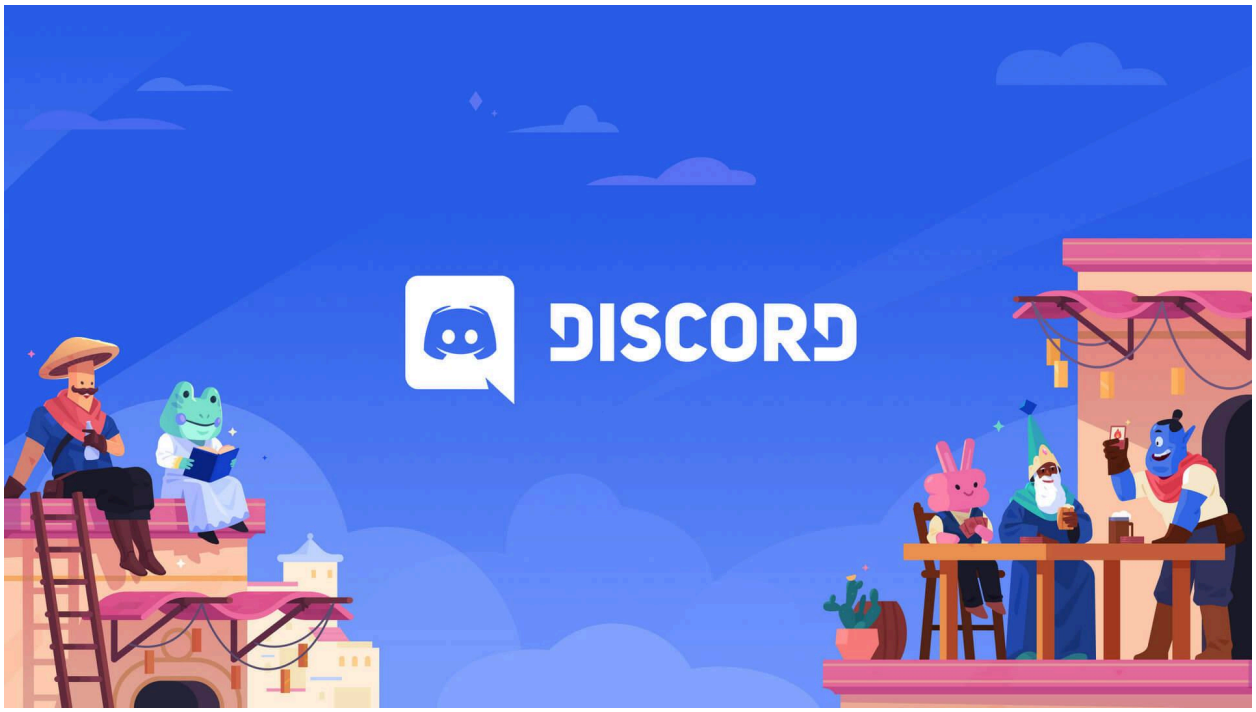
Analysis of Ocean Protocol Community on Discord



Cyril Mawutor Agbenyenu
March 2024

Table of Contents

Abstract

The challenge provides insights into community engagement and trends on Ocean Protocol's Discord server, offering a deeper understanding of dynamics and predicting future activity patterns. Through analyses of message counts, user activity, technical issues, and spam detection, stakeholders gain actionable insights into user behavior and server health. Identification of top contributors, analysis of message content, and detection of technical issues contribute to maintaining a positive user experience and fostering a vibrant community environment. The predictive modeling of future activity patterns enables proactive measures to adapt strategies and resources effectively. Overall, the challenge serves as a foundation for ongoing monitoring, analysis, and optimization of community engagement efforts, fostering collaboration, innovation, and growth within Ocean Protocol's ecosystem.

Introduction

Ocean Protocol is a decentralized data exchange protocol that facilitates permissionless data exchange between various parties. Ocean achieves this by using decentralized technology where a central entity does not control the underlying system. This makes Ocean a live tech stack on which autonomous applications including artificial intelligence (AI).

There are two specific sides to Ocean, these are; the core protocol, and its vibrant community. The core protocol is made up of *datatokens*, and *compute-to-data*. Datatokens are decentralized data assets that enable decentralized access control, via token-gating. The key principle here is that data services published as ERC721 non-fungible tokens (NFTs), can be consumed using the required amount of datatokens.

Compute-to-data (C2D) on the other hand is a privacy solution built for parties that wish to buy and sell private data. With C2D, access is granted to run compute against the data but the data itself does not leave the premises, only the results are visible to the consumer. C2D is suitable for situations where there are concerns over privacy like when companies want to sell their data sets.

The community of Ocean protocol is as live as it gets. With an army of enthusiastic builders, the number of decentralized applications (dApps) within the Ocean ecosystem has consistently grown over the years. The Ocean community also houses data scientists and OCEAN token holders who are excited about the future of AI and data. These members participate in predictions, data challenges (like this one), and more.

The vibrant Ocean community can interact directly on the Discord server. In the Discord server, everyone is welcome to throw ideas back and forth, developers can share their coding issues for real-time help from other builders, and much more. There are also Ambassadors, who are enthusiasts, who take it upon themselves, to spread the good news and the offerings of Ocean protocol.

This report delves into the inner workings of the Ocean Protocol's Discord community. By analyzing conversations and user activity, we aim to understand how people engage, what topics spark the most interest, and how the community evolves. This knowledge will help predict future activity and foster a thriving space for Ocean protocol enthusiasts.

Moreover, by understanding how engagement fluctuates (thin surges in messages or new users), we can anticipate future trends and be prepared for future spikes in activity. This allows for proactive measures, like adding moderators or addressing recurring technical issues. We can also identify common questions and user concerns, allowing us to tailor specific resources and support.

Data Cleaning

The cleaning process for the Discord dataset involved several steps to ensure that the data was structured, consistent, and ready for analysis. The *Channel* column contained information about the specific channel within the Discord server. We used a regular expression pattern '**\[(\d+)\]**' to extract the integers between the '[' and ']' and characters in the Channel column.

The *Author ID* column was left unchanged as it serves as a unique identifier for each user within the Discord server. No cleaning or modification was required for this column. Similarly, the *Author* column which contains the usernames of the individuals who posted messages, was left untouched during the cleaning process. Preserving the original author names maintains the integrity of the data and allows for accurate identification of contributors.

The Date column required conversion to a *datetime* format to facilitate temporal analysis. Using Python's *datetime* module, we parsed the date and time information provided in the original dataset. This ensured that timestamps were correctly interpreted and could be utilized for chronological analysis and trend identification.

The *Content* column contained the text of the message posted on the Discord server. To clean this column, we removed any URLs present in the message using regular expressions. This step eliminated extraneous information and focused solely on the textual content relevant to the analysis. The Attachments column indicates whether messages included attachments such as images or documents.

We transformed this column to a binary indicator, and labeled it as 'Yes' if attachments were present, and 'No', if not. This simplification allows for easy identification of messages with attachments for easy further analysis. A new column, *Attachment Type*, was created to categorize attachments based on their file extensions. By extracting the extension from the attachment URLs using regular expressions, we identified attachment types such as images, documents, or other file formats.

This additional information enhances the granularity of the dataset and enables specific analysis of attachment types. Lastly, the *Reactions* column contained various emojis and their corresponding counts, reflecting user engagement with the messages. To standardize the representation of reactions, we aggregated the counts of all reactions within each message. By summing the counts, we obtained a single integer value representing the overall reaction activity for each message.

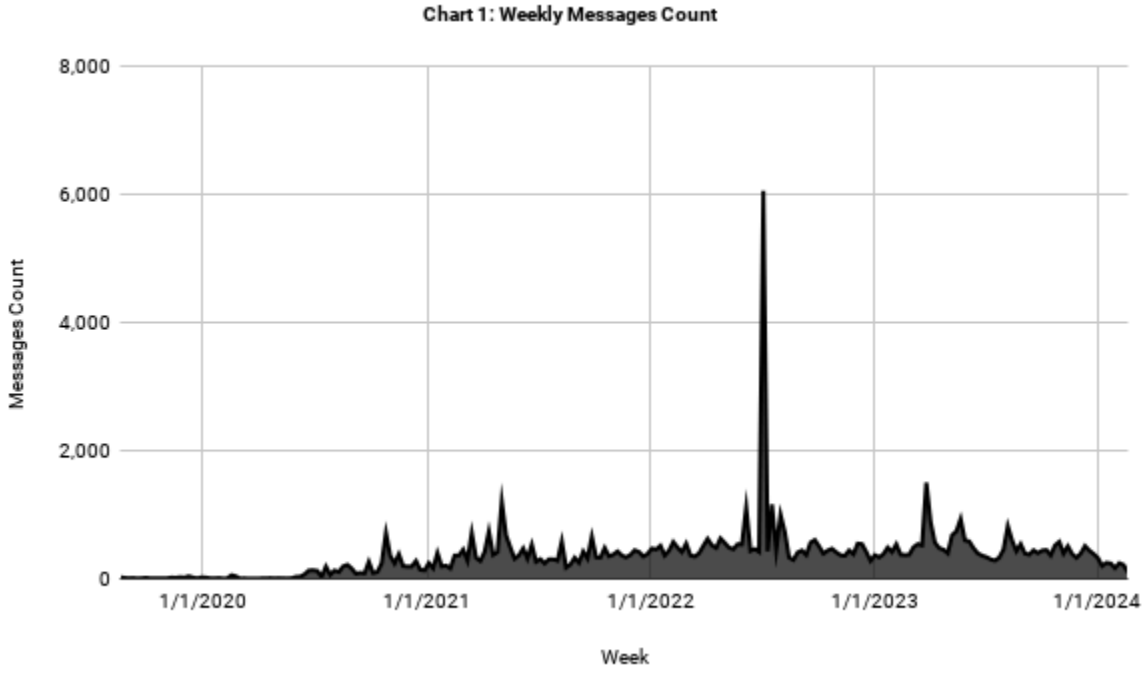
By systematically cleaning each column of the Discord dataset, we ensured that the data was prepared for comprehensive analysis of community engagement and trends within the Ocean Protocol's Discord server. These cleaning steps contribute to the reliability and accuracy of the insights derived from the dataset, enabling meaningful interpretations and informed decision-making.

Key Findings

With a total of **21 channels**, the Discord server provides a well-organized space for diverse conversations. It also boasts an impressive **9,025 authors**, indicating a highly engaged user base. So far, there have been over **1.9 million words** exchanged on the server over 1,465 days equivalent to four and a half **(4.5) years**.

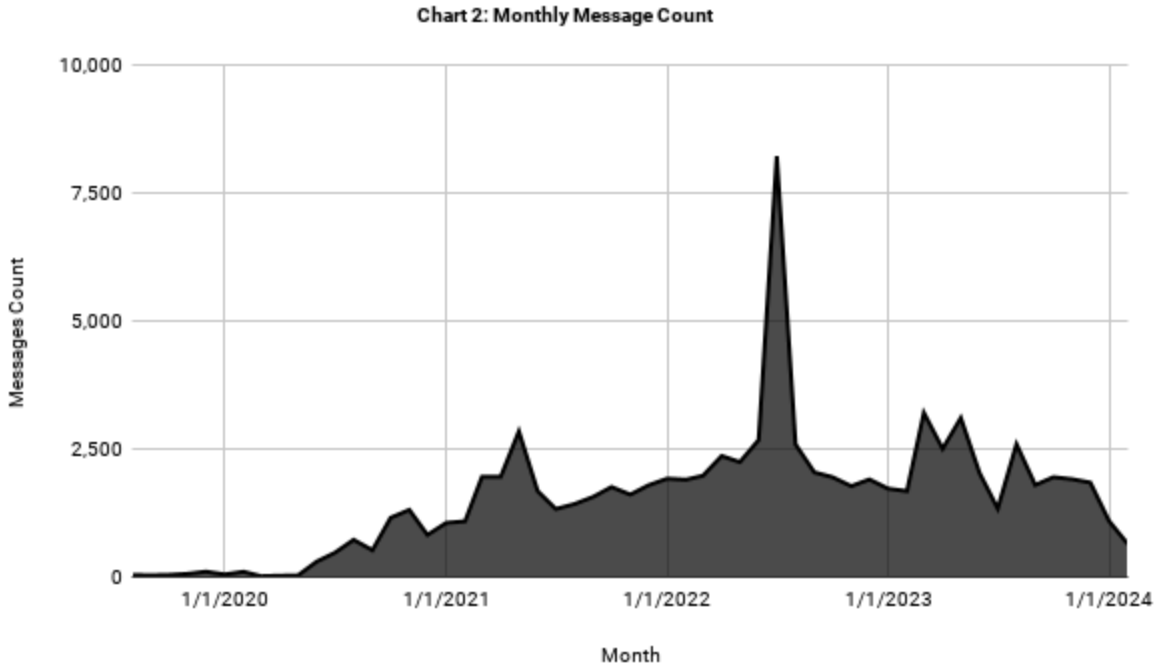
General Trends

The server started with a trickle of messages, averaging around 10 messages per week. This initial phase likely reflects a small, core group of enthusiasts gathering and laying the groundwork for the community. Around June 2022, we see a dramatic surge in activity, with message count reaching triple digits at times. This could be due to several factors, such as increased awareness of Ocean Protocol, news announcements, or exciting project developments.



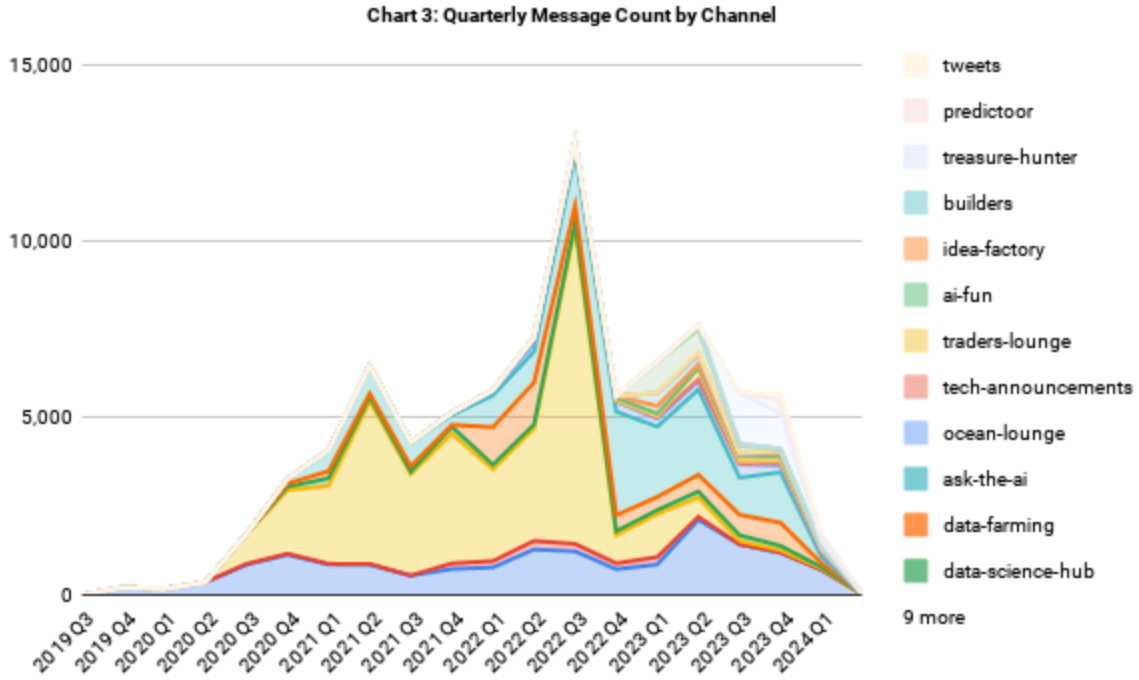
Following the initial burst, message frequency settled into a steadier range, averaging around 300-400 messages. This suggests a more established community with a consistent level of engagement. There are still occasional spikes like in March 2023 and July 2023, hinting at potential events hinting at renewed interest.

Monthly message counts steadily **climbed from 38** in August 2019 **to 299 in June 2020**. This suggests a slow but consistent growth in the community, confirming the analysis of the weekly message count.



The most recent months (January and February 2024) show a decrease in activity compared to the previous year, with messages dropping below 1,200 per month. This could be due to a variety of factors such as community fatigue or lack of major news or update. It could not have been market conditions as the crypto market is in an uptrend.

The data reveals a clear distinction between channels. *General-chat* stands out as the central hub for communication, exhibiting consistent activity with a gradual upward trend until recently. This suggests a core community thrives on general discussions. It can be observed from the chat below that not all channels follow the same trend, they exhibit many varying levels of activity and engagement.



The significant rise in “welcome” activity in 2021 is an outlier period. This likely reflects a specific marketing campaign or event that successfully attracted new users. Identifying and replicating such successful strategies could be instrumental in future user acquisition efforts.

Correlations

The correlation between the price of $OCEAN and the number of messages, new users, and active individuals on the server was analyzed using a systematic approach. Initially, the Discord dataset, containing message content, author details, and timestamps, was prepared for analysis. This involved organizing the data and extracting relevant metrics for correlation analysis.

For the correlation with the number of messages, no preprocessing was necessary as the focus was on directly counting the daily message volumes. However, for new users and active individuals, specific groups were defined to capture pertinent engagement patterns. New users were identified as those who posted their inaugural message on a given day, while active individuals were those who contributed at least one message on the same day.

With the data suitably organized, the correlation coefficients between the price of $OCEAN and each engagement metric were calculated. This involved merging the relevant dataset with the daily average price of $OCEAN and computing the correlation coefficient using statistical methods. The correlation coefficient serves as a

statistical indicator of the strength and direction of the relationship between variables.

The correlation coefficients between the price of $OCEAN and the number of messages, new users, and active individuals on the server were calculated using Pearson's correlation coefficient, which is a widely used statistical method for measuring the linear relationship between two variables. Pearson's correlation coefficient, denoted by the symbol "r", quantifies the strength and direction of the linear association between two continuous variables. It ranges from -1 to 1, where:

- A correlation coefficient close to 1 indicates a strong positive linear relationship.
- A correlation coefficient close to -1 indicates a strong negative linear relationship.
- A correlation coefficient close to 0 indicates little to no linear relationship between the variables.

By applying Pearson's correlation coefficient, we were able to statistically assess the extent of the relationship between the price of $OCEAN and the engagement metrics observed on the Discord server. This method enabled us to derive valuable insights into the dynamics and trends within the community, facilitating informed decision-making and strategic planning.

Table 1: Correlation Between $OCEAN Price and Discord Activity

Metric	**Correlation Coefficient**	**Insight**
Messages	-0.1752	Weak negative correlation. Message volume might decrease slightly with increasing $OCEAN price, but the effect is minimal.
New Users	0.0299	Essentially no correlation. Price changes don't significantly impact new user acquisition.
Active Users	-0.0132	Essentially no correlation. Price changes don't significantly impact user engagement.

The correlation coefficients between the price of $OCEAN and server activity metrics reveal a weak negative correlation with message count (-0.175) and statistically insignificant correlations with new and active users (0.029 & -0.013). There might be a very slight decrease in message volume as the price of $OCEAN increases. However, the weakness of the correlation (-0.175) indicates this connection is minimal and could be due to random chance.

There's essentially no correlation between price and the number of new or active users joining the server. This suggests price fluctuations don't significantly impact user acquisition or engagement. While correlations provide a starting point, further

analysis is needed to understand the relationship between price and server activity. Here are some methods to consider:

1. Time Series Analysis: Analyze price and server activity data chronologically to identify potential trends or patterns over time.
2. Regression Analysis: Build a regression model to quantify the strength and direction of the relationship between price changes and server activity metrics.
3. Granger Causality Test: Explore if price changes cause fluctuations in server activity or vice versa.
4. Event Study: If there are specific events impacting price (e.g., news announcements), analyze server activity around those events to see if there's a corresponding change in user behavior.

By employing these methods, we can gain a deeper understanding of the relationship between price and the Ocean Protocol Discord server's activity.

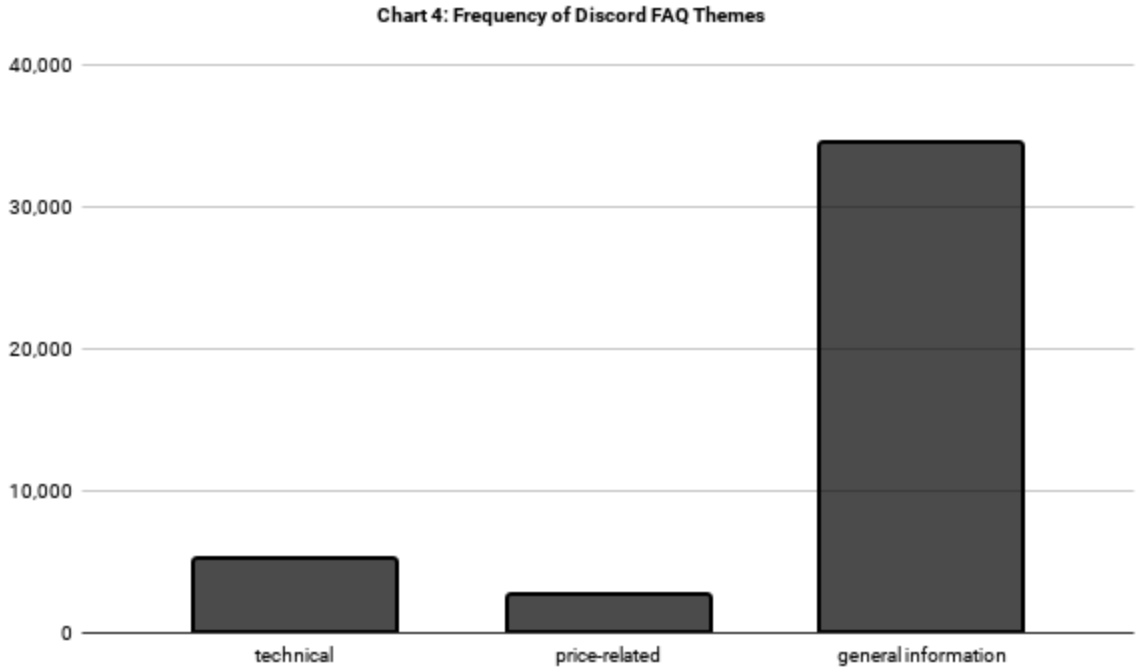
Community Questions

Identifying and categorizing frequently asked questions (FAQs) within a Discord server involves leveraging Python modules such as pandas, nltk, and collections to extract insights from user interactions. Initially, the text data from the Discord dataset is preprocessed using techniques such as tokenization and stopword removal. This preprocessing step ensures that the text is standardized and suitable for analysis.

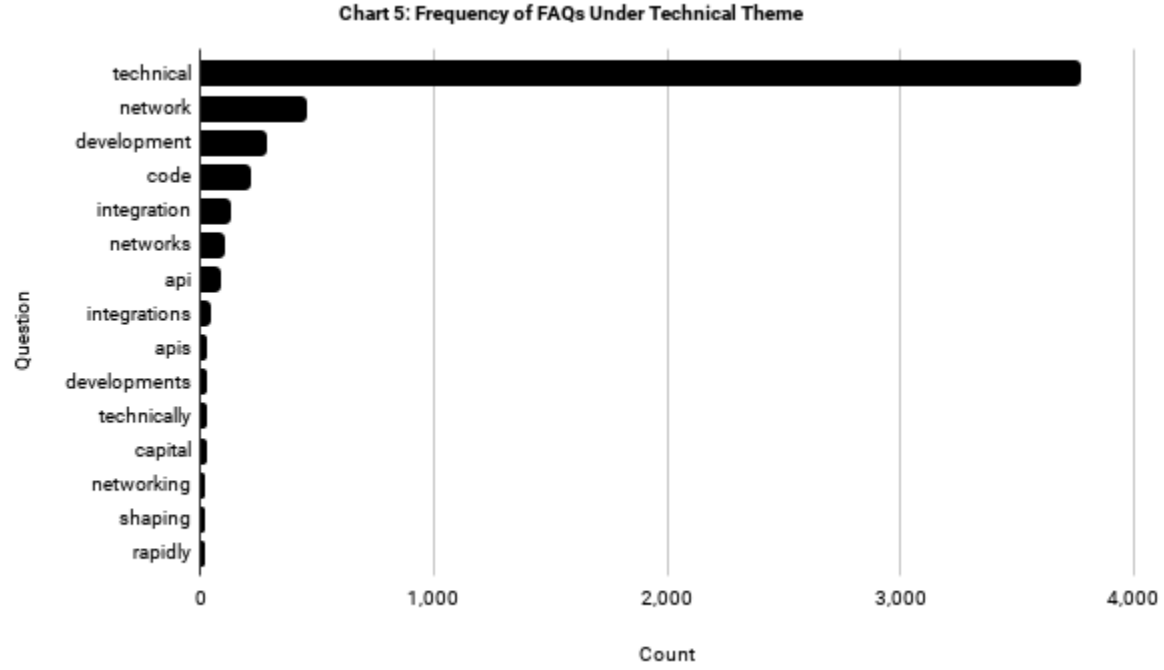
Utilizing the nltk module, the frequency of each word or token is computed using the Counter class. This frequency analysis allows us to discern which terms occur most frequently in the Discord conversations, providing valuable insights into prevalent topics. Subsequently, predefined categories or themes, such as 'technical', 'price-related', and 'general information', are established based on the observed discussion patterns within the community.

During the categorization process, words or tokens are examined to determine their alignment with predefined themes. This determination is based on the presence of specific keywords or phrases associated with each theme. The nltk module's stopwords corpus assists in filtering out common words that do not carry significant meaning in the context of the analysis.

Once categorized, questions within each theme are sorted based on their frequency count using Python's collections module. This sorting enables the identification of the most frequently asked questions within each thematic category. Finally, the categorized questions, along with their associated themes and frequency counts, are compiled into a structured format, such as a CSV file, using the pandas module.



Upon analysis, it's evident that the theme generating the most questions is 'general information', with a significant count of 18039 questions. These questions encompass a broad range of topics, including general inquiries about the project, documentation, and community-related queries.



Within the 'technical' theme, various subcategories are discernible, such as network-related queries, development topics, and API integrations. Although the 'technical' theme exhibits a high volume of questions, the distribution is more fragmented across specific technical aspects and terminology.

Chart 6: Frequency of FAQs Under Price Theme

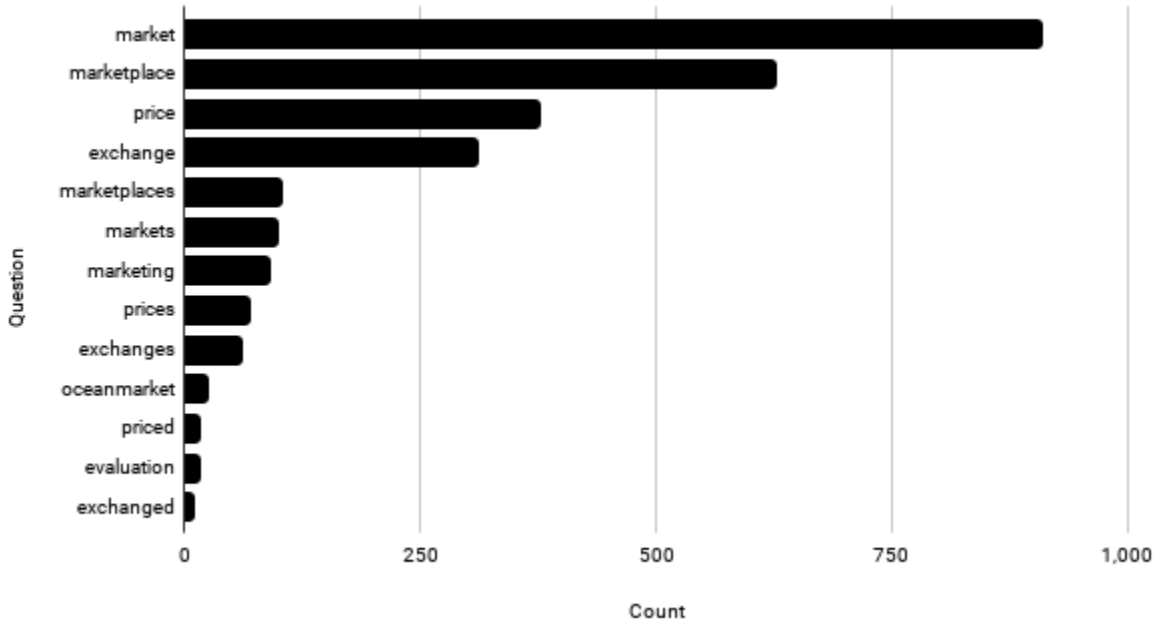
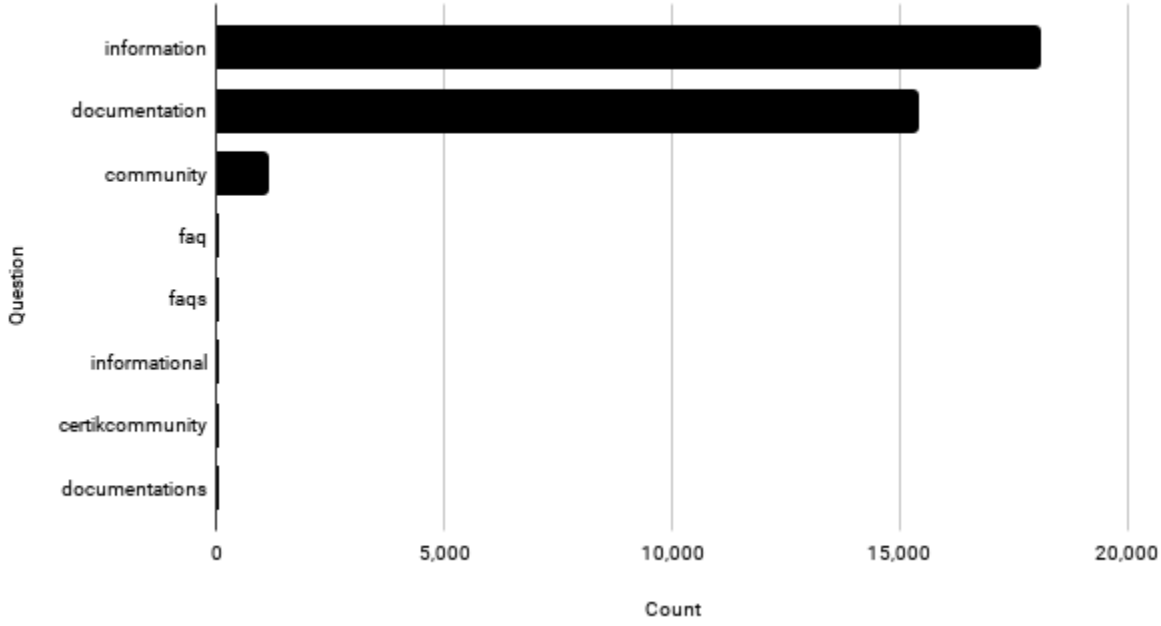


Chart 7: Frequency of FAQs Under General Info Theme

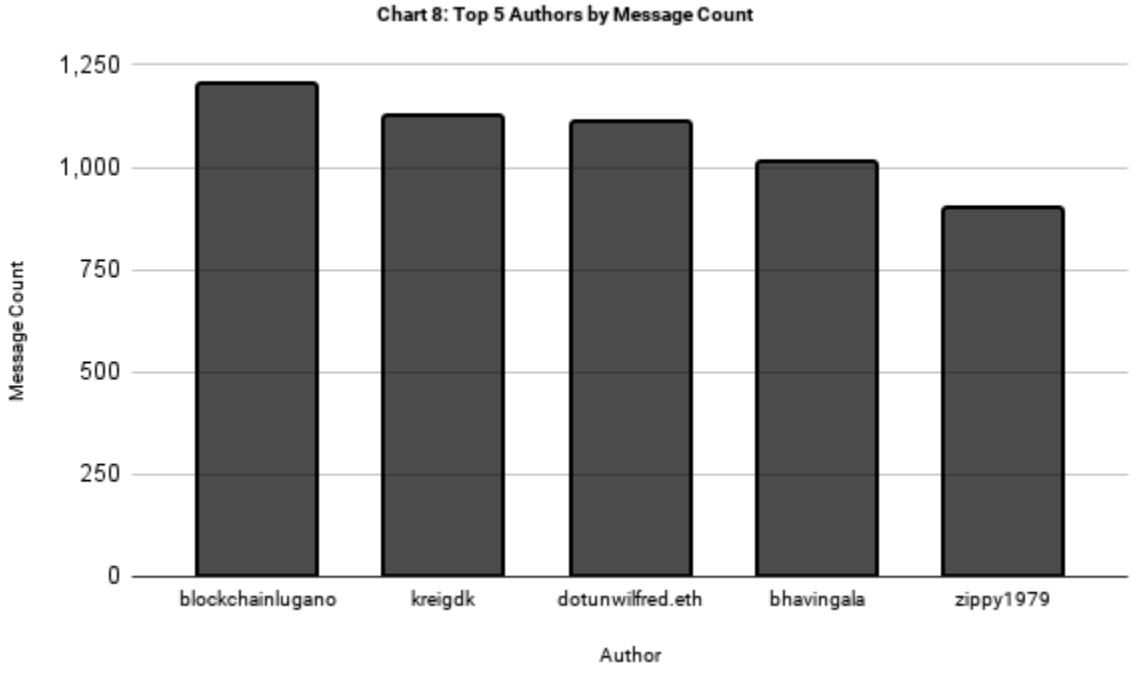


In comparison, the 'price-related' theme demonstrates a focus on market-related discussions, including market evaluation, pricing, and exchanges. While the volume of questions in the 'price-related' category is substantial, it is surpassed by the sheer breadth of inquiries categorized under 'general information'.

Users within the Ocean Protocol Discord community display a keen interest in obtaining general information about the project, its documentation, and community-related matters. This trend underscores the importance of providing comprehensive and accessible information to users, fostering community engagement, and addressing common queries effectively.

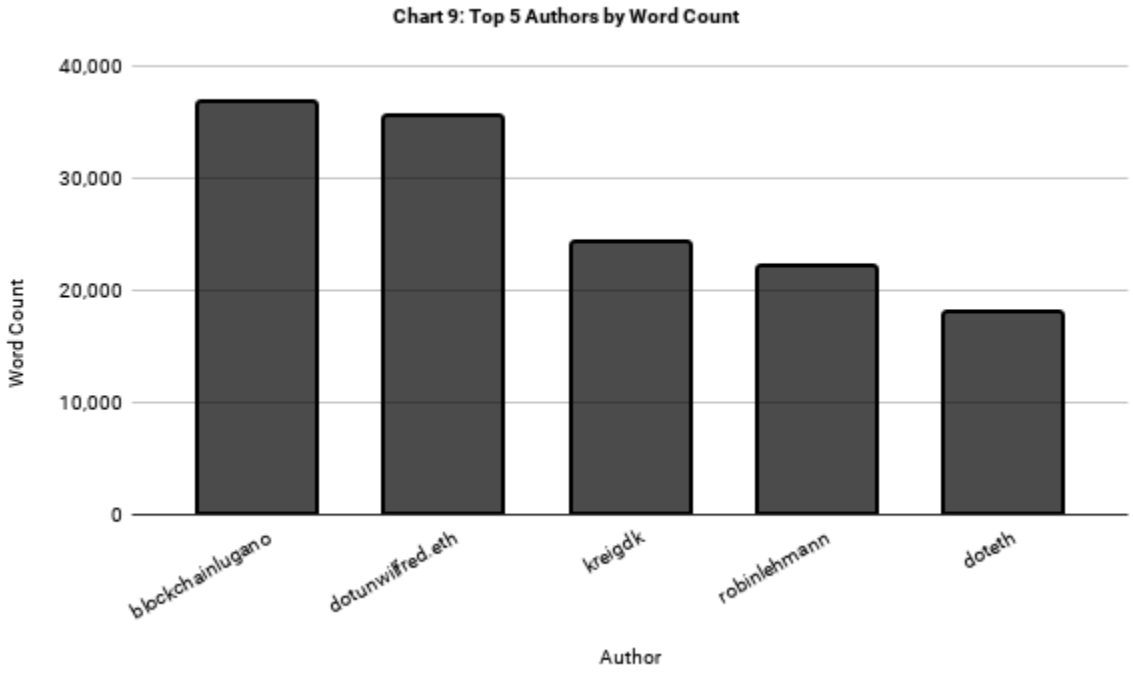
Community Activity

At the forefront of Ocean Protocol's discord users with the most messages is 'blockchainlugano' with 1207 messages, indicating significant participation and engagement within the community. Following closely are 'kreigdk' and 'dotunwilfred.eth' with 1126 and 1113 messages, respectively, suggesting active involvement and consistent contribution to discussions.



'Bhavingala', with 1016 messages, also emerges as a prominent contributor, reflecting a strong presence and participation in community interactions. 'Zippy1979' rounds off the top 5 authors with 904 messages, demonstrating substantial engagement and involvement in Discord conversations. Collectively, these top authors represent a core segment of the Ocean Protocol Discord community, playing pivotal roles in fostering discussions, sharing insights, and contributing to the vibrant exchange of ideas within the platform.

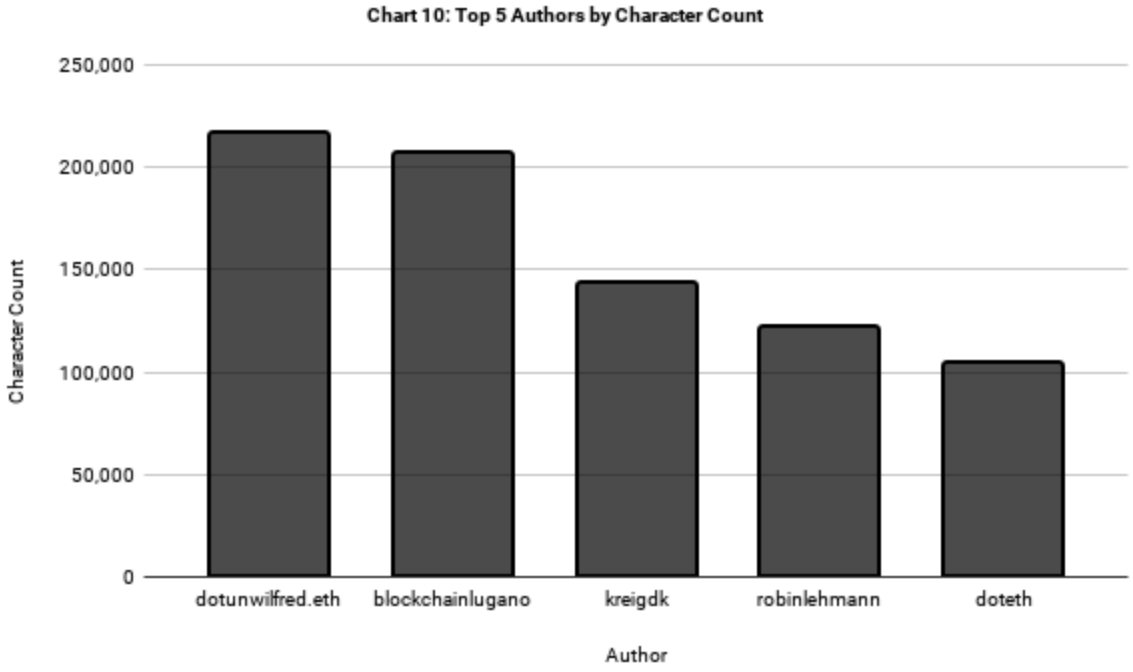
The top 5 authors in Ocean Protocol's Discord server by word count exhibit significant engagement and contribution within the community. Topping the list is blockchainlugano, whose word count of 36,920 indicates substantial participation in discussions and conversations. Dotunwilfred.eth closely follows with 35,562 words, reflecting active involvement and meaningful contributions to various topics.



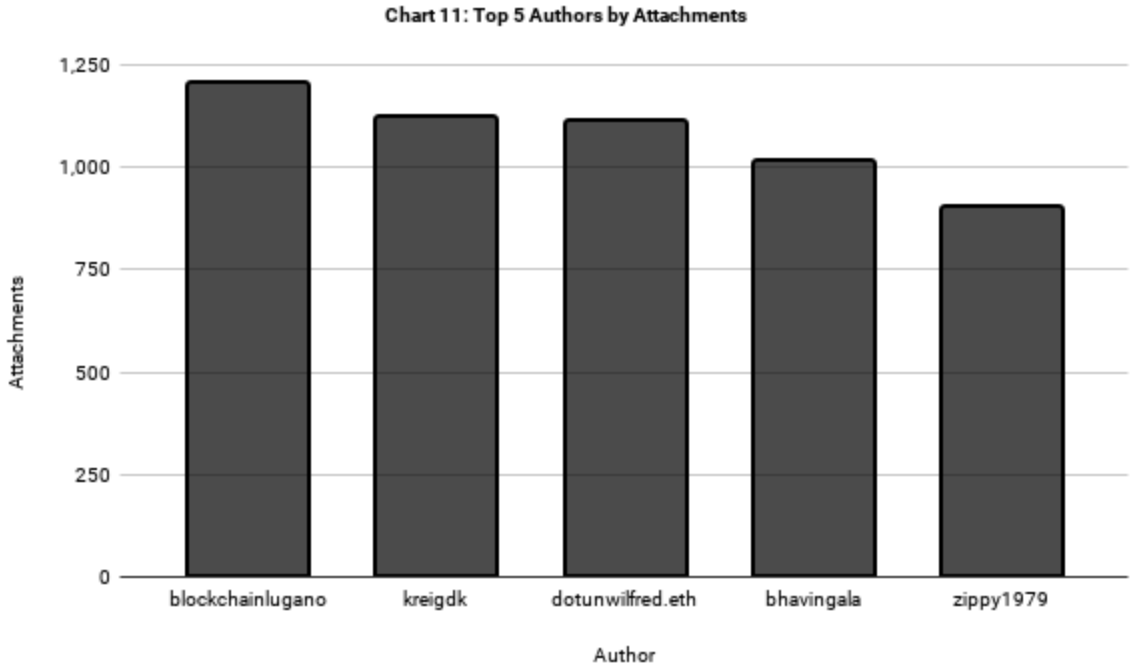
Kreigdk ranks third with 24,329 words, demonstrating consistent engagement and communication within the community channels. Robinlehmann's word count of 22,280 underscores a notable presence and valuable input in discussions and interactions. Doteth rounds up the top 5 authors with 18,017 words, showcasing consistent engagement and active participation in community conversations.

In Ocean Protocol's Discord community, the top 5 authors by character count showcase a diverse range of active participants. Dotunwilfred.eth emerges as the most prolific contributor, demonstrating significant engagement through a remarkable character count of 217,440. Blockchainlugano closely follows, exhibiting substantial involvement with 207,819 characters contributed.

Kreigdk, with 143,748 characters, and Robinlehmann, with 122,662 characters, further exemplify consistent participation and dedication to discussions within the community. Doteth rounds up the top 5, showcasing significant involvement with 104,565 characters contributed.



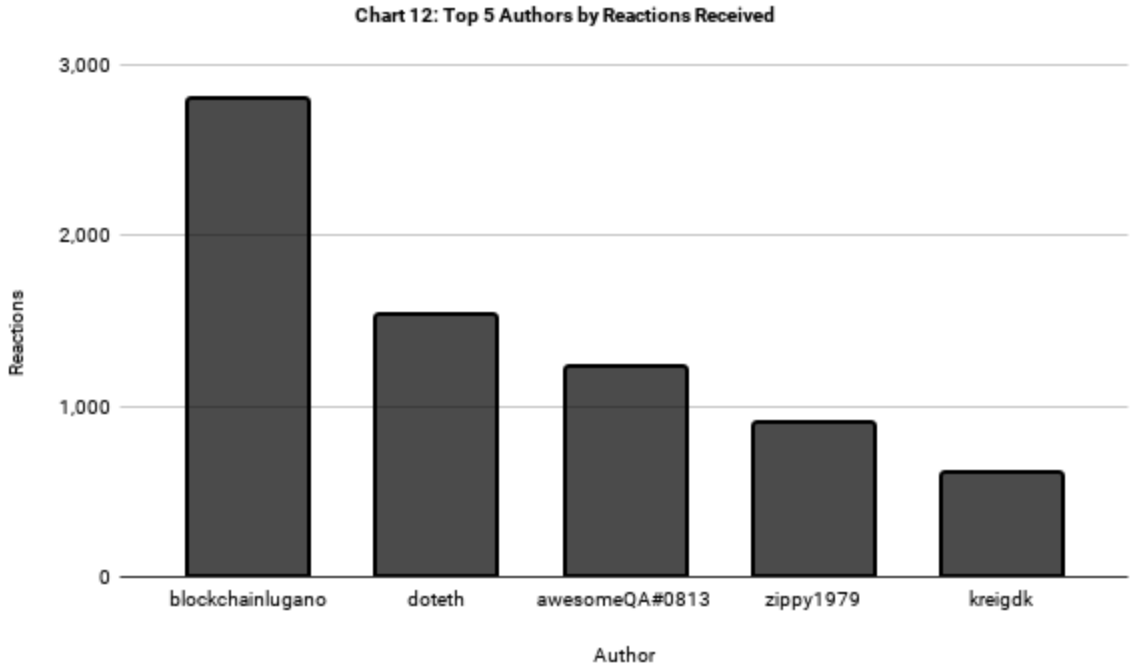
The chart below presents the top 5 authors in Ocean Protocol's Discord based on their attachment count. At the forefront is 'blockchainlugano' with 1207 attachments, followed closely by 'kreigdk' with 1126 attachments. 'dotunwilfred.eth' occupies the third position with 1113 attachments, indicating active engagement within the community. 'bhavingala' and 'zippy1979' complete the list with 1016 and 904 attachments, respectively. These authors appear to be prolific contributors to discussions, often supplementing their messages with various attachments.



The high number of attachments suggests a rich exchange of information, potentially comprising images, documents, or links to external resources. It's indicative of a vibrant community where members share and discuss diverse content related to Ocean Protocol. The consistency among the top contributors underscores the collaborative spirit and active participation within the Discord server, fostering a conducive environment for knowledge sharing and engagement.

The top 5 authors in Ocean Protocol's Discord by reactions count provide valuable insights into the community's dynamics and engagement patterns. At the forefront, blockchainlugano's substantial 2810 reactions indicate a high level of involvement and influence within the community. Their contributions likely include thought-provoking insights, informative content, or active participation in discussions.

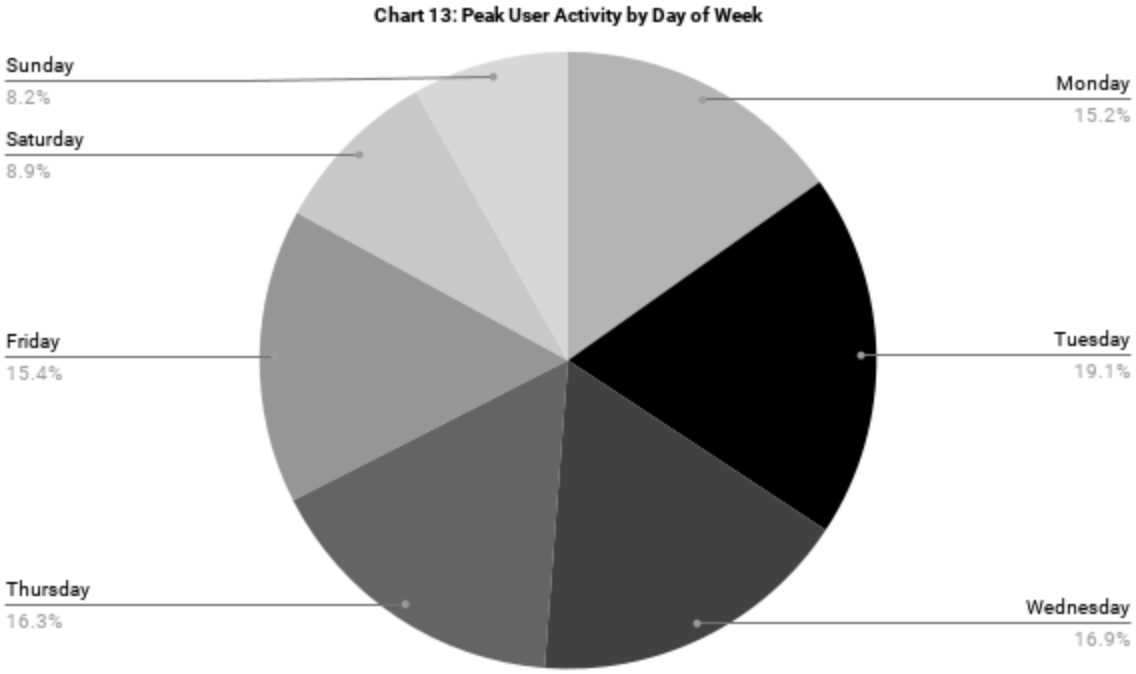
Following closely, doteth's 1545 reactions underscore their significant impact and engagement, suggesting a consistent presence and valuable contributions to the community discourse. In the third position, awesomeQA#0813's 1240 reactions reflect active participation and contributions that resonate with community members, fostering meaningful interactions and discussions. Zippy1979, holding the fourth spot with 910 reactions, contributes significantly to the community's engagement, likely through insightful comments, supportive interactions, or valuable resources shared.



Finally, kreigdk's 619 reactions demonstrate consistent involvement and positive interaction within the community, albeit slightly lower than the top contributors. Together, these top authors play crucial roles in shaping the discourse, fostering collaboration, and maintaining a vibrant community atmosphere within Ocean Protocol's Discord. Their active engagement, thoughtful contributions, and supportive interactions contribute to the community's growth, knowledge sharing, and overall cohesion.

Analyzing the peak user activity per day of the week, we observe significant variations in message counts across different days. Tuesday emerges as the busiest day with a message count of 16,173, indicating heightened engagement within the Ocean Protocol Discord community. Following closely, Wednesday records 14,357 messages, maintaining a consistent level of activity throughout the week.

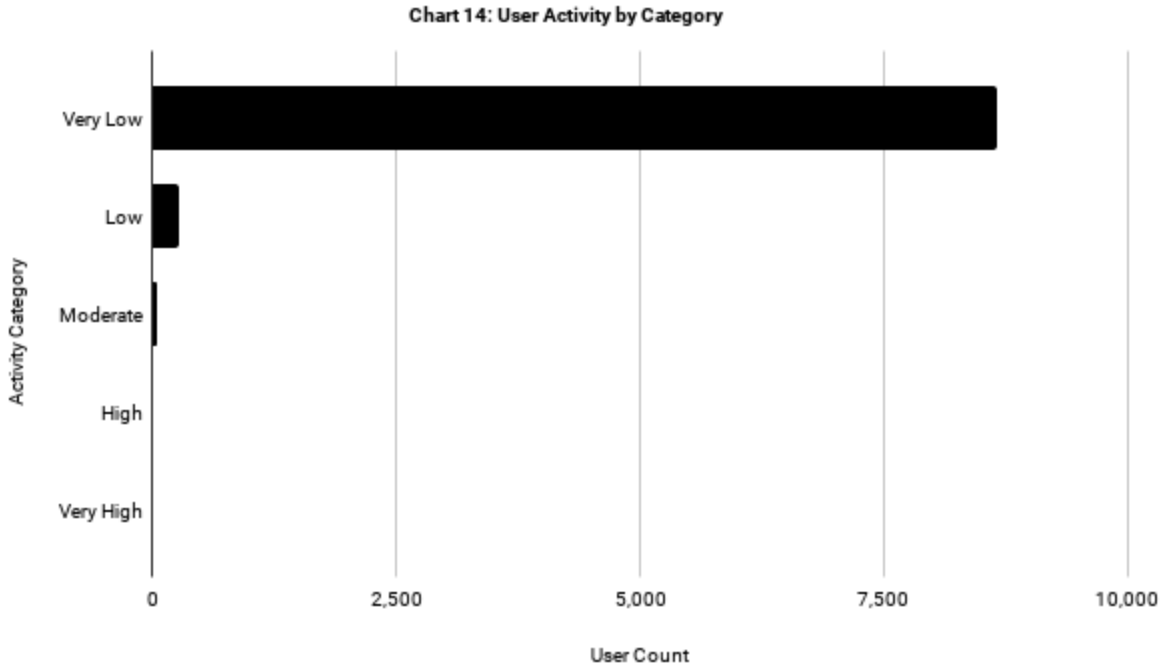
Thursday and Friday display slightly lower activity levels compared to the peak days, yet they remain notably active, with 13,804 and 13,083 messages, respectively. The weekends witness a decline in activity, with Saturday and Sunday registering 7,517 and 6,948 messages, respectively, reflecting a more relaxed engagement pattern among community members during leisure days.



The analysis of user activity by category reveals a notable distribution across various levels of engagement within the Ocean Protocol's Discord community. The majority of users, approximately 8666 individuals, fall within the "Very Low" activity category, indicating minimal participation or engagement within the platform. This suggests that a substantial portion of users may be passive observers rather than active contributors to discussions and interactions.

In contrast, a relatively smaller number of users, specifically 285 individuals, fall within the "Low" activity category, indicating slightly more engagement than the "Very Low" category but still relatively minimal participation overall. This suggests that while there is a larger base of minimally engaged users, a smaller subset of individuals is more actively involved in the community.

Furthermore, the distribution becomes significantly narrower as the activity levels increase, with only 46 users categorized as "Moderate," 10 users categorized as "High," and a mere 4 users classified under "Very High" activity. This indicates a pyramid-shaped distribution of user engagement, with a vast base of less active users and a progressively smaller number of highly engaged participants.



Overall, this distribution underscores the importance of understanding user behavior and engagement patterns within online communities, as it can inform community management strategies and provide insights into the dynamics of user interaction and participation. It also suggests opportunities for community-building initiatives aimed at increasing engagement levels and fostering a more vibrant and interactive online environment.

The code segments users into activity categories based on the count of messages per user. It uses the following message count thresholds for categorization: 'Very Low' for 0-10 messages, 'Low' for 11-100 messages, 'Moderate' for 101-500 messages, 'High' for 501-1000 messages, and 'Very High' for over 1000 messages.

Scam & Spam

The model employed in this scenario is a Random Forest Classifier, a type of ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes as the prediction. Random forests are advantageous due to their ability to handle high-dimensional data, feature importance ranking, and resistance to overfitting.

To train the model, the dataset was preprocessed by taking messages from '**Deleted User**' as sample spam and splitting it into training and testing sets. The text data was converted into numerical vectors using the TF-IDF Vectorizer, which represents the importance of a word in a document relative to a collection of documents. The Random

Forest Classifier was then trained on the training set, using a collection of decision trees.

During prediction, the trained model utilized the learned decision rules from the ensemble of trees to classify messages as either spam or non-spam. The classification report provides insights into the model's performance, including precision, recall, and F1-score for both spam and non-spam classes, as well as overall accuracy.

Table 2: Random Forest Classifier Model Evaluation Report For Spam Identification

	Precision	Recall	F1-Score	Support
non-spam	0.916	0.852	0.883	1162
spam	0.862	0.922	0.891	1167
accuracy	0.887	0.887	0.887	0.887
macro avg.	0.889	0.887	0.887	2329
weighted avg.	0.889	0.887	0.887	2329

Looking at the classification report, the model achieved high precision and recall for both spam and non-spam classes, indicating its effectiveness in correctly identifying instances of each class. The F1-score, which is the harmonic mean of precision and recall, provides a balanced assessment of the model's performance. The accuracy metric shows the proportion of correctly classified instances overall.

The precision for spam messages suggests the percentage of correctly identified spam among all messages classified as spam, while recall indicates the proportion of actual spam messages correctly identified by the model. Similarly, precision and recall for non-spam messages provide insights into the model's ability to distinguish non-spam messages accurately.

The model's choice of Random Forest Classifier was justified by its ability to handle text data effectively, its robustness against overfitting, and its suitability for binary classification tasks like spam detection. Characteristics of spam/scam messages typically include unsolicited content, suspicious links, excessive use of capital letters, and content related to fraudulent activities or promotions.

classification_report.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

# Load the cleaned dataset
df = pd.read_csv('cleaned_discord.csv')

# Drop rows with missing values in the 'Content' column
df = df.dropna(subset=['Content'])

# Filter out messages from 'Deleted User' as potential spam/scam
deleted_user_messages = df[df['Author'] == 'Deleted User']

# Filter out non-spam messages from other authors
non_deleted_user_messages = df[df['Author'] != 'Deleted
User'].sample(len(deleted_user_messages))

# Combine spam and non-spam messages
spam_nonspam_data = pd.concat([deleted_user_messages, non_deleted_user_messages])

# Split data into features and labels
X = spam_nonspam_data['Content']
y = np.where(spam_nonspam_data['Author'] == 'Deleted User', 'spam', 'non-spam')

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert text data into numerical vectors using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_vec, y_train)

# Predictions on the test set
y_pred = rf_classifier.predict(X_test_vec)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred, output_dict=True)

# Convert classification report to dataframe and save to CSV
classification_df = pd.DataFrame(classification_rep).transpose()
classification_df.to_csv('classification_report.csv')

print("Classification report saved to 'classification_report.csv'.")
```

Image 1: Code of The Random Forest Classifier

Technical Issues

The categorization of technical issues serves several important purposes within the context of managing the Ocean Protocol's Discord community. It helps to systematically identify and classify the various types of problems or challenges that users encounter while interacting with the platform. By categorizing these issues, community moderators and technical support teams can better understand the nature and scope of the challenges faced by users.

The method used for categorization in the provided code leverages Python and the pandas library to analyze messages extracted from the Ocean Protocol's Discord platform. Initially, the script loads a cleaned dataset using pandas, which contains messages from the Discord server. It then iterates through each message to identify potential technical issues based on specific keywords or patterns, such as 'error,' 'bug,' or 'crash.' These keywords serve as indicators of technical problems that users may have encountered.

Once the technical issues are identified, the script categorizes them into three main categories: User-Related, System-Related, and External-Factors. To categorize the issues, it examines the context of each message, looking for keywords that indicate whether the issue is related to users, the system infrastructure, or external factors beyond the platform's control. This process involves checking for keywords such as 'user,' 'account,' 'server,' 'database,' 'network,' and others commonly associated with each category.

For each category, the script maintains a count of the identified issues. Finally, it outputs the categorized technical issues along with their respective counts to a CSV file named 'categorized_technical_issues.csv.' This output provides a structured overview of the types and frequencies of technical issues encountered within the Ocean Protocol's Discord community, enabling further analysis and action to address recurring problems and improve user experience.

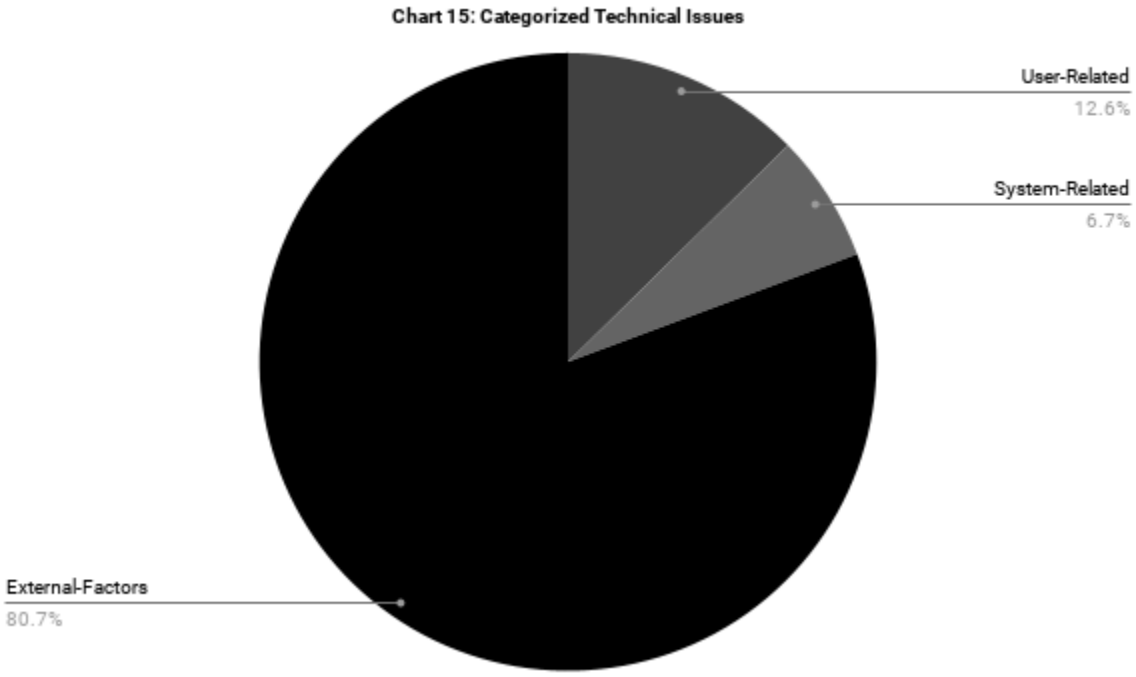
categorized_technical_issues.py

```
import pandas as pd

# Load the cleaned dataset
df = pd.read_csv('cleaned_discord.csv')

# Identify Technical Issues
technical_issues = []
for message in df['Content']:
    # Check if the message is a valid string
    if isinstance(message, str):
        # Example: Identify technical issues based on specific keywords or patterns
        if 'error' in message.lower() or 'bug' in message.lower() or 'crash' in
message.lower():
            technical_issues.append(message)

# Categorize Based on Context
categorized_issues = {'User-Related': 0, 'System-Related': 0, 'External-Factors': 0}
for issue in technical_issues:
    # Check if the issue is a valid string
    if isinstance(issue, str):
        # Example: Categorize based on keywords in the message
        if 'user' in issue.lower() or 'account' in issue.lower() or 'access' in issue.lower():
            categorized_issues['User-Related'] += 1
        elif 'server' in issue.lower() or 'database' in issue.lower() or 'network' in
issue.lower():
            categorized_issues['System-Related'] += 1
        else:
            categorized_issues['External-Factors'] += 1

# Output the Results to CSV
output_data = [{'Category': category, 'Issue_Count': count} for category, count in
categorized_issues.items()]
output_df = pd.DataFrame(output_data)
output_df.to_csv('categorized_technical_issues.csv', index=False)
print("Categorized technical issues saved to 'categorized_technical_issues.csv'.")
```

User-related issues, numbering 34, suggest that a portion of the technical difficulties stems from user behavior or misunderstanding of platform features. This category might include inquiries about functionality or navigation difficulties experienced by users. System-related issues, comprising 18 instances, indicate internal platform challenges such as bugs, errors, or performance issues that originate from within the system itself. Addressing these issues is crucial to ensure the stability and reliability of the platform.

On the other hand, external factors contribute significantly to technical challenges, with 217 reported instances. These issues may arise from external disruptions, such as server outages, network congestion, or maintenance activities affecting platform accessibility and performance. External factors often require collaboration with

external service providers or infrastructure adjustments to mitigate their impact effectively. Understanding the prevalence of external factors underscores the importance of monitoring and responding to external events that may affect platform operations.



In summary, the analysis highlights the diverse technical issues encountered within the Ocean Protocol's Discord community. Community managers and technical support teams can better prioritize their efforts, allocate resources effectively, and implement targeted solutions to improve platform functionality, reliability, and user experience over time by categorizing these issues into user-related, system-related, and external factors.

Prediction Model

Developing a forecasting model for Ocean Protocol Discord user activity is crucial for several reasons. It helps the community managers and moderators to anticipate and prepare for fluctuations in user engagement and activity levels. By understanding future trends, they can allocate resources effectively, plan events, and ensure adequate support for users during peak times.

The model implemented above is based on SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables), a popular time series forecasting method. SARIMAX is suitable for modeling time series data with seasonal patterns and can incorporate external factors that might influence the series.

model.py

```python
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error

# Load the monthly message count data
data = pd.read_csv('monthly_message_count.csv')

# Convert 'month_start_date' column to datetime
data['month_start_date'] = pd.to_datetime(data['month_start_date'])

# Set 'month_start_date' as the index
data.set_index('month_start_date', inplace=True)

# Train-test split
train = data.iloc[:-1]
test = data.iloc[-1:]

# Fit SARIMAX model
order = (1, 1, 1)
seasonal_order = (1, 1, 1, 12)
model = SARIMAX(train['messages_count'], order=order, seasonal_order=seasonal_order)
result = model.fit()

# Generate predictions for the next month
forecast = result.get_forecast(steps=len(test))
forecast_values = forecast.predicted_mean

# Evaluate the model
mse = mean_squared_error(test['messages_count'], forecast_values)
rmse = np.sqrt(mse)

# Export the evaluation report as CSV
evaluation_report = pd.DataFrame({
    'RMSE': [rmse]
})
evaluation_report.to_csv('evaluation_report.csv', index=False)

# Combine actual and predicted values into a DataFrame
forecast_df = pd.DataFrame({
    'Date': test.index,
    'Actual': test['messages_count'].values,
    'Predicted': forecast_values.values
})

# Export the actual and predicted values as CSV
forecast_df.to_csv('forecast_values.csv', index=False)
```

In this case, the model uses historical monthly message count data from the Ocean Protocol Discord server to forecast future activity. SARIMAX models consist of several components: autoregressive (AR), differencing (I), moving average (MA), and seasonal (S) terms. These components capture different aspects of the time series data, such as trend, seasonality, and noise. By specifying appropriate orders for these components, the model can effectively capture the underlying patterns in the data.

The rationale for choosing SARIMAX for this task lies in its ability to handle time series data with complex patterns, including seasonality and trends. The monthly message count in the Discord server likely exhibits seasonal patterns due to periodic events or trends in user activity. SARIMAX can capture these seasonal fluctuations and provide accurate forecasts based on historical patterns.

Furthermore, SARIMAX is a well-established and widely used model in time series analysis, with robust mathematical foundations and implementations in libraries like statsmodels. Its flexibility and ability to incorporate exogenous variables make it suitable for various forecasting tasks, including predicting server activity based on historical message counts.

The forecasting model developed for Ocean Protocol Discord user activity can be adapted and applied to similar data structures across various projects with some adjustments and considerations. First, the model architecture, including the choice of time series forecasting methods like ARIMA or SARIMA, remains applicable as long as the data exhibits temporal dependencies and seasonality patterns.

Second, the feature engineering process, which involves identifying relevant predictors and extracting informative features from the data, can be tailored to suit the specific context of the new project. Third, the model training and evaluation pipeline, which includes data splitting, model fitting, and performance evaluation, can be generalized and automated using standard libraries like scikit-learn or statsmodels.

Fourth, domain-specific knowledge and insights should guide the selection of input features and the interpretation of forecasting results to ensure the model captures relevant dynamics unique to the target domain. Lastly, continuous monitoring and

refinement of the model are essential to adapt to changing data patterns and ensure its effectiveness over time. By following these principles and adapting the model to the specific requirements of each project, organizations can leverage forecasting techniques to make informed decisions and anticipate future trends across a wide range of applications and industries.

Conclusion

The analysis conducted throughout this challenge offers valuable insights into the community engagement and trends observed within Ocean Protocol's Discord server. By examining message counts, user activity, technical issues, and spam detection, a comprehensive understanding of the community's dynamics emerges. The identification of top contributors, analysis of message content, and detection of technical issues shed light on the server's overall health and activity levels.

Furthermore, the predictive modeling of future activity patterns provides stakeholders with proactive measures to anticipate and accommodate shifts in community engagement. Through these analyses, Ocean Protocol gains actionable insights into user behavior, allowing for targeted interventions to enhance community engagement and address emerging issues promptly. The identification of technical issues and spam detection mechanisms contribute to maintaining a positive user experience and fostering a vibrant community environment.

Additionally, the forecasting of future activity patterns enables Ocean Protocol to adapt its strategies and resources effectively, aligning with the evolving needs and expectations of its community members. Overall, this challenge serves as a foundation for ongoing monitoring, analysis, and optimization of community engagement efforts within Ocean Protocol's Discord server, fostering collaboration, innovation, and growth within its ecosystem.

Appendix

User Engagement Analysis:

- Utilized message counts to gauge user activity levels.
- Identified top contributors and their contributions.
- Analyzed message distribution over days of the week for peak activity insights.

Content Analysis:

- Examined message content to categorize technical issues, spam, and user engagement.
- Developed models to classify spam messages and detect technical issues.
- Categorized technical issues into user-related, system-related, and external factors.

Forecasting Model:

- Developed a forecasting model to predict future server activity (monthly message count).
- Evaluated the model's performance and identified areas for improvement.
- Generated insights into community dynamics and trends on Ocean Protocol's Discord server.

Conclusion:

- The analysis offers valuable insights into user engagement, content dynamics, and community trends.
- Predictive modeling enables proactive measures to adapt strategies and resources effectively.
- Ongoing monitoring and analysis foster collaboration, innovation, and growth within the community.

For access to the codebase and detailed implementation, please visit the GitHub repository: Ocean Protocol Discord Analysis.